NAME

archive − Sets archive attributes and schedules files for immediate archiving

SYNOPSIS

archive [−**C**] [−**d**] [−**f**] [−**n**] [−**w**] *filename* . . .

archive [−**C**] [−**d**] [−**f**] [−**n**] [−**w**] −**r** *dirname* . . . [*filename* . . .]

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **archive** command sets archive attributes on files and directories. It also specifies archiving for one or more files.

By default, a file is archived some time after its creation. Your site's default archiving operation is configured by the system administrator. If neither the −**d** nor the −**n** options are specified, files are marked to be archived immediately.

When archive attributes are set on a directory, all files or directories subsequently created in that directory inherit those attributes.

OPTIONS

This command accepts the following arguments:

−**C** Specifies concurrent archiving, which means that a file can be archived even if opened for write. The archive time is regulated by the modification time. By default, archiving is disallowed while a file is opened for write. Note that NFS files are not opened and are concurrently archived by default.

Concurrent archiving is useful for databases, however caution is advised because archiving can occur while the file is being modified. This can result in wasted media.

−**d** Resets the archive attributes on a file to the default attributes. When this option is specified, attributes are first reset to the default, and then all other attribute-setting options are processed. The only action taken is that attributes are reset. No archiving is performed.

−**f** Suppresses error messages.

−**n** Disables archiving for a file. This option specifies that a file never be archived. Only a superuser can set this attribute on a file. When this option is specified, the only action taken is that the attribute is set.

This option cannot be specified for a file that has either the checksum generate or checksum use attributes set. These attributes are set by using the **ssum**(1) command's −**g** and −**u** options, respectively. For more information on **ssum**(1), see the **ssum**(1) man page.

−**w** Waits for a file to have at least 1 archive copy before completing. This option cannot be specified on the command line in conjunction with the −**d** or −**n** options. Note that it may take a long time for a file to be archived.

Note that when archiving many files at once (such as with **archive -r -w .**) the "-w" option causes each file to be completely archived before the archive request for the next file is issued. In order to get the best performance in this situation, do the following:

```
archive -r .
archive -r -w .
```

−**r** *dirname* . . .
Recursively archives or sets attributes for files contained in the specified *dirname* and its subdirectories. More than one *dirname* can be specified.

If used in conjunction with other command line options, the **−r** *dirname* option must be specified prior to any individual files listed (using the *filename* argument), but it must be specified after any other individual options.

filename ...
Specifies one or more file names. If the **−r** *dirname* option is also specified, individual *filename* arguments must appear after all *dirname* specifications.

EXAMPLES

The following command resets all attributes to the default settings on all files in the current directory and all files in subdirectories beneath:

```
archive -d -r .
```

SEE ALSO

ssum(1), **stage(1), release(1).**

NAME

release − Releases disk space and sets release attributes

SYNOPSIS

release [−**a**] [−**d**] [−**f**] [−**n**] [−**p**] [−**s** *partial_size*] [−**V**] *filename* . . .

release [−**a**] [−**d**] [−**f**] [−**n**] [−**p**] [−**s** *partial_size*] [−**V**] −**r** *dirname* . . . [*filename* . . .]

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **release** command sets release attributes for a file and releases the disk space associated with one or more files. At least one archive image must exist for each file before its disk space is released. By default, the releaser daemon automatically drops disk space when the file system's high water mark is reached.

If the −**a**, −**d**, −**n**, −**p**, or −**s** options are specified, only the attribute is set; the disk space is not released.

When release attributes are set on a directory, files and directories subsequently created in that directory inherit those attributes.

OPTIONS

This command accepts the following arguments:

−**a** Sets the attribute that specifies that a file's disk space be released when at least one archive copy of the file exists. This option cannot be specified on the command line in conjunction with the −**n** option.

−**d** Resets the release attributes on the file to the default attributes. When this option is specified, attributes are first reset to the default, and then all other attribute-setting options are processed.

If the *partial* attribute is reset, all blocks are released for an offline regular file.

−**f** Suppresses error messages.

−**n** Specifies that the disk space for this file never be released. Only a superuser can set this attribute on a file. This option cannot be specified on the command line in conjunction with the −**a** or −**p** options.

−**p** Sets the partial release attribute on the file so that when the file's disk space is released, the first portion of that disk space is retained on the disk.

By default, the minimum size of the portion retained on disk is controlled by the −**o partial=***n***k** option on the **mount_samfs**(1M) command. This amount can be adjusted by using the −**s** option on the **release** command.

If this option is specified for an offline file, the partial blocks are not on the disk, and the entire file is be staged if accessed. You can use the **stage**(1) command's −**p** option to stage the partial blocks to the disk.

This option cannot be specified under the following circumstances:

- This option cannot be specified on the command line in conjunction with the −**n** option.
- This option cannot be specified for a file that has either the checksum-generate or checksum-use attributes set. These attributes are set by using the **ssum**(1) command's −**g** or −**u** options, respectively.
- The database license key allows you to specify the **release** command's −**p** option in

conjunction with the **stage**(1) command's **−n** option. The **stage**(1) command's **−n** option enables the never-stage attribute.

If the database license key is disabled and the never-stage attribute is set, this option cannot be specified for the file. For more information on the **stage**(1) command, see the **stage**(1) man page.

−s *partial_size*
Specifies the number of kilobytes to be retained on disk when a file with the partial-release attribute is released. When the file's disk space is released, the first *partial_size* kilobytes of that disk space are retained.

By default, the minimum *partial_size* is 8 kilobytes, and the maximum *partial_size* is 16 kilobytes or whatever the **−o maxpartial=***maxpartial* setting is for this file system as specified on the **mount**(1M) command. For more information on the **mount**(1M) command, see the **mount_samfs**(1M) man page.

This option cannot be specified under the following circumstances:

- This option cannot be specified on the command line in conjunction with the **release** command's **−n** option.
- This option cannot be specified for a file that has either the checksum-generate or checksum-use attributes set. These attributes are set by using the **ssum**(1) command's **−g** or **−u** options, respectively.
- The database license key allows you to specify the **release** command's **−p** option in conjunction with the **stage**(1) command's **−n** option. The **stage**(1) command's **−n** option enables the never-stage attribute.

−r Recursively releases disk space or sets release attributes for files contained in the specified *dirname* and its subdirectories. More than one *dirname* can be specified.

If used in conjunction with other command line options, the **−r** *dirname* option must be specified prior to any individual files listed (using the *filename* argument), but it must be specified after any other individual options.

−V Enables a detailed, verbose display. A message is displayed for each file for which release is attempted.

filename Specifies one or more file names. If the **−r** *dirname* option is also specified, *filename* arguments must appear after all *dirname* specifications.

SEE ALSO

archive(1), **ssum**(1), **stage**(1).

mount_samfs(1M).

NAME

request − Create a removable-media file

SYNOPSIS

request −m *media* [**−v** *vsn1[/vsn2/vsn3/...]*] [**−p** *position1[/position2/position3/...]* | **−l** *vsnfile*] [**−N**] [**−s** *size*] [**−f** *file_id*] [**−n** *version*] [**−o** *owner*] [**−g** *group*] [**−i** *information*] *file*

AVAILABILITY

SUNWsamfs

DESCRIPTION

request creates a removable-media file allowing access to either tape or optical disk. The media is specified by the required parameter **-m.** (see **mcf**(4)). The Volume Serial Name(s) of the media may be specified by the parameter **-v** which is the list of vsns or the parameter **-l.** which is the name of a file which contains the vsns. Multiple vsns for the **-v** parameter are specified by separating them with "/". Multiple vsns for the **-l** parameter are specified as one vsn per line (optionally followed by whitespace and a position on the same line) in the vsnfile. For tape, each *vsn* is limited to 6 characters. For optical media the limit is 31 characters. *file* is the file path name of the removable-media file to be created. The file must reside on a Sun SAM-FS or Sun SAM-QFS file system. Subsequent access to the file results in access to the specified removable-media.

When a removable-media file is written by opening it with oflag equal to O_WRONLY, O_RDWR, O_CREAT, or O_TRUNC, the removable-media information in the Sun SAM-FS or Sun SAM-QFS file system is updated to reflect the data's position on the medium. Subsequent read access (open with oflag equal to O_RDONLY) to the file will result in access to the data written during creation.

OPTIONS

−s *size*
The required size in bytes. When *file* is opened for write access, sufficient space on the media must be available before the first write is done.

−p *position*
The position of the removable media file on the media, specified in decimal (or hexadecimal if preceded by "0x"). If a vsnfile is supplied, the position is specified after the vsn on the same line, separated by whitespace. Note that Sun SAM-FS and Sun SAM-QFS utilities usually print the position of the file on the medium in hexadecimal. If specified, the media is positioned to *position* on each vsn. The number of positions must match the number of vsns. You must be super-user to set position.

TAPE OPTIONS

−N
The tape is a foreign tape (not written in a Sun SAM-FS nor Sun SAM-QFS environment), is barcoded, write protected and is opened for read access only. It is positioned to 0. This option requires the foreign tape license option.

OPTICAL MEDIA OPTIONS

−f *file_id*
The recorded file name of the file to access (up to 31 characters). The default is the file name portion (basename) of the path specified by *file*. For requests where *file* is greater than 31 characters, no default exists and the **−f** parameter is required.

−n *version*
Version number of the file. If *version* is 0, the most current version will be used for read access and a new version will be created for write access. The default value is **0**. For write access, the *file* will be updated with the new version number.

−o *owner*
Owner identifier (up to 31 characters). The default is the current user.

−g *group*
Group identifier (up to 31 characters). The default is the user's current group.

−i *information*
User information string. The information string is written in the file's label at creation time (up to 159 characters).

EXAMPLE

Here is an example for disaster recovery of a tape-resident archive file at position 286 hexadecimal on DLT volume YYY:

```
request -m lt -v YYY -p 0x286 /sam1/xxx
```

Here is an example using multiple VSNs:

```
request -m lt -v YYY/VVV/WWW -p 0x286/0x3f07/0x0x4 /sam1/xox
```

Here is the same example using the −*l* option:

```
request -m lt -l vsns /sam1/xox
```

file "vsns":

```
YYY 0x286
VVV 0x3f07
WWW 0x0x4
```

NOTE

Removable-media files are not supported over NFS.

For optical disk files that are to be used to read archive images, the group (−**g**) must be the group **sam_archive**, and the owner (−**o**) must be **sam_archive**.

For tape files, each write to the media results in one tape block. Each read of the media returns a tape block, or the first buffer-size bytes of the tape block, whichever is smaller. The buffer size must be equal to or larger than the tape block in order to read the entire block.

SEE ALSO

basename(1), **open**(2), **mcf**(4)

NAME

sdu − summarize disk usage

SYNOPSIS

sdu [−abchklsxDLS] [−−all] [−−total] [−−count-links] [−−summarize] [−−bytes] [−−human-readable] [−−kilobytes] [−−one-file-system] [−−separate-dirs] [−−dereference] [−−dereference-args] [−−help] [−−version] [filename...]

DESCRIPTION

This manual page documents the GNU version of **du** as enhanced by Sun Microsystems for SAM-FS and SAM-QFS. **sdu** displays the amount of disk space used by each argument and for each subdirectory of directory arguments. The space is measured in 1K blocks by default, unless the environment variable POSIXLY_CORRECT is set, in which case 512-byte blocks are used. **sdu** displays actual disk blocks for online SAM-FS and SAM-QFS files, and an estimate of disk blocks (based on file size) for offline SAM-FS and SAM-QFS files. To get actual disk block usage for both online and offline files, use **du**(1).

OPTIONS

−a, −−all
Display counts for all files, not just directories.

−b, −−bytes
Print sizes in bytes.

−c, −−total
Write a grand total of all of the arguments after all arguments have been processed. This can be used to find out the disk usage of a directory, with some files excluded.

−h, −−human-readable
Print sizes in human readable format (e.g. 1K, 234M, 2G)

−k, −−kilobytes
Print sizes in kilobytes. This overrides the environment variable POSIXLY_CORRECT.

−l, −−count-links
Count the size of all files, even if they have appeared already in another hard link.

−s, −−summarize
Display only a total for each argument.

−x, −−one-file-system
Skip directories that are on different filesystems from the one that the argument being processed is on.

−D, −−dereference-args
Dereference symbolic links that are command line arguments. Does not affect other symbolic links. This is helpful for finding out the disk usage of directories like /usr/tmp where they are symbolic links.

−L, −−dereference
Dereference symbolic links (show the disk space used by the file or directory that the link points to instead of the space used by the link).

−S, −−separate-dirs
Count the size of each directory separately, not including the sizes of subdirectories.

−−help Print a usage message on standard output and exit successfully.

−−version
Print version information on standard output then exit successfully.

NAME

segment − Sets segment file attributes

SYNOPSIS

segment [−**d**] [−**f**] [−**s** *stage_ahead*] [−**V**] −**l** *segment_size filename* . . .

segment [−**d**] [−**f**] [−**s** *stage_ahead*] [−**V**] −**l** *segment_size* −**r** *dirname* . . . [*filename* . . .]

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **segment** command sets the segment attribute for an existing file. At a minimum, the −**l** *segment_size* and the *filename* must be specified. If a file is segmented, it is archived to and staged from its volumes in *segment_size* chunks.

When file attributes are set on a directory, files and directories subsequently created in that directory inherit those attributes.

OPTIONS

−**d** Returns the segment attributes on the file to the default. When −**d** is specified, attributes are first reset to the default, then other attribute-setting options are processed. It not possible to reset a file that has already been segmented.

−**f** Suppresses errors.

−**l** *segment_size*
Specifies the segment size. The *segment_size* must be an integer and must be greater than or equal to one megabyte. The integer specified must be followed by **k** (for kilobytes), **m** (for megabytes), or **g** (for gigabytes). For example:

−**l 1024k**

This segment size specifies the size at which the file is segmented on the file system for archiving and staging. A file is segmented when it reaches the specified segment size. If a file has already been segmented, the segment size cannot be changed. A pre-existing file cannot be segmented if it exceeds the specified segment size.

−**s** *stage_ahead*
Specifies the number of segments to stage ahead when staging a segmented file. This means when an offline segment is read, in addition to staging the current segment, the next *stage_ahead* segments are also staged. The default value of *stage_ahead* is zero, which means there is no stage read ahead. The maximum *stage_ahead* value is 255.

−**r** Recursively sets the **segment** file attribute for all files contained in the specified *dirname* or its subdirectories.

−**V** Enables the verbose display. Displays a message for each file on which attributes are set.

NOTES

The −**drives** directive in the **archiver.cmd** file specifies the number of drives that should be used for archiving and staging.

A segmented file is automatically striped across several volumes when it is archived if the following conditions are in effect:

- More than one drive is available.
- The −**drives** directive is in effect.

A segmented file is automatically striped from several volumes when it is striped if the following conditions are in effect:

- The file was archived as striped.
- More than one drive is available.
- The −**drives** directive is in effect.

SEE ALSO

stage(1), **archive**(1), **archiver.cmd**(4)

NAME

setfa − set file attributes

SYNOPSIS

setfa [−**d**] [−**f**] [−**D**] [−**g** *stripe_group*] [−**l** *length[***k**|**m**|**g**] [−**s** *stripe*] [−**V**] *filename*. . .

setfa [−**d**] [−**f**] [−**D**] [−**g** *stripe_group*] [−**l** *length[***k**|**m**|**g**] [−**s** *stripe*] [−**V**] −**r** *dirname*. . . [*filename*. . .]

AVAILABILITY

SUNWsamfs

DESCRIPTION

setfa sets attributes for a new or existing file. The file is created if it does not already exist.

When file attributes are set on a directory, files and directories subsequently created in that directory inherit those attributes.

OPTIONS

−d Return the file attributes on the file to the default. When **−d** is specified attributes are first reset to the default, then other attribute-setting options are processed.

−f Do not report errors.

−D Specifies the direct I/O attribute be permanently set for this file. This means data is transferred directly between the user's buffer and disk. This attribute should only be set for large block aligned sequential I/O. The default I/O mode is buffered (uses the page cache). Directio will not be used if the file is currently memory mapped. See man directio(3C) for Solaris 2.6 and above for more details, however the Sun SAM-FS and Sun SAM-QFS directio attribute is permanent.

−g *stripe_group*

Specifies the number of the striped group in which the file is to be first allocated. *stripe_group* is a number 0 .. 127, and must be a stripe group defined in the file system. If round robin is set (see option **-s** below), the file is completely allocated on the designated stripe group.

Note, the *stripe_group* attribute is inherited. It is possible to create a directory and set a stripe group for that directory; then all files created in that directory are allocated on the specified stripe group. For example:

```
setfa -g 0 -s 0 audio
setfa -g 1 -s 0 video
```

Files created in **audio** are allocated on striped group **0** and files created in **video** are allocated on stripe group **1**.

−l *length*

Specifies the number of bytes to be preallocated to the file. This can only be applied to a file with no disk blocks assigned. This option is ignored for a directory. If an I/O attempts to extend a preallocated file, the caller will get an ENXIO error. If an attempt is made to preallocate a file with disk blocks assigned, or a segmented file, the caller will get an EINVAL error.

−s *stripe*

Specifies the number of allocation units to be allocated before changing to the next unit. If *stripe* is 1, this means the file will stripe across all units with 1 disk allocation unit (DAU) allocated per unit. If *stripe* is 0, this means the file will be allocated on one unit until that unit has no space. The default stripe is specified at mount. (see **mount_samfs (1M)**). The error "Invalid argument" is returned if the user sets stripe > 0 and mismatched stripe groups exist. Mismatched stripe groups means all striped groups do not have the same number of partitions. Striping across mismatched stripe groups is not allowed.

−r Recursively performs the operation (setting file attributes) for any files contained in the specified *dirname* or its subdirectories.

−V Turns on verbose display. A message will be displayed for each file on which attributes are set.

SEE ALSO

archive(1), **release**(1), **ssum**(1), **stage**(1).

mount_samfs(1M).

NAME

sfind − Search for files in a directory hierarchy

SYNOPSIS

sfind [path...] [expression]

AVAILABILITY

SUNWsamfs

DESCRIPTION

This manual page documents the Sun Microsystems version of the GNU version of **find**. The GNU version of **find** is modified to support the features of the Sun SAM-FS and Sun SAM-QFS file system. The following added tests reference characteristics of files resident on a Sun SAM-FS or Sun SAM-QFS file system:

> any_copy_d, any_copy_r, archdone, archived, archive_d, archive_n, copies, copy, copy_d, copy_r, damaged, mt, mt1, mt2, mt3, mt4, offline, online, partial_on, release_d, release_a, release_n, release_p, rmin, rtime, ssum_g, ssum_u, ssum_v, stage_a, stage_d, stage_n, vsn, vsn1, vsn2, vsn3, vsn4, xmin, xtime

sfind searches the directory tree rooted at each given file name by evaluating the given expression from left to right, according to the rules of precedence (see section OPERATORS), until the outcome is known (the left hand side is false for *and* operations, true for *or*), at which point **find** moves on to the next file name.

The first argument that begins with '−', '(', ')', ',', or '!' is taken to be the beginning of the expression; any arguments before it are paths to search, and any arguments after it are the rest of the expression. If no paths are given, the current directory is used. If no expression is given, the expression '−print' is used.

sfind exits with status 0 if all files are processed successfully, greater than 0 if errors occur.

EXPRESSIONS

The expression is made up of options (which affect overall operation rather than the processing of a specific file, and always return true), tests (which return a true or false value), and actions (which have side effects and return a true or false value), all separated by operators. −and is assumed where the operator is omitted. If the expression contains no actions other than −prune, −print is performed on all files for which the expression is true.

OPTIONS

All options always return true.

−daystart
: Measure times (for −amin, −atime, −cmin, −ctime, −mmin, and −mtime) from the beginning of today rather than from 24 hours ago.

−depth
: Process each directory's contents before the directory itself.

−follow
: Dereference symbolic links. Implies −noleaf.

−maxdepth *levels*
: Descend at most *levels* (a non-negative integer) levels of directories below the command line arguments. '−maxdepth 0' means only apply the tests and actions to the command line arguments.

−mindepth *levels*
: Do not apply any tests or actions at levels less than *levels* (a non-negative integer). '−mindepth 1' means process all files except the command line arguments.

−noleaf
: Do not optimize by assuming that directories contain 2 fewer subdirectories than their hard link count. This option is needed when searching filesystems that do not follow the Unix directory-link convention, such as CD-ROM or MS-DOS filesystems or AFS volume mount points. Each

directory on a normal Unix filesystem has at least 2 hard links: its name and its '.' entry. Additionally, its subdirectories (if any) each have a '..' entry linked to that directory. When **sfind** is examining a directory, after it has statted 2 fewer subdirectories than the directory's link count, it knows that the rest of the entries in the directory are non-directories ('leaf' files in the directory tree). If only the files' names need to be examined, there is no need to stat them; this gives a significant increase in search speed.

−version
: Print **sfind** version number on standard error.

−xdev
: Don't descend directories on other filesystems.

TESTS

Numeric arguments can be specified as

+n for greater than *n*,

−n for less than *n*,

n for exactly *n*.

−amin *n*
: File was last accessed *n* minutes ago.

−anewer *file*
: File was last accessed more recently than *file* was modified. −anewer is affected by −follow only if −follow comes before −anewer on the command line.

−any_copy_d
: File has an archive copy that is damaged.

−any_copy_r
: File has an archive copy marked for rearchiving by the rearch command or the recycler.

−archdone
: The archiver has no further work to do on the file at this time. Note that this does not mean that the file has been archived.

−archive_d
: File has had the equivalent of "archive -d" run against it, and so has the default handling by the archiver.

−archive_n
: File has had the equivalent of "archive -n" run against it, and so will never be archived.

−archived
: File is archived.

−atime *n*
: File was last accessed *n*∗24 hours ago.

−cmin *n*
: File's status was last changed *n* minutes ago.

−cnewer *file*
: File's status was last changed more recently than *file* was modified. −cnewer is affected by −follow only if −follow comes before −cnewer on the command line.

−copies *n*
: File has *n* archive copies.

−copy *n*
: File has an archive copy number *n*.

−copy_d *n*

File has an archive copy number *n* that is damaged.

−copy_r *n*
: File has an archive copy number *n* marked for rearchiving by the rearch command or the recycler.

−ctime *n*
: File's status was last changed *n*∗24 hours ago.

−damaged
: File is damaged.

−empty File is empty and is either a regular file or a directory.

−false Always false.

−fstype *type*
: File is on a filesystem of type *type*. The valid filesystem types vary among different versions of Unix; an incomplete list of filesystem types that are accepted on some version of Unix or another is: ufs, 4.2, 4.3, nfs, tmp, mfs, S51K, S52K. You can use −printf with the %F directive to see the types of your filesystems.

−gid *n* File's numeric group ID is *n*.

−group *gname*
: File belongs to group *gname* (numeric group ID allowed).

−ilname *pattern*
: Like −lname, but the match is case insensitive.

−iname *pattern*
: Like −name, but the match is case insensitive. For example, the patterns 'fo∗' and 'F??' match the file names 'Foo', 'FOO', 'foo', 'fOo', etc.

−inum *n*
: File has inode number *n*.

−ipath *pattern*
: Like −path, but the match is case insensitive.

−iregex *pattern*
: Like −regex, but the match is case insensitive.

−links *n*
: File has *n* links.

−lname *pattern*
: File is a symbolic link whose contents match shell pattern *pattern*. The metacharacters do not treat '/' or '.' specially.

−mmin *n*
: File's data was last modified *n* minutes ago.

−mt *media-type*
: File has an archive copy on the specified *media-type* on any copy.

−mt1 *media-type*

−mt2 *media-type*

−mt3 *media-type*

−mt4 *media-type*
: File has an archive copy on the specified *media-type* for the indicated copy number (1-4).

−mtime *n*
: File's data was last modified *n*∗24 hours ago.

−name *pattern*
: Base of file name (the path with the leading directories removed) matches shell pattern *pattern*. The metacharacters (‘∗’, ‘?’, and ‘[]’) do not match a ‘.’ at the start of the base name. To ignore a directory and the files under it, use −prune; see an example in the description of −path.

−newer *file*
: File was modified more recently than *file*. −newer is affected by −follow only if −follow comes before −newer on the command line.

−nouser
: No user corresponds to file’s numeric user ID.

−nogroup
: No group corresponds to file’s numeric group ID.

−offline File is offline.

−online File is online.

−partial_on
: File has the partial-release attribute set and the partially retained portion of the file is online.

−path *pattern*
: File name matches shell pattern *pattern*. The metacharacters do not treat ‘/’ or ‘.’ specially; so, for example,

```
sfind . −path ’./sr∗sc’
```

will print an entry for a directory called ’./src/misc’ (if one exists). To ignore a whole directory tree, use −prune rather than checking every file in the tree. For example, to skip the directory ‘src/emacs’ and all files and directories under it, and print the names of the other files found, do something like this:

```
sfind . −path ’./src/emacs’ -prune -o -print
```

−perm *mode*
: File’s permission bits are exactly *mode* (octal or symbolic). Symbolic modes use mode 0 as a point of departure.

−perm −*mode*
: All of the permission bits *mode* are set for the file.

−perm +*mode*
: Any of the permission bits *mode* are set for the file.

−regex *pattern*
: File name matches regular expression *pattern*. This is a match on the whole path, not a search. For example, to match a file named ‘./fubar3’, you can use the regular expression ‘.∗bar.’ or ‘.∗b.∗3’, but not ‘b.∗r3’.

−release_d
: File has had the equivalent of "release -d" run against it, and thus has the default release handling.

−release_a
: File has had the equivalent of "release -a" run against it, and thus has will be released immediately after being archived.

−release_n
: File has had the equivalent of "release -n" run against it, and thus will never be released.

−release_p
: File has had the equivalent of "release -p" run against it, and thus will be partially released.

−rmin *n*

File's residence was changed *n* minutes ago.

−rtime *n*
: File's residence was changed *n*∗24 hours ago.

−size *n*[bcgkmt]
: File uses *n* 512-byte blocks (bytes if 'b' or 'c' follows *n*, kilobytes if 'k' follows *n*, megabytes if 'm' follows *n*, gigabytes if 'g' follows *n*, terabytes if 't' follows *n*). The size does not count indirect blocks, and does count blocks in sparse files that are not actually allocated.

−ssum_g
: File has had the equivalent of "ssum -g" run against it, and thus will have a checksum value generated and stored for it when it is archived.

−ssum_u
: File has had the equivalent of "ssum -u" run against it, and thus will have a checksum value verified (used) when it is staged.

−ssum_v
: File has a valid checksum value.

−stage_a
: File has had the equivalent of "stage -a" run against it, and thus will have associative staging behavior.

−stage_d
: File has had the equivalent of "stage -d" run against it, and thus will have the default staging behavior.

−stage_n
: File has had the equivalent of "stage -n" run against it, and thus will not be staged into disk cache for read references.

−true Always true.

−type *c* File is of type *c*:

b	block (buffered) special
c	character (unbuffered) special
d	directory
p	named pipe (FIFO)
f	regular file
l	symbolic link
s	socket
R	removable media file

−uid *n* File's numeric user ID is *n*.

−used *n*
: File was last accessed *n* days after its status was last changed.

−user *uname*
: File is owned by user *uname* (numeric user ID allowed).

−vsn *pattern*
: File has an archive copy on a volume with VSN matching shell pattern *pattern* for any copy.

−vsn1 *pattern*

−vsn2 *pattern*

−vsn3 *pattern*

−vsn4 *pattern*
: File has an archive copy on a volume with VSN matching shell pattern *pattern* for the indicated copy (1-4).

−xmin *n*
: File's data was created *n* minutes ago.

−xtime *n*
: File's data was created *n*∗24 hours ago.

−xtype *c*
: The same as −type unless the file is a symbolic link. For symbolic links, if −follow has not been given, true if the file is a link to a file of type *c*; if −follow has been given, true if *c* is 'l'. For symbolic links, −xtype checks the type of the file that −type does not check.

ACTIONS

−exec *command* ;
: Execute *command*; true if 0 status is returned. All following arguments to **sfind** are taken to be arguments to the command until an argument consisting of ';' is encountered. The string '{}' is replaced by the current file name being processed everywhere it occurs in the arguments to the command, not just in arguments where it is alone, as in some versions of **find**. Both of these constructions might need to be escaped (with a '\') or quoted to protect them from expansion by the shell.

−fprint *file*
: True; print the full file name into file *file*. If *file* does not exist when **sfind** is run, it is created; if it does exist, it is truncated. The file names ''/dev/stdout'' and ''/dev/stderr'' are handled specially; they refer to the standard output and standard error output, respectively.

−fprint0 *file*
: True; like −print0 but write to *file* like −fprint.

−fprintf *file format*
: True; like −printf but write to *file* like −fprint.

−ok *command* ;
: Like −exec but ask the user first (on the standard input); if the response does not start with 'y' or 'Y', do not run the command, and return false.

−print True; print the full file name on the standard output, followed by a newline.

−print0 True; print the full file name on the standard output, followed by a null character. This allows file names that contain newlines to be correctly interpreted by programs that process the **sfind** output.

−printf *format*
: True; print *format* on the standard output, interpreting '\' escapes and '%' directives. Field widths and precisions can be specified as with the 'printf' C function. Unlike −print, −printf does not add a newline at the end of the string. The escapes and directives are:

\a	Alarm bell.
\b	Backspace.
\c	Stop printing from this format immediately.
\f	Form feed.
\n	Newline.
\r	Carriage return.
\t	Horizontal tab.

\v	Vertical tab.
\\	A literal backslash ('\').

A '\' character followed by any other character is treated as an ordinary character, so they both are printed.

%%	A literal percent sign.
%a	File's last access time in the format returned by the C 'ctime' function.
%A*k*	File's last access time in the format specified by *k*, which is either '@' or a directive for the C 'strftime' function. The possible values for *k* are listed below; some of them might not be available on all systems, due to differences in 'strftime' between systems.

@	seconds since Jan. 1, 1970, 00:00 GMT.

Time fields:

H	hour (00..23)
I	hour (01..12)
k	hour (0..23)
l	hour (1..12)
M	minute (00..59)
p	locale's AM or PM
r	time, 12-hour (hh:mm:ss [AP]M)
S	second (00..61)
T	time, 24-hour (hh:mm:ss)
X	locale's time representation (H:M:S)
Z	time zone (e.g., EDT), or nothing if no time zone is determinable

Date fields:

a	locale's abbreviated weekday name (Sun..Sat)
A	locale's full weekday name, variable length (Sunday..Saturday)
b	locale's abbreviated month name (Jan..Dec)
B	locale's full month name, variable length (January..December)
c	locale's date and time (Sat Nov 04 12:02:33 EST 1989)
d	day of month (01..31)
D	date (mm/dd/yy)
h	same as b
j	day of year (001..366)
m	month (01..12)
U	week number of year with Sunday as first day of week (00..53)
w	day of week (0..6)
W	week number of year with Monday as first day of week (00..53)
x	locale's date representation (mm/dd/yy)
y	last two digits of year (00..99)
Y	year (1970...)

%b	File’s size in 512-byte blocks (rounded up).
%c	File’s last status change time in the format returned by the C ‘ctime’ function.
%C*k*	File’s last status change time in the format specified by *k*, which is the same as for %A.
%d	File’s depth in the directory tree; 0 means the file is a command line argument.
%f	File’s name with any leading directories removed.
%F	Type of the filesystem the file is on; this value can be used for −fstype.
%g	File’s group name, or numeric group ID if the group has no name.
%G	File’s numeric group ID.
%h	Leading directories of file’s name.
%H	Command line argument under which file was found.
%i	File’s inode number (in decimal).
%k	File’s size in 1K blocks (rounded up).
%l	Object of symbolic link (empty string if file is not a symbolic link).
%m	File’s permission bits (in octal).
%n	Number of hard links to file.
%p	File’s name.
%P	File’s name with the name of the command line argument under which it was found removed.
%s	File’s size in bytes.
%t	File’s last modification time in the format returned by the C ‘ctime’ function.
%T*k*	File’s last modification time in the format specified by *k*, which is the same as for %A.
%u	File’s user name, or numeric user ID if the user has no name.
%U	File’s numeric user ID.

A ‘%’ character followed by any other character is discarded (but the other character is printed).

−prune If −depth is not given, true; do not descend the current directory.
If −depth is given, false; no effect.

−ls True; list current file in ‘ls −dils’ format on standard output. The block counts are of 1K blocks, unless the environment variable POSIXLY_CORRECT is set, in which case 512-byte blocks are used.

OPERATORS

Listed in order of decreasing precedence:

(*expr*) Force precedence.

! *expr* True if *expr* is false.

−not *expr*
Same as ! *expr*.

expr1 expr2
And (implied); *expr2* is not evaluated if *expr1* is false.

expr1 −a *expr2*
Same as *expr1 expr2*.

expr1 −and *expr2*

Same as *expr1 expr2*.

expr1 −o *expr2*
Or; *expr2* is not evaluated if *expr1* is true.

expr1 −or *expr2*
Same as *expr1* −o *expr2*.

expr1 , *expr2*
List; both *expr1* and *expr2* are always evaluated. The value of *expr1* is discarded; the value of the list is the value of *expr2*.

EXAMPLES

Find all files in the /sam4 directory which aren't archived:

```
sfind /sam4 ! -archived
```

Find all regular files in the current directory which are archived, online, and nonzero length:

```
sfind . -archived -online ! -empty -type f -print
```

Find all regular files in the current directory which have archive copies on VSNs matching the shell pattern TP??3? . Note that shell wildcard characters must be escaped or quoted.

```
sfind . -vsn "TP??3?" -type f
```

or alternatively,

```
sfind . -vsn TP\?\?3\? -type f
```

NAME

sls − Lists directory content

SYNOPSIS

sls [−**abcdf**] [−−**full−time**] [−**g**] [−−**help**] [−**iklmnpqrstu**] [−−**version**] [−**w** *cols*] [−**x**] [−**ABCDFG**] [−**I** *pattern*] [−**KLNQRS**] [−**T** *cols*] [−**UX12**] [*file* ...]

AVAILABILITY

SUNWqfs

SUNWsamfs

DESCRIPTION

This **man**(1) page describes the Sun Microsystems extensions to the GNU version of the **ls**(1) command. Sun Microsystems modified the **ls**(1)command and added the following features to support Sun SAM-FS and Sun SAM-QFS software:

- −**D**, which lists a detailed description of each file.
- −**2**, which lists two lines of output for each file.
- −**K**, which lists all segments of a segmented file.

The **sls** command generates information for each given *file* or directory path. Directory contents are sorted alphabetically. By default, if standard output is a terminal, files are listed in columns, sorted vertically. Otherwise they are listed one per line.

The **sls** command also accepts verbose, multicharacter equivalents of many single-character options. These multicharacter options are not listed in the SYNOPSIS section of this man page, but they are noted in the option descriptions.

OPTIONS

The **sls**(1) command accepts the following options:

−a Lists all files in directories. Includes all files that start with a period (**.**). Equivalent to specifying −−**all**.

−b Quotes nongraphic characters in file names using alphabetic and octal backslash sequences like those used in C. Equivalent to specifying −−**escape**.

−c Sorts directory contents according to the file status change times instead of the modification times. If the long listing format is being used, it generates the status change time instead of the modification time. Equivalent to specifying −−**time=ctime** and −−**time=status**.

−d Lists directories like other files rather than listing their contents. Equivalent to specifying −−**directory**.

−f Does not sort directory contents. Lists them in whatever order they are stored on the disk. The same as specifying both −**a** and −**U** and disabling −**l**, −**s**, and −**t.**

−−full−time
Lists times in full, rather than using the standard abbreviation heuristics.

−g Ignored. For UNIX compatibility.

−−help
Writes a usage message to standard output and exits successfully.

−i Prints the inode number of each file to the left of the file name. If −**2** is also specified, the inode number of the directory is printed on the second line. If −**D** is also specified, the inode numbers are printed. Equivalent to specifying −−**inode**.

−k If file sizes are being listed, prints them in kilobytes. This overrides the **POSIXLY_CORRECT** environment variable. Equivalent to specifying −−**kilobytes**.

−l In addition to the name of each file, prints the file type, permissions, number of hard links, owner name, group name, size in bytes, and timestamp (the modification time unless other times are selected). For files with a time that is more than 6 months old or more than 1 hour into the future, the timestamp contains the year instead of the time of day. Equivalent to specifying **−−format=long** and **−−format=verbose**.

−m Lists files horizontally, with as many as fit on each line, separated by commas. Equivalent to specifying **−−format=commas**.

−n Lists the numeric UID and GID instead of the names. Equivalent to specifying **−−numeric−uid−gid**.

−p Appends a character that indicates the file type to each file name.

−q Prints question marks instead of nongraphic characters in file names. Equivalent to specifying **−−hide−control−chars**.

−r Sorts directory contents in reverse order. Equivalent to specifying **−−reverse**.

−s Prints the size of each file in 1-kilobyte blocks to the left of the file name. If the **POSIXLY_CORRECT** environment variable is set, 512-byte blocks are used instead. Equivalent to specifying **−−size**.

−t Sorts directory contents by timestamp instead of alphabetically. The newest files are listed first. Equivalent to specifying **−−sort=time**.

−u Sorts the directory contents according to the files' last access time instead of the modification time. If the long listing format is being used, prints the last access time instead of the modification time. Equivalent to specifying **−−time=atime**, **−−time=access**, and **−−time=use**.

−−version
Writes version information to standard output and exits successfully.

−w *cols*
Assumes the screen is *cols* columns wide. The default is taken from either the terminal driver (if possible) or the **COLUMNS** environment variable (if set). Otherwise the default is **80**. Equivalent to specifying **−−width** *cols*.

−x Lists the files in columns, sorted horizontally. Equivalent to specifying **−−format=across** and **−−format=horizontal**.

−A Lists all files in directories, except for those beginning with a period (**.**) or two periods (**..**). Equivalent to specifying **−−almost−all**.

−B In the output, suppresses files that end with a tilde (˜) unless they are specified on the command line. Equivalent to specifying **−−ignore−backups**.

−C Lists files in columns, sorted vertically. Equivalent to specifying **−−format=vertical**.

−D Uses the long-line format (**−l**) and lists a detailed description for each file. Additional lines are listed with the file attributes, archive copies, and the times. For removable media files, the output shows the media type, blocksize, the VSN(s), the sizes, and position(s).

Example:

```
server# sls -D mickey.gif
mickey.gif:
  mode: -rw-r--r--  links:   1  owner: root      group: other
  length:    319279  admin id:      7  inode: 1407.5
  offline;  archdone;  stage -n;
  copy 1: ---- May 21 10:29     1e4b1.1    lt DLT001
  access:      May 21 09:25     modification: May 21 09:25
  changed:     May 21 09:26     attributes:   May 21 10:44
  creation:    May 21 09:25     residence:    May 21 10:44
```

The first line indicates the file's mode or permissions, the number of links to the file, the owner (or user) of the file, and the group to which the owner belongs.

The second line indicates the file's length in bytes, the administrative ID number (see **samchaid(1M)**), and the inode number plus generation number.

The third line shows the file states and attributes. Possible file states, which are set by the system, are as follows:

State	**Meaning**
damaged	The file is damaged.
offline	The file is offline.
archdone	Indicates that the archiver has completed processing the file. There is no more work that the archiver can do on a file. Note that **archdone** does not indicate that the file has been archived.

Possible file attributes, which are set by the user, are as follows:

Attribute	**Meaning**
archive −n	The file is marked never archive (superuser only).
archive −C	The file is marked for concurrent archiving.
release −n	The file is marked for never release.
release −a	This file is marked for release as soon as 1 copy is made.
release −p	The file is marked for partial release. **partial=***n***k** indicates that the first *n* kilobytes of disk space are retained in disk cache for this file. **offline**/**online** indicates the first *n* kilobytes of disk space are offline/online.
stage −n	The file is marked never stage.
stage −a	The file is marked for associative staging.
setfa −D	The file is marked for direct I/O.
setfa −g*n*	The file is marked for allocation on stripe group *n*.
setfa −s*m*	The file is marked for allocation with a stripe width of *m*.
segment *n***m stage_ahead** *x*	The file is marked for segment access. **segment=***n***m** indicates *n* megabytes is the segment size. **stage_ahead=***x* indicates *x* segments will be staged ahead of the current segment.

The next line appears only for a segment index. The line is as follows:

```
segments n , offline o , archdone a , damaged d
```

In this line, *n* is the number of data segments; *o* is the number of data segments offline; *a* is the number of data segments that have met their archiving requirements; and *d* is the number of data segments that are damaged.

The archive copy line is displayed only if there is an active or stale copy. An example of archive copy line output is as follows:

```
copy 1: ---- Sep 11 10:43    3498f.1    mo OPT001
```

The first field indicates the archive copy number.

The second field consists of four dashes, as follows:

• Dash 1 indicates a stale or active entry, as follows:

Content **Meaning**

S	The archive copy is stale. This means that the file has been modified, and this archive copy is for a previous version of the file.
U	The copy has been unarchived.
−	The archive copy is active and valid.

• Dash 2 indicates the archive status, as follows:

Content	**Meaning**
r	The archiver will rearchive this copy.
−	This archive copy will not be rearchived.

• Dash 3 is unused.

• Dash 4 indicates either a damaged or undamaged status, as follows:

Content	**Meaning**
D	The archive copy is damaged. This archive copy will not be staged.
−	The archive copy is not damaged. It is a candidate for staging.

The third field shows the date and time when the archive copy was written to the media.

The fourth field contains two hex numbers separated by a period (**.**). The first hex number, **3498f**, is the position of the beginning of the archive file on the media. The second hex number is the file byte offset divided by 512 of this copy on the archive file. In this example, **1** means that this is the first file on the archive file because it is offset by 512 bytes, which is the length of the **tar**(1) header. Disk archive copies always have the entry 0.0.

The last two fields indicate the media type and the volume serial name on which the archive copy resides.

Various times are displayed for the file as follows:

Time Type	**Meaning**
access	Time the file was last accessed.
modification	Time the file was last modified.
changed	Time the information in the inode was last changed.
attributes	Time that Sun SAM-FS or Sun SAM-QFS file system attributes were last changed.
creation	Time the file was created.
residence	Time the file changed from offline to online or vice versa.

The checksum attributes are displayed on the line as follows.

```
checksum:  gen  use  val  algo:  1
```

The previous line is displayed for a file with all three checksum attributes (**generate**, **use**, and **valid**) set. If the **generate** attribute is not set, **no_gen** appears in place of **gen**. Similarly, if the **use** attribute is not set, **no_use** appears in place of **use**. **val** is displayed if a valid checksum value exists for the file; if one does not exist, **not_val** appears in place of **val**. The keyword **algo:** precedes the numeric algorithm indicator which specifies which algorithm is used when generating the checksum value.

For a removable media file, the following lines are displayed:

```
iotype: blockio  media: lt  vsns: 1 blocksize: 262144 section 0:
104071168       a358.0    CFX808
```

The first line shows the I/O type (always blockio), the media type, number of volumes, and blocksize. The second and following lines show the section length, position and offset, and VSN for each volume. There will only be one section line except in the case of volume overflow. The blocksize will be zero until the first time the volume is loaded, at which time it will be filled in with the correct value.

The **−D** option is equivalent to specifying **−−format=detailed**.

−F Suffixes each file name with a character that indicates the file type. For regular files that are executable, the suffix is an asterisk (∗). For directories, the suffix is a slash (/). For symbolic links, the suffix is an at sign (@). For FIFOs, the suffix is a pipe symbol (|). For sockets, the suffix is an equal sign (=). There is no suffix for regular files. Equivalent to specifying **−−classify**.

−G Suppresses group information in a long format directory listing. Equivalent to specifying **−−no−group**.

−I *pattern*
Suppresses files whose names match the shell pattern *pattern* unless they are specified on the command line. As in the shell, an initial period (**.**) in a file name does not match a wildcard at the start of *pattern*. Equivalent to specifying **−−ignore** *pattern*.

−K Lists all segments for a segmented file. Must be specified in conjunction with the **−2** or **−D** options.

−L Lists the files linked to by symbolic links instead of listing the content of the links. Equivalent to specifying **−−dereference**.

−N Does not quote file names. Equivalent to specifying **−−literal**.

−Q Encloses file names in double quotes and quotes nongraphic characters as in C. Equivalent to specifying **−−quote−name**.

−R Lists the content of all directories recursively. Equivalent to specifying **−−recursive**.

−S Sorts directory content by file size instead of alphabetically. The largest files are listed first. Equivalent to specifying **−−sort=size**.

−T *cols*
Assumes that each tab stop is *cols* columns wide. The default is **8**. Equivalent to specifying **−−tabsize** *cols*.

−U Does not sort directory content. Content is listed in the order it is stored in on the disk. Equivalent to specifying **−−sort=none**.

−X Sorts directory content alphabetically by file extension according to the characters after the last period (**.**). Files with no extension are sorted first. Equivalent to specifying **−−sort=extension**.

−1 Lists one line per file. Equivalent to specifying **−−format=single−column**.

−2 Lists two lines per file. The first line is identical to that obtained when you specify long format output using the **−l** option. The second line lists the file attributes, media requirements, and the creation time. Removable media files show the media type and the VSN. Nonchecksum file attributes are formatted as a string of ten characters.

The file attributes in the second line are indicated by their position, as follows:

• Position 1 - Offline/damaged status

O	The file is offline.
P	The file is offline with partial online.
E	The file is damaged.
−	The file is online.

- Position 2-4 - Archiver attributes
  - **n** Never archive the file.
  - **a** Archive the file immediately after creation or modification (see **archive**(1) to set). Ignore archive set age times. This attribute remains set until a different archive command is issued for the file (see **archive**(1)).
  - **r** The file is scheduled to be re-archived on a different volume. This attribute is set by the recycler.
  - **−** The attribute is not set.
- Position 5-7 - Releaser attributes
  - **n** Never release the file (only the superuser can set this).
  - **a** Release as soon as 1 copy is archived.
  - **p** Partially release the file. The first portion is left on disk after release.
  - **−** The attribute is not set.
- Position 8-9 - Stage attributes
  - **n** Direct access to removable media (never stage on read).
  - **a** Associatively stage this file.
  - **−** The attribute is not set.
- Position 10 - Not used. Always a dash (−).
- Position 11 - Blank space.
- Position 12-14 - Checksum attributes. Set by the **ssum**(1) command.
  - **g** Generate a checksum value when archiving.
  - **u** Checksum the file when staging.
  - **v** A valid checksum exists.
  - **−** The attribute is not set.
- Position 15-16 - Not used. Always a dash (−).
- Position 17 - Blank space.
- Position 18 - Segment attributes.
  - **s** The segment attribute is set.
  - **−** The attribute is not set.
- Position 19 - Index and segment attributes.
  These attributes do not appear if the segment attribute (position 17) is not set.
  - **S** This is a data segment.
  - **I** This is an index for a file segment. Four additional numbers contained within braces ({}) are written, as follows: {*n*, *o*, *a*, *d*}. The numbers within the braces indicate the following:
    - *n* The number of data segments in the segmented file.
    - *o* The number of data segments which are offline.
    - *a* The number of data segments which are archdone.
    - *d* The number of data segments which are damaged.
  - **−** The attribute is not set.

The next four fields indicate the media type for archive copies 1-4, if present.

Example 1. The **sls −2** command generates the following output for a nonsegmented file:

```
-rwxrwxrwx   1 smith  dev    10876  May  16 09:42  myfile
O----apn-- g-v-- -- lt
```

The preceding output shows that the file is offline and has the partial release, release after archive, and never stage attributes set. It also has the checksum generate attribute set, and a valid checksum value exists for the file. The file has copy 1 archived on **lt** (digital linear tape).

Example 2. The **sls −2** command generates the following output for a segmented file:

```
-rwxrwxrwx   1 abc  dev    10876  May 16 9:42  yourfile
---------- ----- sI {5,0,0,0} lt
```

file ...
Specifies a *file* name or full path name.

EXAMPLES

The following output is obtained from specifying **sls -D** for a file archived to disk:

```
/sam1/testdir0/filea:
  mode: -rw-r-----  links:   1  owner: root      group: other
  length:    306581  admin id:      0  inode:    11748.11
  copy 1: ---- Oct 31 13:52        15.0    dk disk01
  access:      Oct 31 13:50  modification: Oct 31 13:50
  changed:     Oct 31 13:50  attributes:   Oct 31 13:50
  creation:    Oct 31 13:50  residence:    Oct 31 13:50
```

BUGS

On BSD systems, the **−s** option reports sizes that are half the correct values for files that are NFS-mounted from HP-UX systems. On HP-UX systems, it reports sizes that are twice the correct values for files that are NFS-mounted from BSD systems. This is due to a flaw in HP-UX; it also affects the HP-UX **ls**(1) program.

SEE ALSO

archive(1), **ls**(1), **release**(1), **samchaid**(1M), **ssum**(1), **stage**(1), **tar**(1).

NAME

squota − Reports quota information

SYNOPSIS

squota [−**a**] [−**g**] [−**h**] [−**k**] [−**u**] [*file*]

AVAILABILITY

SUNWqfs

DESCRIPTION

The **squota** command displays file, block, and quota usage statistics.

Only a superuser can change quotas (see **samquota**(1M)).

By default, **squota**(1) writes the user's applicable group ID and user ID quotas and usages on all mounted Sun QFS file systems to **stdout**.

An admin set quota applies to a set of files and directories. Typically an admin set quota could be set for a large project that involves users from several groups and spans several files and directories. The admin set IDs must be assigned using the **samchaid**(1M) command. The **samchaid**(1M) command allows a system administrator to assign files and directories to individual admin sets. Admin set IDs are not tied to any set of permissions associated with the user. That is, a user can have a set of directories and files on one Sun QFS file system with a particular admin set ID, and the same user can have another set of directories and files on another file system (or even the same one) with a completely different admin set ID. A writable file is therefore used as a surrogate to determine that a user has permission to view an admin set's quota values.

OPTIONS

This command accepts the following options:

−**a**	Returns admin set quota statistics.
−**g**	Returns group quota statistics.
−**h**	Prints a brief usage summary and exits.
−**k**	Display all storage units (block quantities) in units of 1024-byte blocks. When specified, all block counts are returned in units of 1024-byte blocks.
−**u**	Returns user quota statistics.
file	Return the quota information pertaining to *file*. If *file* is writeable by the user issuing the command, information about the applicable user, group, and admin set IDs is returned. If *file* is not writeable by the user issuing the command, information about the quotas for the user's GID and UID on the filesystem that *file* resides on is returned.

EXAMPLES

Example 1. The following example is from a system upon which **/qfs1** is a mounted Sun QFS file system with group and admin set quotas enabled:

```
server% squota
                                    Limits
        Type    ID    In Use     Soft     Hard
/qfs1
Files  group   101         1     1000     1200
Blocks group   101         8    20000    30000
Grace period                    3d
No user quota entry.
```

Example 2. The following example is from the same system:

```
server% squota /qfs1/george
                                    Limits
        Type    ID    In Use     Soft     Hard
```

```
/qfs1/george
Files  admin    12         4        0        0
Blocks admin    12      6824        0        0
Grace period                      0s
---> Infinite quotas in effect.
/qfs1/george
Files  group   101         1     1000     1200
Blocks group   101         8    20000    30000
Grace period                      3d
No user quota entry.
```

EXIT STATUS

This command returns the following:

- 0 on successful completion.
- 1 on a usage or argument error.
- 10 on an execution error.

FILES

filesytem/**.quota_a**	Admin set quota information
filesystem/**.quota_g**	Group quota information
filesystem/**.quota_u**	User quota information

SEE ALSO

samquota(1M)

samfsck(1M)

passwd(4) - User ID information

group(4) - Group ID information

DIAGNOSTICS

No user quota entry.
User quotas are not active on the file system.

No group quota entry.
Group quotas are not active on the file system.

No admin quota entry.
Admin set quotas are not active on the file system.

NOTES

File system quotas are supported on Sun QFS file systems only.

NAME

ssum − Set file checksum attributes

SYNOPSIS

ssum [−**d**] [−**f**] [−**g**] [−**u**] [−**w**] *filename*. . .

ssum [−**d**] [−**f**] [−**g**] [−**u**] [−**w**] −**r** *dirname*. . . [*filename*. . .]

AVAILABILITY

SUNWsamfs

DESCRIPTION

ssum sets the checksum attributes on one or more files. If the *generate* attribute is set (−**g**), a 128-bit value is generated when the file is archived. When the file is subsequently staged, the checksum is again generated and is compared against the value generated at archive time if the *use* attribute is set (−**u**). By default, no checksum value is generated or used when archiving or staging a file.

The *generate* attribute must be set on a file before any archive copy has been made. Likewise, the selected algorithm cannot be changed after an archive copy has been made.

Direct access (**stage −n**) and partial release (**release −p**) are not allowed on a file that has either of the checksum *generate* or *use* attributes set. Also, it is not valid to specify that a file never be archived (**archive −n**) as well as specify that a checksum be generated and/or used. Therefore, when a direct access, partial release, or archive never attribute is set on a file, attempting to set the checksum *generate* or *use* attribute on the file will result in an error and the attributes will be unchanged. Similarly, when either the checksum *generate* or *use* attribute is set on a file, attempting to set a direct access, partial release, or archive never attribute on the file will result in an error and the attributes will be unchanged.

A file that has the checksum *use* attribute set cannot be memory mapped. The file also must be completely staged to the disk before access is allowed to the file's data. This means that accessing the first byte of offline data in an archived file that has this attribute set will be slower than accessing the same archived file when it does not have this attribute set. This also means that staging will operate the same way as for staging with the **−w** option for a file with the *use* attribute not set.

OPTIONS

−d Return the file's checksum attributes to the default, which turns off checksumming. Using the -d option will not reset the 'checksum valid' flag if a valid checksum has been generated for a file. The -d option deletes the checksum attributes, if no valid checksum has been generated.

−f Do not report errors.

−r Recursively set the attributes for any files contained in the specified *dirname* and its subdirectories.

−g Generate a checksum value for the file when archiving.

−u Use the checksum value for the file when staging. The *generate* attribute must have been previously set, or must be set simultaneously.

SEE ALSO

stage(1), **release**(1), **archive**(1), **sls**(1)

NAME

stage − Set staging attributes and copy off-line files to disk

SYNOPSIS

stage [−**a**] [−**c** *n*] [−**d**] [−**f**] [−**w**] [−**n**] [−**p**] [−**V**] [−**x**] *filename*. . .

stage [−**a**] [−**c** *n*] [−**d**] [−**f**] [−**w**] [−**n**] [−**p**] [−**V**] [−**x**] −**r** *dirname*. . . *[*filename*. . .]*

AVAILABILITY

SUNWsamfs

DESCRIPTION

stage sets staging attributes on a directory or file, transfers one or more off-line files from the archive media to magnetic disk, or cancels a pending or active stage request. By default, staging is automatically done when the file is accessed. If none of the −**a**, −**d**, −**n**, or −**x** options is specified, staging is initiated.

When stage attributes are set on a directory, files or directories subsequently created in that directory inherit those attributes.

Stage attributes may be set only by the owner of the file or the superuser. Staging can be initiated or canceled either by the owner, superuser, or other user with read or execute permission.

OPTIONS

−**a** Set the associative staging attribute on the file or directory. Associative staging is activated when a regular file that has the associative staging attribute set is staged. All files in the same directory that have the associative staging attribute set are staged. If a symbolic link has the associative staging attribute set, the file pointed to by the symbolic link is staged. Not valid with stage never attribute **-n**.

−**c** *n* Stage from the archive copy number *n*.

−**d** Returns staging attributes on the file to the default. When this option is specified the attributes are first reset to the default, then other attribute-setting options are processed. The only action taken is that attributes are reset.

−**f** Do not report errors.

−**w** Wait for each file to be staged back on-line before completing. Note that for files with the checksum *use* attribute set, the file must be completely staged to the disk before access is allowed to the file's data, whether or not the −**w** option is set. Not valid with −**d**, or −**n**.

Note that when staging many files at once (such as with **stage -r -w .**) the "-w" option causes each file to be completely staged before the stage request for the next file is issued. This does not allow the system to sort the stage requests in the order that the files are archived on the media. In order to get the best performance in this situation, do the following:

```
stage -r .
stage -r -w .
```

−**n** Specifies that the file never be automatically staged. The file will be read directly from the archive media. The mmap function is not supported if the stage **-n** attribute is set. The stage **-n** attribute is not valid with the associative staging attribute **-a**. The stage **-n** attribute is not valid with either of the checksum *generate* or *use* attributes (**ssum** −**g** or **ssum** −**u**). The stage **-n** attribute is mutually exclusive with the **release** −**p** attribute unless enabled by the data base license key.

−**p** Specifies that the offline regular file's partial blocks be staged.

−**r** Recursively performs the operation (staging or setting staging attributes) on any files contained in the specified *dirname* or its subdirectories.

−V Turns on verbose display. A message will be displayed for each file on which a stage will be attempted.

−x Cancel a pending or active stage request for the named file(s).

NOTE

If the application writes (see **write**(2)) to a file or the application mmaps (see **mmap**(2)) a file with prot set to PROT_WRITE, the file is staged in and the application waits until the stage has completed. The **stage −n** attribute is ignored and the file is completely staged back online.

SEE ALSO

release(1), **archive**(1), **ssum**(1), **mount_samfs(1M), mmap**(2), **write**(2)

NAME

archive_audit − Generate an archive audit

SYNOPSIS

/opt/SUNWsamfs/sbin/archive_audit [**−f** *audit_file*] [**−V**] [**−d**] [**−c** *archive_copy_number*]... *root_path*

AVAILABILITY

SUNWsamfs

DESCRIPTION

archive_audit generates an audit of all archived files and removable media files (excluding archiver removable media files) in the Sun SAM-FS or Sun SAM-QFS directory *root_path* by media type and VSN. The audit results are written to the VSN audit file. An optional summary of all archive VSNs is written to standard output.

OPTIONS

−c *archive_copy_number*
: Only archive copies for the indicated *archive_copy_number* will be examined. Multiple **-c** *archive_copy_number* options may be given; then archive copies for *any* of the *archive_copy_number*s will be examined.

−d Only damaged archive copies are listed in the VSN audit file.

−f *audit_file*
: The name of the VSN audit file. If -f is not specified, or if *audit_file* is "-", then the output is written to standard out. **Archive_audit** appends to the *audit_file*.

−V Verbose. Write the optional summary to standard output. Each file is summarized in the following format:

media VSN n `files,` *s* `bytes,` *d* `damaged copies.`

Where *media* is the media type, *VSN* is the VSN, *n* is the number of files on that VSN, and *s* is the number of bytes of data archived on that VSN. *d* is the number of damaged archive copies on that VSN.

VSN AUDIT FILE

The VSN audit file contains a 1-line entry for each section on an archived file or removable media file. Each entry has this information:

media vsn status copy section position size file

The format for the line is
"%s %s %s %d %d %llx.%llx %lld %s\n".

media is the archive media.

VSN is the archive VSN.

status is the archive copy status. *Status* is 4 dashes with 3 possible flags: S = Stale, r = rearchive, D = damaged.

copy is the number (1..4) of the archive copy residing on that VSN. or zero if the file is a removable media file,

section is the section number (0..n),

position is position and file offset.

size is the size of the file/section.

file is the path name of the archived file or the removable media file.

The following is an example of the archive_audit line.
lt DLT000 ---- 1 0 4ffd.9fa5e 169643 /sam5/QT/rainbow.sgi

The first two fields indicate the media type and the volume serial name on which the archive copy or removable media file resides.

The next field consists of four dashes as follows:

Dash 0 - Stale or active entry

S	the archive copy is stale. This means the file was modified and this archive copy is for a previous version of the file.
−	the archive copy is active and valid.

Dash 1 - Archive status

r	The archiver will rearchive this copy.
−	This archive copy will not be rearchived.

Dash 3 - Damaged or undamaged status

D	the archive copy is damaged. This archive copy will not be staged.
−	the archive copy is not damaged. It is a candidate for staging.

The next field shows copy number, 1..4, for the archive copy or zero for the removable media file.

The next field shows section number, 0..n, for a multi-volume archive file or removable media file.

The first hex number, 4ffd, is the position of the beginning of the archive file on the media. The second hex number, 9fa5e, is the file byte offset divided by 512 of this copy on the archive file. For example, 1 means this is the first file on the archive file because it is offset by 512 bytes, which is the length of the tar header.

The next field shows section size (file size if only 1 section) for an archive file or the file size for a removable media file.

The last field is the name of the archive file or removable media file.

SEE ALSO

sam-archiverd(1M), **mcf**(4)

NAME

archiver − Sun SAM-FS and Sun SAM-QFS file archiver command file processor

SYNOPSIS

/opt/SUNWsamfs/sbin/archiver *directive* **[***value***]**

/opt/SUNWsamfs/sbin/archiver [−**A**] [−**c** *archive_cmd*] [−**f**] [−**l**] [−**n** *filesystem*] [−**v**]

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **archiver** command has two functions. It is used by the archiver daemon (**sam-archiverd**) to process the archiver command file. The command file used by the archiver daemon is **/etc/opt/SUNWsamfs/archiver.cmd**. This file does not have to be present for the archiver to execute. If the **archiver.cmd** file is present, however, it must be free of errors. Errors in the **archiver.cmd** file prevent the archiver from executing. If the **archiver.cmd** file is not present, all files on the file system are archived to the available removable media according to archiver defaults.

The second function allows you to use the command with the options to evaluate the archiver commands file, *archive_cmd*. No archiving is performed when the command is used in this manner. When options are used, information about archiving operations is written to standard output. It is recommended that you test your archiver commands file each time it is changed because any error found prevents the archiver from running. If an *archive_cmd* file is not specified, **/etc/opt/SUNWsamfs/archiver.cmd** is assumed.

OPTIONS

−A Turn on all list options.

−c *archive_cmd*
The name of the archiver command file to be evaluated. Default is **/etc/opt/SUNWsamfs/archiver.cmd.**

−f List file system content. Sample output:

```
Filesystems:
samfs1 eq:11 mount:/sam1 interval:300 log_file:/tmp/archiver.log

Inodes allocated: 129 (64.3k bytes)  Free inodes: 82


File type           Count  Percent   Bytes  Percent

All                    46  100.00%   40.1M  100.00%
    offline             0
    archdone           11   23.91%   10.1M   25.12%
    copy1               1    2.17%   10.0M   24.93%
    copy2               0
    copy3               0
    copy4               0

Regular                38   82.61%   40.1M   99.89%
    offline             0
    archdone            8   17.39%   10.1M   25.09%
    copy1               1    2.17%   10.0M   24.93%
    copy2               0
    copy3               0
    copy4               0
```

```
Directories              5   10.87%   32.1k    0.08%
    offline              0
    archdone             0
    copy1                0
    copy2                0
    copy3                0
    copy4                0

Removable media          3    6.52%   12.0k    0.03%
    offline              0
    archdone             3    6.52%   12.0k    0.03%
    copy1                0
    copy2                0
    copy3                0
    copy4                0
```

Column 2 is the number of files. Column 3 is the percent of the total number of files. Column 4 is the total size in bytes.

−l List input lines. Sample output:

```
1: logfile = /tmp/archiver.log
2: interval = 5m
3: big . -minsize 500k
4: all .
5:      1 30s
6: vsns
7: samfs1.1 mo .*
8: all.1    mo .*
9: big.1    lt .*
```

−n *filesystem*
List file system content (same as **−f**) for a single filesystem.

−v List VSNs. Sample output:

```
device:mo20 drives_available:1 archive_drives:1
  Catalog:
  mo.mo0002               capacity: 621.6M space: 288.4M
device:lt30 drives_available:1 archive_drives:1
  Catalog:
  lt.LT0001               capacity:   9.5G space:   8.5G

.
.
.

Archive sets:
all.1
 media: mo
 VSNs:
   mo0002
big.1
 media: lt
```

```
                VSNs:
                  LT0001
               samfs1.1
                media: mo
                VSNs:
                  mo0002
```

Sample default output:

```
Reading archiver command file "example1.cmd"

Notify file: /opt/SUNWsamfs/sbin/archiver.sh

Archive media:
media:lt archmax:  512.0M Volume overflow not selected  not available
media:mo archmax:    4.8M Volume overflow not selected

Archive libraries:
Device:tp31 drives_available:1 archive_drives:1

Device:mo20 drives_available:1 archive_drives:1

Archive file selections:
Filesystem samfs1  Logfile: /tmp/archiver.log
samfs1 File system data
    copy:1  arch_age:240
big  path:. minsize:502.0k
    copy:1  arch_age:240
all  path:.
    copy:1  arch_age:30


Archive sets:
allsets

all.1
 media: mo
 Total space available:  288.4M

big.1
 media: lt
 Total space available:    8.5G

samfs1.1
 media: mo
 Total space available:  288.4M
```

SEE ALSO

archiver.cmd(4), **sam-archiverd**(1M), **sam-arcopy**(1M), **sam-arfind**(1M)

NAME

auditslot − Audit slots in a robot

SYNOPSIS

/opt/SUNWsamfs/sbin/auditslot [**−e**] *eq*:*slot*[:*partition*] [*eq*:*slot*[:*partition*]...]

AVAILABILITY

SUNWsamfs

DESCRIPTION

auditslot will send a request to the robot specified by the equipment identifier *eq* to audit the media in the specified *slot*. The *slot*s must be in use and occupied (that is, the media cannot be mounted in a drive). If *slot* contains a two-sided optical cartridge, then both sides will be audited.

OPTIONS

−e If *slot* is tape, skip to EOD and update space available. **Caution:** Skip to EOD is not interruptible and under certain conditions can take hours to complete.

FILES

mcf	The configuration file for Sun SAM-FS and Sun SAM-QFS environments

SEE ALSO

export(1M), **import**(1M), **move**(1M), **mcf**(4), **sam-robotsd**(1M)

NAME

backto331.sh − move files and convert catalogs to their SAM-FS 3.3.1 states

SYNOPSIS

backto331.sh

AVAILABILITY

SUNWsamfs

DESCRIPTION

backto331.sh moves selected Sun SAM-FS and Sun SAM-QFS configuration files from their 4.0 locations to their 3.3.x locations, and converts 4.0-style media catalogs to the 3.3.x format. This script may be used when changing from a Sun SAM-FS or Sun SAM-QFS 4.0 system to an earlier (3.3.0 or 3.3.1) system. It should be run before the 4.0 package is removed. Since some files may have had paths or arguments added that don’t work on earlier systems, these files are not moved directly. You will need to either go back to the 3.3.x version of the file (in /etc/opt/SUNWsamfs/samfs.old) or edit the 4.0 version of the file to remove path changes and new features.

Of special note are the following files:

mcf - If you have removed the "raw" device paths, you must put them back.

defaults.conf - If you have added daemon tracing options, you must
remove them and put them back in the old place (<daemon>.cmd files).

NAME

backto350.sh − move files and convert catalogs to their SAM-FS 3.5.0 states

SYNOPSIS

backto350.sh

AVAILABILITY

SUNWsamfs

DESCRIPTION

backto350.sh moves selected Sun SAM-FS and Sun SAM-QFS configuration files from their 4.0 locations to their 3.5.0 locations, and converts 4.0-style media catalogs to the 3.5.0 format. This script may be used when changing from a Sun SAM-FS or Sun SAM-QFS 4.0 system to an earlier 3.5.0 system. It should be run before the 4.0 package is removed. Since some files may have had paths or arguments added that don't work on earlier systems, these files are not moved directly. You will need to either go back to the 3.5.0 version of the file (in /etc/opt/LSCsamfs) or edit the 4.0 version of the file to remove path changes and new features.

Of special note are the following files:

mcf - If you have removed the "raw" device paths, you must put them back.

defaults.conf - If you have added daemon tracing options, you must
remove them and put them back in the old place (<daemon>.cmd files).

NAME

build_cat − Build a media changer catalog file

SYNOPSIS

/opt/SUNWsamfs/sbin/build_cat [**−s** *size*] [**−t** *media*] *file catalog*

/opt/SUNWsamfs/sbin/build_cat [**−s** *size*] [**−t** *media*] − *catalog*

AVAILABILITY

SUNWsamfs

DESCRIPTION

build_cat will build a catalog file from *file*. If '-' is substituted for *file*, standard input will be used. If neither *file* or '-' is given, the usage message is emitted and *build_cat* exits.

Each line in the input file describes one piece of media in the catalog. The first four fields are required. The remaining fields should not be supplied except if generated by the *dump_cat* utility. Manually creating or editing of these fields can produce undesirable results.

The fields, in order, on each line are:

Index	The index of this entry within the catalog. The index must be an incrementing integer starting at zero.
vsn	The volume serial name of the media. If there is no volume serial name then the character "?" should be used.
bar code	The bar code or volser for the media. If there is no bar code then the string NO_BAR_CODE should be used.
media type	The media type for this media (see **mcf**(4)).
ptoc-fwa	The next position to be used to write data to the media.
access count	The number of times the media has been mounted.
capacity	The capacity of the device in 1024-byte units.
space avail	The amount of space left in 1024-byte units.
flags	The flags field from the catalog entry, in numeric form.
sector size	The tape block size or optical disk sector size.
label time	The time that the medium was labeled.
slot	The slot containing the volume within the automated library.
partition	The partition or side of a magneto-optical cartridge. The value of partition is 0 for tapes, 1 or 2 for m-o cartridges.
modification time	The time the medium was last modified.
mount time	The time the medium was last mounted.
reserve time	The time the volume was reserved. A value of 0 means no reservation.
reservation	The volume reservation - archive-set/owner/filessystem.

OPTIONS

−s *size* Set the initial maximum size of the catalog to *size* entries. Note, the catalog will still grow beyond this initial maximum size if more slots are added. The default initial catalog size is 32 entries.

−t *media*

Set the media type of the catalog to *media* (see **mcf**(4). If the media option is specified, the *media type* field from the input file must match the media type specified by *media*. If the media option is not specified, no enforcement of media type is performed.

STRANGE MEDIA

build_cat can be used to generate a catalog which contains a combination of usual Sun SAM-FS or Sun SAM-QFS media and so-called foreign media. Foreign media are those which are in non-SAM-FS format. The migration toolkit (SAMmigkit) provides hooks for the site to use to enable Sun SAM-FS and Sun SAM-QFS file systems to stage (and optionally re-archive) data from strange media.

When building a catalog for foreign media, the **-t** *media* option must be used to set the physical media type. For example, if the library contains DLT tapes, you would use **−t lt** on the command line. In the input file, for each volume which is strange, specify a media type beginning with ’z’.

SEE ALSO

dump_cat(1M), **export**(1M), **import**(1M), **mcf**(4), **sam-robotsd**(1M)

NAME

chmed − Set or clear library catalog flags and values

SYNOPSIS

/opt/SUNWsamfs/sbin/chmed +*flags specifier*

/opt/SUNWsamfs/sbin/chmed -*flags specifier*

/opt/SUNWsamfs/sbin/chmed -capacity *capacity specifier*

/opt/SUNWsamfs/sbin/chmed -space *space specifier*

/opt/SUNWsamfs/sbin/chmed -time *time specifier*

/opt/SUNWsamfs/sbin/chmed -count *count specifier*

/opt/SUNWsamfs/sbin/chmed -vsn *vsn specifier*

/opt/SUNWsamfs/sbin/chmed -mtype *media specifier*

AVAILABILITY

SUNWsamfs

WARNING

chmed sets or clears flags and values in a library catalog entry. These values are critical to the operation of Sun SAM-FS and Sun SAM-QFS environments and should be modified by administrators only in unusual circumstances. Administrators should exercise caution in using this powerful command, as there is no checking to ensure that the catalog remains consistent.

ARGUMENTS

These arguments are used in various combinations by the different forms of the command.

capacity is the total number of bytes contained on the volume. The capacity may be specified with ’k’, ’M’, ’G’, ’T’, ’P’, and ’E’ multipliers. e.g. 2.43G or 0.7G.

The updated capacity is interpreted in units of 1024k blocks. For example, if ’1023’ is specified, a value of 0k capacity is displayed. If ’1023k’ is specified, the updated capacity is displayed as 1023k.

The space may also be specified in octal or hexadecimal using ’0’ or ’0x’ respectively. However, fractional values and multipliers are not allowed when using octal or hexadecimal representation. For example, ’0400000’ or ’0x800000’.

count is the number of times a volume has been mounted since import, or the number of times a cleaning cartridge may be mounted before it is considered exhausted.

eq gives the equipment number (as defined in the mcf file) for the robot being operated on.

flags is a string of one or more of the following case-sensitive characters. Each character specifies one flag in the catalog entry. The characters are the same as the flags that are shown in the "flags" column of the robot VSN catalog:

A	needs audit
C	element address contains cleaning cartridge
E	volume is bad
N	volume is not in SAM format
R	volume is read-only (software flag)
U	volume is unavailable (historian only)
W	volume is physically write-protected
X	slot is an export slot
b	volume has a bar code
c	volume is scheduled for recycling
f	volume found full by archiver
i	element address in use
d	volume has a duplicate vsn

l	volume is labeled
o	slot is occupied
p	high priority volume

media specifies the media type. Valid values include (among others) **mo** and **lt**, for magneto-optical and DLT tape, respectively. See **mcf**(4) for the complete list of media types supported by Sun SAM-FS and Sun SAM-QFS file systems.

space is the total number of bytes remaining to be written on the volume. The space may be specified with 'k', 'M', 'G', 'T', 'P', and 'E' multipliers. e.g. 200.5M or 0.2005G.

The updated space is interpreted in units of 1024k blocks. For example, if '1023' is specified, a value of 0k space is displayed. If '1023k' is specified, the updated space is displayed as 1023k.

The space may also be specified in octal or hexadecimal using '0' or '0x' respectively. However, fractional values and multipliers are not allowed when using octal or hexadecimal representation. For example, '0400000' or '0x800000'.

specifier identifies the volume to be affected by the **chmed** command, in one of two forms: *media_type.vsn* or *eq:slot[:partition]*.

time is the time the volume was last mounted in a drive. Several formats are allowed for *time*. Examples are:

"2000-09-19"; "2000-07-04 20:31"; 23:05; "Mar 23"; "Mar 23 1994"; "Mar 23 1994 23:05"; "23 Mar"; "23 Mar 1994"; "23 Mar 1994 23:05".

Month names may be abbreviated or spelled out in full. Time-of-day is given in 24-hour format. Years must use all four digits. If the *time* contains blanks, the entire time must be enclosed in quotation marks.

vsn gives the VSN of the volume to be affected.

DESCRIPTION

The first form sets (+*flags*) and the second clears (-*flags*) the flags for for the given volume.

The third and fourth forms set the capacity and space, respectively, for the given volume.

The fifth form sets the last-mounted time for the volume.

The sixth form sets the mount-count value for the volume.

The final two forms sets the media type and vsn, respectively, for the given volume.

NON-SAM MEDIA

chmed can be used to modify existing catalog entries so that they denote so-called non-SAM media. Non-SAM media are those which are in non-SAM-FS format. The migration toolkit (SAMmigkit) provides hooks for the site to use to enable Sun SAM-FS and Sun SAM-QFS file systems to stage (and optionally re-archive) data from the non-SAM media.

When a non-SAM volume is imported to a library, it probably will not be found to have an ANSI-standard label. The volume's VSN will show as "nolabel". The following **chmed** commands can be used to assign a media type, VSN and non-SAM status to the volume (assuming it's in element address 5 of equipment 30):

```
chmed -mtype lt 30:5
chmed -vsn TAPE1 30:5
chmed +N 30:5
```

If you have many non-SAM cartridges, you can use build_cat to bulk-load a catalog.

EXAMPLES

```
chmed -RW lt.TAPE0
chmed +c lt.CYCLE
chmed -capacity 19.5G lt.TAPE0
chmed -space 8.2G lt.TAPE0
chmed -time "Mar 23 10:15" lt.TAPE0
chmed -time "Nov 28 1991 10:15" lt.TAPE0
chmed -vsn TAPE1 30:5
```

SEE ALSO

build_cat(1M), **mcf**(5), **sam-recycler**(1M), **robottool**(1M), **samu**(1M)

NAME

cleandrive − Clean drive in media changer

SYNOPSIS

/opt/SUNWsamfs/sbin/cleandrive *eq*

AVAILABILITY

SUNWsamfs

DESCRIPTION

cleandrive requests that tape device *eq* be loaded with a cleaning cartridge.

Sun SAM-FS and Sun SAM-QFS environments support the use of a cleaning tape, if cleaning tapes are supported by the hardware and if your media library has barcodes enabled. If you request that a tape drive be cleaned, then a cleaning tape is inserted automatically.

Cleaning tapes must have a VSN starting with the letters "CLN" in the label or must have the word "CLEAN" in the label. Multiple cleaning tapes are allowed in a system.

Cleaning tapes are only useful for a limited number of cleaning cycles. The number of remaining cycles can be viewed in the **robottool (1M)** VSN catalog display under the "count" field. Both Sun SAM-FS and Sun SAM-QFS environments track the number of cleaning cycles used for each cleaning tape. If the media changer supports the export operation, Sun SAM-FS and Sun SAM-QFS file systems will export the tape when the number of remaining cycles equals zero. A DLT cleaning tape has 20 cycles and an Exabyte cleaning tape has 10 cycles. Each time a cleaning tape is imported, the cleaning cycle is reset to the highest number of cycles for that type of tape.

FILES

mcf	The configuration file for Sun SAM-FS and Sun SAM-QFS environments

SEE ALSO

mcf(4), **sam-robotsd**(1M), **robottool**(1M)

NAME

d2format − Formats and labels an Ampex DD2 tape

SYNOPSIS

d2format −vsn *vsn_id* **−new −old** *vsn_id* **−Asize** *n* **−Bsize** *n* **−Acount** *n* **−Bcount** *n* **−format** *format* **−layout** *layout* **−slot** *n* [**−b** *blksize*] [**−erase**] [**−w**] [**−V**] *eq_ord*

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **d2format** command formats and labels an Ampex DD2 tape on the specified *eq_ord*. Several of this command's options and operands interoperate with Ampex tape features, so it is recommended that you have the *DST SCSI Tape Drive DD2 Format Guide*, which is distributed by the Ampex Corporation, accessible to you when using this command.

This command writes the following sequence of labels:

VOL1
HDR1
HDR2
tapemark
EOF1
tapemark
tapemark

The labels conform to ANSI X3.27-1987, *File Structure and Labeling of Magnetic Tapes for Information Interchange*. For tapes that are formatted with multiple partitions, a tape label is written to each partition.

The following two sections, **OPTIONS** and **OPERANDS**, describe both the optional and the required arguments, respectively, for this command.

OPTIONS

The following command line arguments are optional:

−b *blksize* Specifies the block size for this tape. Enter one of the following for *blksize*: **16**, **32**, **64**, **128**, **256**, **512**, **1024**, or **2048**. This value represents the size of the tape block in units of 1024. This option overrides the default blocksize.

−erase Erases the cartridge completely before a label is written. This is a security feature, and it is normally not necessary. Complete media erasure takes a long time to perform because all data on the media is erased. On the Ampex DD2 tapes, this option performs an initialize format of the tape prior to writing the labels.

−w Waits for the labeling operation to complete before returning a Solaris system prompt. This command can take quite awhile to complete on its own and return a prompt.

If an error occurs, it is reported along with a completion code of **1**. All labeling errors are logged to the main logging facility as defined in **/etc/syslog.conf** at installation time. NOTE: Canceling a command that is waiting for completion does not cause the operation itself to be canceled.

−V Specifies verbose mode, in which label information is written to **stdout**.

OPERANDS

The following command line arguments are required:

−vsn *vsn_id*

Specifies the volume serial name (VSN) of the tape being labeled. The VSN must be from 1 to 6 six characters in length. Valid characters for use in the *vsn_id* are as follows: the 26 uppercase letters (**A** through **Z**), the 10 digits (**0** through **9**), and the following special characters: **!"%&'()**∗**+,-./:;**<=>**?_**.

−new Specifies that the cartridge is not labeled (it is blank). Must be specified to prevent a label comparison from being made. NOTE: You must specify either **−new** or **−old**, but do not specify the **−new** option in conjunction with the **−old** option.

−old *vsn_id*
Specifies that the cartridge being labeled already has a label, so it is being relabeled. The existing *vsn_id* must be specified. During a relabeling operation, the old *vsn_id* is compared with the *vsn_id* presently on the cartridge to insure that the correct cartridge is being relabeled. NOTE: You must specify either **−old** or **−new**, but do not specify the **−old** option in conjunction with the **−new** option.

−Asize *n* Specifies the size, in double frames, of the A partitions. For information on double frames and A partitions, see your Ampex documentation.

−Bsize *n* Specifies the size, in double frames, of the B partitions. For information on double frames and B partitions, see your Ampex documentation.

−Acount *n* Specifies the number of Group A partitions to formatted on the tape. For information on A partitions, see your Ampex documentation.

−Bcount *n* Specifies the number of Group B partitions to formatted on the tape. For information on B partitions, see your Ampex documentation.

−format *format*
Specifies the type of format to use. For information on format types, see your Ampex documentation.

−layout *layout*
Specifies the tape layout. For information on layout types, see your Ampex documentation.

−slot *n* Specifies the slot in the library occupied by the cartridge to be formatted, labeled, or relabeled.

eq_ord Specifies the **Equipment Ordinal** of a library as defined in the **mcf** file.

EXAMPLES

Example 1. This example shows how to fully format a new tape. The tape resides in slot 2 of the catalog for the library. The library is identified with Equipment Ordinal 100. The tape is divided into 2 Group A partitions and 1 Group B partition.

The following command is used:

```
server#  d2format -vsn TESTER -new -slot 2 -format full \
-layout pack -Asize 235 -Bsize 0 -Acount 2 -Bcount 1 100
```

After the **d2format** command completes, the tape is formatted into 3 partitions, 2 of Group A type and 1 of Group B type. Each of the three partitions is labeled.

After this command is completed, the catalog entries look like this:

```
1     2001/04/04 13:39    1   0% -il-o-------  d2 GOATS
2:1   2001/04/04 13:40    1   0% -il-o-------  d2 TESTER:0
2:2   2001/04/04 13:40    1   0% -il-o-------  d2 TESTER:1
2:3   2001/04/04 13:40    1   0% -il-o-------  d2 TESTER:2
3     2001/04/04 13:41    4   0% -il-o-------  d2 SHEEP
4     2001/04/04 13:42    6   0% -il-o-------  d2 CATTLE
```

Example 2. The following command partially formats a **d2** tape from library 100 with a single 269-megabyte A partition:

```
server#  d2format -vsn HORSE -old GOATS -slot 1 \
-format partial -layout pack -Asize 235 -Bsize 0 \
-Acount 1 -Bcount 0 100
```

SEE ALSO

DST SCSI Tape Drive DD2 Format Guide, which is distributed by the Ampex Corporation.

NAME

devicetool − Graphical user interface for managing devices associated with Sun SAM-FS and Sun SAM-QFS file systems.

SYNOPSIS

/opt/SUNWsamfs/sbin/devicetool

AVAILABILITY

SUNWsamfs

DESCRIPTION

devicetool presents a graphical user interface for viewing information about and managing devices associated with Sun SAM-FS and Sun SAM-QFS file systems.

devicetool allows you to view information in several different ways. This is controlled by the *Format* menu, located in the upper left corner. The menu appears when you click the mouse *menu* button on the *Format* abbreviated menu button. Selecting *all* displays all configured devices. Selecting an equipment type from the *Format* menu displays configured devices of that type. For example, selecting *tapes* displays all tape devices. Selecting *configuration* displays configuration information.

To select a device from the display, click the mouse *select* button on the line displayed for that device. Selecting a device from the display allows you to perform appropriate actions on that device. The actions are represented by buttons below the display. To perform an action on a device, click the mouse *select* button on the desired action. The possible actions are: *Change State*, *Unload*, *Audit*, *Label*, and *Apply Thresholds*. **devicetool** makes available only those actions that make sense for the selected device and the users operational authorities.

Thresholds are changed by using the sliders located at the bottom of the window. New values can be set by sliding the square along the slider, or by positioning the cursor in the field to the left of the slider, typing in a new value, and then pressing return. The new thresholds are applied only when you click on the *Apply Thresholds* button.

The display can be updated by clicking on the *Update* button, located in the upper right of the window. The display will automatically update when a refresh interval greater than zero is set (in the upper right corner) and refresh is turned on. Refresh is turned on when a checkmark appears in the box immediately to the left of *refresh*; clicking on this box toggles automatic refresh on and off. By default, refresh is turned on and the interval is set to five seconds. To stop automatic refresh, toggle the refresh checkbox or set the interval to zero.

The characters in the status string, appearing from left to right, have the following meanings:

s	media is being scanned
m	the file system is mounted
M	maintenance mode
E	device received an unrecoverable error in scanning
a	device is in audit mode
l	media has a label
I	device is in idle wait
A	needs operator attention
U	unload has been requested
R	the device is reserved
w	a process is writing on the media
o	the device is open
P	the device is positioning

F	all storage slots occupied
R	device is ready and the media is read-only
r	device is spun up and ready
p	device is mounted

Operational authority is defined within **defaults.conf**(4). Users with root authority will have full access to all the functions within **devicetool**. Users who are part of the sam operator group will have access removed for certain functions. By default, the restrictions include no access to: audit (robots), label, device state (except to ON), media and format menus, max contiguous and threshold settings, and refresh. These restrictions, except for max contiguous and threshold settings, can be changed via **defaults.conf**(4).

SCREEN RESOURCES

The font for device panel lists can be changed via a resource setting. The following resource can be defined:

fontfamily

Defines the font family to be used for panel lists. An example, define the following line in the .Xdefaults resource file:

```
devicetool.fontfamily: fixed
```

FILES

mcf	The configuration file for **Sun SAM-FS and Sun SAM-QFS environments**

SEE ALSO

robottool(1M), **previewtool**(1M), **samtool**(1M), **mcf**(4), **defaults.conf**(4),

NAME

dmpshm − dump Sun SAM-FS and Sun SAM-QFS shared memory segments

SYNOPSIS

dmpshm

AVAILABILITY

SUNWsamfs

DESCRIPTION

dmpshm emits to stdout a compressed, uuencoded copy of the three Sun SAM-FS or Sun SAM-QFS shared memory segments. The output is useful only to Sun Microsystems support providers.

NAME

dump_cat − Dump the media changer catalog file in text format

SYNOPSIS

/opt/SUNWsamfs/sbin/dump_cat [**−n**] | [**−o**] [**−V**] *catalog*

AVAILABILITY

SUNWsamfs

DESCRIPTION

dump_cat writes a readable form of the *catalog* specified on the command line to standard output. See **build_cat**(1M) for the format of the output.

OPTIONS

−n Outputs the count of entries in the catalog.

−o Lists media that is no longer present in the catalog; i.e., the in-use flag is not set but there is an entry present.

−V Verbose, lists flags and label times as comments. Lists volume reservations as comments that may be used to build a ReservedVSNs file.

SEE ALSO

build_cat(1M), **sam-robotsd**(1M)

NAME

dump_log − Dump the contents of the fifo and ioctl log buffers

SYNOPSIS

/opt/SUNWsamfs/sbin/dump_log [-f] [-p *fifo_log*] [-i *ioctl_log*]

AVAILABILITY

SUNWsamfs

DESCRIPTION

dump_log dumps the contents of the fifo and ioctl log buffers.

OPTIONS

-f	This causes dump_log to run continuously; the default is to dump the circular buffer once and then terminate.
fifo_log	The log buffer of fifo commands sent by the Sun SAM-FS or Sun SAM-QFS file system to the sam-initd daemon. If none is specified the default is to use **/var/adm/log/fs_fifo_log**.
ioctl_log	The log buffer of the ioctl daemon commands sent to the Sun SAM-FS or Sun SAM-QFS file system. If none is specified the default is to use **/var/adm/log/fs_ioctl_log**.

You must have logging of the fifo and ioctl commands enabled by setting the

```
   debug   logging
```

option in the **/etc/opt/SUNWsamfs/defaults.conf** file. **dump_log** is not intended for general use, and is provided as a utility to supply Sun Microsystems analysts with troubleshooting information when necessary.

FILES

/var/adm/log/fs_fifo_log	fifo buffer used by Sun SAM-FS and Sun SAM-QFS file systems
/var/adm/log/fs_ioctl_log	ioctl buffer used by Sun SAM-FS and Sun SAM-QFS file systems

NAME

exarchive − Exchange archive copies

SYNOPSIS

exarchive [**−f**] [**−M**] **−c** *m* **−c** *n filename*. . .

exarchive [**−f**] [**−M**] **−c** *m* **−c** *n* **−r** *dirname*. . . [*filename*. . .]

AVAILABILITY

SUNWsamfs

DESCRIPTION

exarchive exchanges archive copies for one or more files or directories. Selection is controlled by the two archive copies. Copy *m* is exchanged with copy *n.*

Exactly two **−c** entries must be specified. The first copy must have a valid archive entry.

OPTIONS

−f Do not report errors.

−M Exarchive meta data only. This includes directories, the segment index, and removable media files. Regular files and symbolic links are not exarchived.

−r Recursively exchange the archive entries of the specified *dirname* and its subdirectories. The archive entries of files in the directories and subdirectories are exchanged.

SEE ALSO

unarchive(1M)

NAME

export, samexport − Export a cartridge from a robot

SYNOPSIS

/opt/SUNWsamfs/sbin/export *eq***:***slot*
/opt/SUNWsamfs/sbin/export *mediatype***.***vsn*
/opt/SUNWsamfs/sbin/samexport *eq***:***slot*
/opt/SUNWsamfs/sbin/samexport *mediatype***.***vsn*

AVAILABILITY

SUNWsamfs

DESCRIPTION

export sends a request to the library specified by *eq* to place the specified cartridge in the mail-slot of the library. For the form *mediatype.vsn*, *eq* and *slot* are determined from the catalog entry. All other volumes on the cartridge are also exported.

For the network-controlled libraries such as the GRAU using the GRAU ACI interface, IBM 3494, or STK libraries using ACSLS, this utility only removes the catalog entry for the cartridge from the catalog. Physical removal and addition of cartridges within these libraries is performed by utilities supplied by GRAU, IBM, and STK.

Volumes on cartridges exported from a library will be tracked in the **historian**(7). The historian acts as a virtual library. Volumes on cartridges that have been exported from a library will, by default, be considered available for archiving and staging activities. Operator intervention is required to provide access to exported cartridges to satisfy load requests.

See the **historian**(7) man page for details about the historian and for the default settings that control access to exported cartridges.

Note: A cartridge may be exported from the historian. The information about volumes on this cartridge will be lost.

The export and samexport commands are identical; the samexport name is provided to avoid a conflict with the Bourne shell intrinsic of the same name.

FILES

mcf	The configuration file for Sun SAM-FS and Sun SAM-QFS environments

SEE ALSO

import(1M), **build_cat**(1M), **dump_cat**(1M), **sam-robotsd**(1M), **mcf**(4), **historian**(7)

NAME

gnutar − GNU version of tar

SEE ALSO

For information about gnutar, type "/opt/SUNWsamfs/sbin/gnutar --help"

NAME

import − Imports cartridges into a library or the historian

SYNOPSIS

/opt/SUNWsamfs/sbin/import [[−**v** *volser*] | [−**c** *num* −**s** *pool*]] [−**e**] [−**l**] [−**n**] *eq*

/opt/SUNWsamfs/sbin/import −**v** *volser* | −**b** *barcode* [−**n**] −**m** *type eq*

AVAILABILITY

SUNWsamfs

DESCRIPTION

The first form of the **import** command sends a request to the automated library specified by *eq* to import media. The cartridge is placed in the first available slot in the library. For example:

```
import 27
```

The second form of the **import** command can be used only when *eq* is the Equipment Identifier of the default **historian**(7). This form adds an entry to the historian's catalog for the given *type* and the given *barcode* or *volser*. At least one of the −**b** *barcode* or −**v** *volser* identifiers must be present. For example:

```
import -b 007001 -m lt 27
```

OPTIONS

This command accepts several options. Some of the options affect only certain automated libraries. See the option descriptions and the NOTES section for information pertinent to vendor-specific automated libraries. The options for the **import** command are as follows:

−**b** *barcode*
: The barcode assigned to the cartridge. If the second form of the command is used, either a −**v** *volser* or a −**b** *barcode* option is required.

−**c** *num* −**s** *pool*
: (Network-attached StorageTek automated libraries only.)

  For StorageTek automated libraries using the first form of the **import** command, either a −**v** *volser* identifier or a −**c** *num* −**s** *pool* identifier must be used. If used, the −**c** *num* and −**s** *pool* options must be specified together.

  The −**c** *num* option specifies the number of volumes to be taken from the scratch pool specified by the −**s** *pool* option.

  The −**s** *pool* option specifies the scratch pool from which *num* volumes should be taken and added to the catalog.

−**e**
: Specifies that all newly added cartridges be audited. This includes an EOD search and updating the catalog with actual capacity and space-remaining values.

−**l**
: (Network-attached StorageTek automated libraries only.)

  The −**l** option requests that the new VSN numbers be written to standard output. If present, this option must be specified in conjunction with the −**c** *num* and −**s** *pool* options.

−**m** *type*
: The media type of the cartridge. For more information on valid media type codes, see the **mcf**(4) man page.

−**n**
: Specifies that the media is unlabeled foreign tape (not Sun SAM-FS nor Sun SAM-QFS media). It is write protected and can be only used for read access.

−**v** *volser*
: (Network-attached ADIC/GRAU, StorageTek, and IBM 3494 automated libraries only. For

the IBM 3494 library, this option is accepted only when running in shared mode; for more information, see the **ibm3494**(7) man page.)

This option creates a catalog entry with *volser* as the barcode. Physical import and export of cartridges within ADIC/Grau and StorageTek libraries are performed by utilities supplied by the vendor.

eq The Equipment Identifier as entered in the **mcf** file. For more information on the **mcf** file, see the **mcf**(4) man page.

If the first form of the **import** command is used, *eq* must be the equipment identifier of an automated libarary.

If the second form of the **import** command is used, *eq* must be the equipment number of the default historian.

NOTES

If you are using ADIC/Grau optical cartridges, it is very important to import the **A** side of barcoded optical media. Sun SAM-FS and Sun SAM-QFS media daemon query the ADIC/Grau database to find the barcode for the **A** side, and they fill in the catalog entry for the **B** side appropriately. The **A** side of optical media in the ADIC/Grau is the left side of a slot as you face the slots.

If you are using the first form of the command with a network-attached StorageTek automated library, you can identify the cartridge being imported by using either the **−v** *volser* option or by using the **−s** *pool* and **−c** *num* options together.

FILES

mcf The configuration file for Sun SAM-FS and Sun SAM-QFS environments.

SEE ALSO

export(1M), **sam-robotsd**(1M).

mcf(4).

historian(7), **ibm3494**(7).

NAME

info.sh − Sun SAM-FS and Sun SAM-QFS diagnostic report script

SYNOPSIS

/opt/SUNWsamfs/sbin/info.sh

AVAILABILITY

SUNWsamfs

DESCRIPTION

/opt/SUNWsamfs/sbin/info.sh is a script that produces a diagnostic report of the Sun SAM-FS or Sun SAM-QFS server configuration and collects log information.

info.sh should be run as root. The script builds a diagnostic report in **/tmp/SAMreport**. Upon completion, the /tmp/SAMreport should be sent to your Sun Microsystems authorized service provider or to Sun Microsystems technical support as specified in your maintenance contract.

EXAMPLE

```
neptune# info.sh
Report is being produced into /tmp/SAMreport...

Please wait............................................
Please wait............................................
Please wait............................................
Please wait............................................
Please wait.............
Please run /opt/SUNWsamfs/sbin/info.sh on any remote clients
and include the SAMreports with this one.

Please wait...

The report in /tmp/SAMreport should now be given to your
support provider.  So that you do not experience any unnecessary
delay, please DO NOT encode or compress the report in any way
when sending it.  It is a plain ASCII file and should be
handled as such.

However, if you still feel you must compress the file, use only the
standard Solaris 'compress' utility.

neptune#
```

NAME

itemize − Catalog optical disk or jukebox

SYNOPSIS

/opt/SUNWsamfs/sbin/itemize [**−file** | **f** *identifier*] [**−owner** | **u** *owner*] [**−group** | **g** *group*] [**−2**] *device*

AVAILABILITY

SUNWsamfs

DESCRIPTION

itemize generates a list of files on a specific optical disk or generates a catalog listing of disks or tapes for a robot, as specified by *device*. *device* is the device name or equipment ordinal defined in the **mcf** file.

If *device* is an optical disk, **itemize** generates a list of the files cataloged on the given optical disk. The list is generated by scanning the PTOC (partition table of contents) on the given disk. The options (see OPTIONS) apply only when using **itemize** on an optical disk. The following information is returned when itemizing an optical disk:

file ID *version* *length* *uid* *gid*

These fields contain the following information:

file ID	The name of the file.
version	The version of the file.
length	The size of the file in bytes.
uid	The users ID of the file.
gid	The group ID of the file.

If *device* is a robot, **itemize** generates a list of optical disks or tapes that are cataloged in that robot. The following information is returned.

EA *access_time* *count* *use* *ty* *vsn*

Where *EA* is the element address within the catalog, *access_time* is the last time this element address was accessed, *count* is the number of accesses, *use* is the percentage of the media already used, *ty* is the media type, and *vsn* is the volume serial name of the media. A *vsn* of **nolabel** indicates that media has been assigned to this element address but has not yet been labeled.

When itemizing a robot, a status may also be returned. This information follows the *vsn* field. The following messages may be listed.

VSN MISSING
: A medium has been assigned to this element address, the status of the element address is labeled, and the VSN is null.

SLOT VACANT
: A medium has been assigned to this element address, the element address is physically empty, and the VSN is null.

NEEDS AUDIT
: The status of the element address has the needs audit flag set.

MEDIA ERROR
: A read or write or positioning error was detected.

OPTIONS

The following options apply only when itemizing an optical disk:

−file | **f** *identifier*
: Lists only files with the specified file *identifier*.

−owner | u *owner*
Lists only files with the specified *owner*.

−group | g *group*
Lists only files with the specified *group*.

−2 Displays two lines per file, which makes more readable output for terminals.

EXAMPLES

The following example lists a catalog for an optical library with a device number of 50:

```
server# itemize 50
Robot VSN catalog: eq: 50   count: 476
EA      access_time  count  use  ty vsn
   0    Jan 22 15:57   117  76%  mo OPT000  SLOT VACANT
   1    Jan 22 17:17    86  76%  mo OPT001  SLOT VACANT
   2    Jan 22 15:57    55  76%  mo OPT002
   3    Jan 22 16:13    72  76%  mo OPT003
   4    Jan 22 16:29   807  76%  mo OPT004
   5    Jan 22 16:45    27  76%  mo OPT005
   6    Jan 22 17:01    44  76%  mo OPT006
   7    Jan 22 15:14    36   0%  mo OPT007
   8    Jan 22 15:14    43   0%  mo OPT008
   9    Jan 22 15:15    30   0%  mo OPT009
  10    Jan 22 13:32    45  99%  mo OPT010
  11    Jan 22 15:47    35  99%  mo OPT011
  12    Jan 22 15:49    43  99%  mo OPT012
  13    Jan 22 15:53    31  84%  mo OPT013
  14    Jan 22 09:44    30   0%  mo OPT014
  15    Jan 22 09:45    52   0%  mo OPT015
  16    Jan 22 15:57  2163  99%  mo OPT016
  17    Jan 22 12:29  1618   0%  mo OPT017
```

This example shows an itemize listing from an optical disk:

```
  server# itemize 20
 some.1      15     2048 sam_archive  sam_archive
 samfs1.1    67     5120 sam_archive  sam_archive
 some.1      14     1024 sam_archive  sam_archive
 samfs1.1    66     5120 sam_archive  sam_archive
 samfs1.1    65     5120 sam_archive  sam_archive
 samfs1.1    64     5120 sam_archive  sam_archive
    .
    .
    .
```

NAME

libmgr − Graphical user interface for displaying information about and managing robots, devices, and mount requests for Sun SAM-FS and Sun SAM-QFS file systems.

SYNOPSIS

/opt/SUNWsamfs/sbin/libmgr
/opt/SUNWsamfs/sbin/libmgr [**−screensize** *mode_number*] [**−locale** *locale_id*] [**−geometry** *WxH+X+Y*]

AVAILABILITY

SUNWsamfs

DESCRIPTION

libmgr presents a graphical user interface for viewing information about and managing robots, devices, and mount requests associated with Sun SAM-FS and Sun SAM-QFS environments.

Information in **libmgr** is mainly represented in the form of objects. These objects can be manipulated using a mouse. Most objects will respond as follows:

left click
selects the object

right click
brings up a menu of actions that can be performed

double click
displays detailed information on that object

The **libmgr** display is broken down into three main sections. The top section represents the robot devices. If you have no robot devices configured, this section will not be displayed. Each robot (and device) object is represented by a title, an image icon, and a short text display. The title, by default, is composed from the device's product ID, vendor ID, and the mcf equipment number. The image is a picture representation of the robot. The text display will show informational messages about the robot. When text messages appear that are longer in length that the text display, they will automatically scroll. Left clicking on the text message will cause the scrolling to stop. Any subsequent left or right click will cause the text display to scroll one window at a time to the right or left, respectively. Both the title and the image can be changed in the **SamGUI.rsc** file.

The middle section contains tab panels with the following information:

- for each robot, a display of all the drives and a media catalog display
- a historian catalog display
- a manual mount display (if needed)
- display of all the drives (if more than one set of drives exist)

The drives objects have the same display attributes as the robot objects described above. The catalog display will be composed of rows and columns. Each row represents either a piece of media or an import/export/storage element, if the robot supports such a feature. The columns are user-definable and default to: Slot, Media, Percent full, and VSN. Columns may be resized by left clicking and dragging the left or right edge of the column header. Rows can be re-sorted by simply clicking on the column header to sort by. The choice of columns appearing the displays can be overridden in the **SamGUI.rsc** file.

The final section displays sam file system bar graphs of storage used and a table of the current mount requests. For each sam file system there will be a tab. Clicking on that tab will bring that display to the front. The mount request table has the same features as the catalog display described above. The default columns in the mount request display are: Slot, Media, Request Count, VSN, and Wait Time.

The **libmgr** display can be resized. As the window resizes horizontally, the middle section will expand or shrink accordingly. As the window resizes vertically, the robot, catalog and mount request displays will expand or shrink accordingly.

ICON ATTRIBUTES

Robot and device icons may display the following attributes:

unavailable
the icon will appear gray

off the icon will not appear

down the icon will appear black

operator attention
there will be a red flash behind the icon

Media icons may show the following attributes:

unavailable
the icon will appear gray

barcoded
there will be a small barcode along the bottom of the icon

damaged
the icon will appear cracked (broken)

read only
a small yellow lock will appear onto of the icon

write protected
a small red lock will appear onto of the icon

cleaning media
the icon will appear yellow

recycle there will be a green triangle on top of the media

OPERATOR AUTHORITY

Operational authority is defined within **defaults.conf**(4). Users with root authority will have full access to the functions within **libmgr**. Users who are part of the sam operator group will have access removed for certain functions. By default, the restrictions include no access to: full audit, label, storage element operations (move, mount, unmount), robot state (except to ON), and refresh. These restrictions can be changed via **defaults.conf**(4).

OPTIONS

−**screensize** *mode_number*
By default, the libmgr tool attempts to choose font and icon sizes that are appropriate for the screen resolution. This feature can be overridden by specifying a screen mode: 0, 1, 2. The exact sizings for these modes can be found in the **SamGUI.rsc** file.

−**locale** *locale_id*
A two character locale code may be specified to have libmgr load the translated text for the given locale. Note: The site must have the appropriate translation text in order for this feature to work.

−**geometry** *WxH+X+Y*
The starting size and position of libmgr. W is the width, H the height. The X and Y coordinates can be either positive (+), in which case they'll orient from the upper left corner, or negative (-), in which case they'll orient from the lower right corner. Either grouping (size or position) is optional. The positioning parameters must lead with either a positive or negative sign.

FILES

mcf	The configuration file for Sun SAM-FS and Sun SAM-QFS environments
SamGUI.rsc	The configuration file for libmgr
defaults.conf	Default configuration settings, including operator authorities

SEE ALSO

mcf(4), **SamGUI.rsc**(4), **defaults.conf**(4)

NAME

samload, load − Load media into a device

SYNOPSIS

/opt/SUNWsamfs/sbin/samload [**−w**] *eq*:*slot*[:*partition*] [*deq*]

/opt/SUNWsamfs/sbin/samload [**−w**] *mediatype*.*vsn* [*deq*]

/opt/SUNWsamfs/sbin/load [**−w**] *eq*:*slot*[:*partition*] [*deq*]

/opt/SUNWsamfs/sbin/load [**−w**] *mediatype*.*vsn* [*deq*]

AVAILABILITY

SUNWsamfs

DESCRIPTION

load requests that the volume specified by *eq*:*slot*[:*partition*] or *mediatype*.*vsn* be loaded into device *deq*. The device specified by *deq* must be a removeable media drive, be in the "unavailable" state (see **set_state**(1M)) and be controlled by a media changer. If *deq* already has a volume loaded, it will be unloaded and the volume put away before the new volume is loaded. If *deq* is not specified, then the volume is loaded into an available drive in the media changer *eq*. The drive that the volume is loaded into will be chosen by Sun SAM-FS or Sun SAM-QFS file system.

Note: Loading media used by a Sun SAM-FS or Sun SAM-QFS file system for archiving could result in the loss of the data contained on that media. Sun Microsystems strongly recommends that archive media **NOT** be loaded in this manner.

The load and samload commands are identical; samload is provided as an alternative to avoid conflict with the Tcl command of the same name.

OPTIONS

−w	**load** will wait for the operation to complete before terminating.

FILES

mcf	The configuration file for Sun SAM-FS and Sun SAM-QFS environments

SEE ALSO

unload(1M), **set_state**(1M), **mcf**(4), **sam-robotsd**(1M)

NAME

load_notify.sh − Sends email when a volume needs to be imported or loaded

SYNOPSIS

load_notify.sh *caller pid log_level message_no VSN*

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **load_notify.sh** script is called by the **sam-fsd**(1M) daemon when a requested volume is not in a library, is not marked unavailable, and the **attended** state is set to **yes**. By default, this script sends email to **root** with the following message:

```
Sun SAM-FS or Sun SAM-QFS needs VSN vsnxxx manually loaded or imported.
Check preview display.
```

To enable this feature, copy the script to **/opt/SUNWsamfs/sbin/load_notify.sh** and modify it to take the desired action for your installation.

FILES

/opt/SUNWsamfs/examples/load_notify.sh

SEE ALSO

sam-fsd(1M), **samset**(1M).

NAME

log_rotate.sh − Rotates log files

SYNOPSIS

log_rotate.sh file [minsize]

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **log_rotate.sh** script rotates log files generated by Sun SAM-FS or Sun SAM-QFS environments and other programs.

The process of rotating log files assumes that you want to keep no more than seven generations of a file in your directories at one time. If the size of *file* is *minsize* or greater the files are "rotated". When the files are rotated, the newest file is renamed *file***.1**, the next-newest file is renamed *file***.2**, and so on. The oldest file in the directory is deleted as new ones are added, so the oldest file in the directory at any time is always called *file***.7**. This process provides two benefits: a given file never becomes so large that it is unwieldy to copy or view, and entries are expired after a period of time, preventing file systems from filling up due to the volume of log entries.

You should send a HUP signal to syslogd after rotating the SAM logfile to make syslogd close and reopen the file in its new location. This is not necessary for files created by SAM processes, since they check to see if the file has been changed whenever it is opened.

The following are some of the Sun SAM-FS and Sun SAM-QFS files you should consider rotating:

File Name or Type	**Location**
SAM log file	See **/etc/syslog.conf** for location.
/devlog files	**/var/opt/SUNWsamfs/devlog/**.
Stage log files	See **/etc/opt/SUNWsamfs/stager.cmd** for location.
Releaser log files	See **/etc/opt/SUNWsamfs/releaser.cmd** for location.
Recycler log files	See **/etc/opt/SUNWsamfs/recycler.cmd** for location.
SEF data files	**/var/opt/SUNWsamfs/sef/sefdata**.

Note that the information in the archiver log is valuable and should be preserved, not discarded after a short period of time.

EXAMPLES

Assume that you want to set up a **crontab**(1) entry to run the **log_rotate.sh**(1M) script at a desired interval for each of the log files you wish to rotate. To rotate file **sam-log** every week, the entry would appear as follows:

```
10 3 * * 0  /opt/SUNWsamfs/sbin/log_rotate.sh /var/adm/sam-log
20 3 * * 0  /bin/kill -HUP ‘/bin/cat /etc/syslog.pid‘
```

This **crontab**(1) file rotates the **/var/adm/sam-log** files every Sunday at 0310. The second line sends a HUP signal to the syslog daemon to notify it to close the file (which has been moved) and open a new one. Note that this action is only useful for files written by syslogd.

To rotate file **releaser-log** every week, the entry would appear as follows:

```
40 2 * * 0  /opt/SUNWsamfs/sbin/log_rotate.sh /var/adm/releaser-log
```

This **crontab**(1) file rotates the **/var/adm/releaser-log** files every Sunday at 0240.

FILES

The **log_rotate.sh** script resides in the following location:

/opt/SUNWsamfs/examples/log_rotate.sh

SEE ALSO

crontab(1), **syslogd**(1M).

NAME

mount_samfs − Mounts a Sun QFS, Sun SAM-FS, or Sun SAM-QFS file system

SYNOPSIS

mount −F samfs [*generic_options*] [**−o** *FSType_specific_options*] *special* | *mount_point*

mount −F samfs [*generic_options*] [**−o** *FSType_specific_options*] *special mount_point*

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **mount** command attaches a file system to the file system hierarchy at the specified *mount_point*, which is the path name of a directory. This man page describes how to mount a Sun QFS, Sun SAM-FS, or Sun SAM-QFS file system, and it explains the unique options that can be used when mounting these file systems.

If the first form of the command is used, which specifies either a *special* or a *mount_point* but not both, the **mount** command searches the **/etc/vfstab** file and fills in missing arguments, including the *FSType_specific_options*. The **mount**(1M) command also searches the **/etc/opt/SUNWsamfs/samfs.cmd** file for mount options.

For more information on the **mount**(1M) command, see the **mount**(1M) man page. For more information on the **/etc/opt/SUNWsamfs/samfs.cmd** file, see the **samfs.cmd**(4) man page.

OPTIONS

−F samfs Specifies that the file system being mounted is of type **samfs**. This is a required option if you are mounting a Sun QFS, Sun SAM-FS, or a Sun SAM-QFS file system. These file systems are all type **samfs**.

generic_options
One or more generic Solaris file system options. For a list of possible *generic_options*, see the **mount**(1M) man page.

−o *FSType_specific_options*
A list of mount options specific to file systems of type **samfs**. If specifying multiple options, separate each option with a comma and no intervening spaces. For the list of possible **−o** *FSType_specific_options*, see one or more of the following headings on this man page:

- Miscellaneous Tuning Options
- I/O Options
- Storage and Archive Management Options
- Shared File System Options
- Multireader File System Options
- Sun QFS and Sun SAM-QFS Options

If no *FSType_specific_options* are specified, the the file system is mounted as a read/write file system.

If invalid options are specified, a warning message is generated and the invalid options are disregarded.

special The Family Set Name from the Sun QFS, Sun SAM-FS, or Sun SAM-QFS master configuration file (**mcf**). For more information on this file, see the **mcf**(4) man page.

mount_point
The path name or directory at which the file system is to be mounted. If the *mount_point* has any contents prior to the **mount** operation, these are hidden until the file system is unmounted.

MISCELLANEOUS TUNING OPTIONS

The following options can be used when mounting a Sun QFS, Sun SAM-FS, or Sun SAM-QFS file system. These options can affect system performance.

sync_meta=*n*
Specifies whether or not the metadata is written to the disk every time it changes, as follows:

- If **sync_meta=0**, metadata is held in a buffer before being written to disk. This delayed write delivers higher performance. This is the default for Sun QFS, Sun SAM-FS, and Sun SAM-QFS file systems that are not mounted as multireader file systems or as Sun QFS shared file systems.
- If **sync_meta=1**, metadata is written to disk every time it changes. This slows performance, but it ensures data consistency. This is the default for Sun QFS file systems that are mounted as multireader file systems or as Sun QFS shared file systems. In a Sun QFS shared file system, this is the setting that must be in effect if failover capability is required.

nosuid Mounts the file system with **setuid** execution disallowed. By default, the file system mounts with **setuid** execution allowed.

notrace | **trace**
The **notrace** option disables file system tracing. The **trace** option enables file system tracing. The default is **notrace**.

stripe=*n* Sets the stripe width for the file system to *n* disk allocation units (DAUs). The stripe width means the $n *$ DAU bytes are written to one data device logical equipment number (LUN) before switching to the next LUN. The DAU size is set on the **sammkfs**(1M) command's −**a** option when the file system is initialized. For *n*, specify an integer such that $0 \le n \le 255$. If *n*=0, files are round robined on each slice.

The default *n* on file systems with an **ms** Equipment Type and on file systems with an *ma* Equipment Type with no striped group (**g***x*) components is as follows:

- 128 kilobytes/DAU for DAUs < 128 kilobytes
- 1 for DAUs ≥ 128 kilobytes

By default, *n*=0 on a Sun QFS shared file system. By default, *n*=0 on file systems with an **ma** Equipment Type with any striped group (**g***x*) components.

NOTE: The system sets **stripe=0** if mismatched striped groups exist.

For more information on file system types, see the **mcf**(4) man page.

I/O OPTIONS

The following options are available for Sun QFS, Sun SAM-FS, and SAM-QFS file systems. They allow changing the type of I/O for a file based on I/O size and history. Note that if direct I/O is specified for a file, these options are ignored and all I/O to regular files is direct, if possible. Well-aligned I/O occurs when the file offset falls on a 512-byte boundary and when the length of the I/O transfer is at least 512 bytes.

dio_rd_consec=*n*
Sets the number of consecutive I/O transfers with a buffer size greater than the specified lower limit (which is **dio_rd_form_min** for aligned reads or **dio_rd_ill_min** for misaligned reads) to *n* operations. By default, *n*=0, which means that no default direct reads occur based on I/O sizes. Also, by default, **dio_rd_form_min** and **dio_rd_ill_min** are ignored.

dio_rd_form_min=*n*
Sets the read well-aligned lower limit to *n* 1024-byte blocks. By default, *n*=256, 1024-byte

blocks. If *n*=0, automatic I/O type switching for well-aligned reads is disabled.

dio_rd_ill_min=*n*
Sets the read misaligned lower limit to *n* 1024-byte blocks. By default, *n*=0, which disables automatic I/O type switching for misaligned reads.

dio_wr_consec=*n*
Sets the number of consecutive I/O transfers with a buffer size above the specified lower limit (which is **dio_wr_form_min** for aligned writes or **dio_wr_ill_min** for misaligned writes) to *n* operations. By default, *n*=0, which means that no default direct writes occur based on I/O sizes. Also, by default, **dio_wr_form_min** and **dio_wr_ill_min** are ignored.

dio_wr_form_min=*n*
Sets the write well-aligned lower limit to *n* 1024-byte blocks. By default, *n*=256 1024-byte blocks. Setting *n*=0 disables automatic I/O type switching for well-aligned writes.

dio_wr_ill_min=*n*
Sets the write misaligned lower limit to *n* 1024-byte blocks. By default, *n*=0, which disables automatic I/O type switching for misaligned writes.

forcedirectio
Specifies direct I/O as the default I/O mode. This means that data is transferred directly between the user's buffer and disk. The **forcedirectio** option should be specified only if the file system is used for large block aligned sequential I/O. For more information, see the **directio**(3C), **setfa**(1), **sam_setfa**(3), and **sam_advise**(3) man pages. The default I/O mode is buffered (uses the page cache).

sw_raid
Causes the file system to align the writebehind buffer. This option should be set if the software raid feature of packages such as Solstice DiskSuite is being used on this file system. This option is off by default.

readahead=*n*
Sets the maximum **readahead** value to *n*. The **readahead** option specifies the maximum number of bytes that can be read ahead by the file system. *n* is in units of kilobytes and must be a multiple of 8. For *n*, specify an integer such that $0 \le n \le 16777216$. The default is 1024 (1,048,576 bytes).

writebehind=*n*
Sets the maximum **writebehind** value to *n*. The **writebehind** option specifies the maximum number of bytes that can be written behind by the file system. *n* is in units of kilobytes and must be a multiple of 8. For *n*, specify an integer such that $0 \le n \le 16777216$. The default is 512 (524,288 bytes).

flush_behind=*n*
Sets the maximum **flush_behind** value to *n*. When enabled, modified pages that are being written sequentially are written to disk asynchronously to help the Solaris VM layer keep the pages clean. This option sets the maximum **flush_behind** value to *n*. *n* is in units of kilobytes. For *n*, specify an integer such that $0 \le n \le 8192$. The default is 0, which disables flush behind.

wr_throttle=*n*
Sets the maximum number of outstanding write bytes for one file to *n* kilobytes. If *n* = 0, there is no limit. The default is 16,384 kilobytes.

STORAGE AND ARCHIVE MANAGEMENT OPTIONS

The following options can be used when mounting a Sun SAM-FS or Sun SAM-QFS file system. These options pertain to the storage and archive management facilities of these file systems.

high=*n* Sets the high-water mark for disk cache utilization to *n* percent. When the amount of space used on the disk cache reaches *n* percent, the Sun SAM-FS or Sun SAM-QFS file systems start the releaser process. For more information, see the **sam-releaser**(1M) man page. The default is 80.

low=*n* Sets the low-water mark for disk cache utilization to *n* percent. When the amount of space used on the disk cache reaches *n* percent, the Sun SAM-FS or Sun SAM-QFS file systems start the releaser process, which stops releasing disk space. The default is 70.

partial=*n* Sets the default partial release size for the file system to *n* kilobytes. The partial release size is used to determine how many bytes at the beginning of a file marked for partial release should be retained on disk cache when the file is released. The user can override the default on a file-by-file basis by specifying a size when marking a file for partial release. For more information, see the **release**(1) man page.

For *n*, specify an integer from 8 to whatever has been set for the **maxpartial** option. For more information on **maxpartial**, see the **maxpartial** option in this list. The default is 16.

maxpartial=*n*
Sets the maximum partial release size for the file system to *n* kilobytes. The partial release size cannot be set larger than this **maxpartial** setting. For *n*, specify an integer such that 0 ≤ *n* ≤ 2097152. The default is 16.

partial_stage=*n*
Sets the partial stage size for the file system to *n* kilobytes. For a partial release file, this value specifies the offset in the file past which access results in the entire file being staged to disk. For *n*, specify a integer from 0 to whatever has been set for the **maxpartial** option. The default is equal to whatever has been set for the **partial** option.

quota Enables quotas for the file system. The default is that quotas are disabled.

stage_n_window=*n*
Sets the **stage −n** size for the file system to *n* kilobytes. This option applies to files that are read directly from the archive media. This attribute is set by using the **stage**(1) command's **-n** option. For a file with this attribute, this is the size that is staged in to the application's buffer at any one time. For *n*, specify an integer such that 64 ≤ *n* ≤ 2097152. The default is 256 for all file systems except the QFS shared file system whose default is set to the mount option minallocsz.

stage_retries=*n*
Sets the number of stage retries attempted per archive copy when certain errors are encountered. For *n*, specify a number such that 0 ≤ *n* ≤ 20. Setting *n*=0 prevents a retry from being initiated. The default is 3.

stage_flush_behind=*n*
Sets the maximum stage flush behind value to *n* kilobytes. Stage pages that are being staged are written to disk asynchronously to help the Solaris VM layer keep pages clean. For *n*, specify an integer such that 0 ≤ *n* ≤ 8192. The default is 0, which means that stage flush behind is disabled.

hwm_archive
Invokes the archiver when the amount of data in the file system increases above the high-water mark.

SHARED FILE SYSTEM OPTIONS

The following options are supported only for Sun QFS and Sun SAM-QFS shared file systems. The stripe width is set by default to round robin (using the **stripe=0** mount option) for the Sun QFS and SAM-QFS shared file system. Note that the Sun QFS and Sun SAM-QFS shared file systems support only file systems configured as an **ma** Equipment Type.

For a description of the **ma** file systems, see the **mcf**(4) man page. For a description of the Sun QFS shared file system, see the *Sun QFS, Sun SAM-FS, and Sun SAM-QFS File System Administrator's Guide*.

shared Specifies that the file system being mounted is a Sun QFS shared file system. The **shared** option must be specified in the **/etc/vfstab** file because it is used in the boot initialization sequence.

bg Specifies that if the first mount attempt fails, the system should retry the mount in the background. If *bg* is not specified, the mount continues in the foreground.

retry=*n* Specifies the number of times to retry the mount operation. For *n*, specify an integer such that 0 ≤ *n* ≤ 20000. By default, *n*=10000.

minallocsz=*n*
Sets the minimum block allocation value for the Sun QFS shared file system to *n*. Specify *n* in units of kilobytes and as a multiple of 8 kilobytes. The **minallocsz** option specifies the minimum number of bytes that are allocated ahead of a write for a Sun QFS shared file system. For *n*, specify an integer such that 16 ≤ *n* ≤ 2097152. By default, *n*=8 ∗ allocation_unit (DAU). See **sammkfs**(1M) command's **−a** option.

maxallocsz=*n*
Sets the maximum block allocation value for the Sun QFS shared file system to *n*. Specify *n* in units of kilobytes and as a multiple of 8 kilobytes. The **maxallocsz** option specifies the maximum number of bytes that are allocated ahead of a write for a Sun QFS shared file system. For *n*, specify an integer such that 16 ≤ *n* ≤ 4194304. By default, *n*=128 ∗ allocation_unit (DAU). See **sammkfs**(1M) command's **−a** option.

rdlease=*n* Sets the read lease time for the Sun QFS shared file system to *n* seconds. The **rdlease** option specifies the maximum number of seconds that a file can be read before reacquiring the read lease. For *n*, specify an integer such that 15 ≤ *n* ≤ 600. By default, *n*=30.

wrlease=*n* Sets the write lease time for the Sun QFS shared file system to *n* seconds. Only one host can write to a file at any one time unless the **mh_write** option is set on the metadata server. If the **mh_write** option is set on the metadata server, multiple hosts can write to and read from the same file at the same time. If multiple hosts are writing, the last write is the one that is effective. The **wrlease** option specifies the maximum number of seconds that a file can be written before reacquiring the write lease. For *n*, specify an integer such that 15 ≤ *n* ≤ 600. By default, *n*=30.

aplease=*n* Sets the append lease time for the Sun QFS shared file system to *n* seconds. Only one host can append to a file at any one time. The **aplease** option specifies the maximum number of seconds that one host can append to a file before reacquiring the append lease. For *n*, specify an integer such that 15 ≤ *n* ≤ 600. By default, *n*=30.

mh_write Enables simultaneous reads and writes to the same file from multiple hosts. This option is effective only on the metadata server host. If this option is specified when mounting the file system on a client host, it is ignored. If the client host becomes the metadata server in the future, however, this option becomes effective. For this reason, it is recommended to use this mount option on the metadata host and all potential metadata server hosts. If the **mh_write** option is not specified on the metadata server, only one host can write at any one time.

nstreams=*n*
Sets the maximum number of concurrent Sun QFS shared file system requests that the metadata server can handle. This concurrency is achieved by **nstreams** threads created in the kernel. The **nstreams=***n* specification should be based on the load expected on this metadata server. For *n*, specify an integer such that 4 ≤ *n* ≤ 256. The default *n*=16.

meta_timeo=*n*
Holds cached attributes for the Sun QFS shared file system at least *n* seconds after file modification. The default *n*=15.

Example 1. If **meta_timeo=0**, the file system always checks to see if the inode is stale. That is, it checks to see if the inode has been changed by the metadata server.

Example 2. If **meta_timeo=15**, which is the default, the file system checks the inode 15 seconds after the last check. This means that if you issue an **ls**(1) command, you might not see a new file for 15 seconds after it has been created by another host.

MULTIREADER FILE SYSTEM OPTIONS

The following options support the single-writer, multireader file system. This file system is mounted on one host system as a single-writer file system that updates the file system. In addition, this file system can be mounted on one or more host systems as a multireader file system.

These options can be specified only on Sun QFS file systems. These options cannot be used if you are mounting the file system as a Sun QFS shared file system.

A major difference between the multireader file system and Sun QFS shared file system is that the multireader host reads metadata from the disk, and the client hosts of a Sun QFS shared file system read metadata over the network.

The system administrator must ensure that only one host in a multireader file system has the file system mounted with the **writer** mount option enabled.

writer Sets the file system to type writer. There can be only one host system that has the file system mounted with the **writer** option at any one time. If **writer** is specified, files are flushed to disk at close and directories are always written through to disk.

Prior to the 4.0 release, the **writer** option was specified as the **shared_writer** option. The older syntax is supported for backward compatibility.

reader Sets the file system to type reader. This mounts the file system as read only. There is no limit to the number of host systems that can have the same file system mounted with the **reader** option. By default, each lookup checks the inode and refreshes the inode pages if the inode has been modified by the writer host. If the **invalid** option is set to a value greater than 0, the inode is checked for modification only after it has aged **invalid** seconds after the last check; for more information, see the **invalid** option.

Prior to the 4.0 release, the **reader** option was specified as the **shared_reader** option. The older syntax is supported for backward compatibility.

invalid=*n* When specified in conjunction with the **reader** option, holds cached attributes for the multireader file system at least *n* seconds after file modification. By default, *n*=0.

Example 1. If **invalid=0**, which is the default, the file system always checks to see if the inode is stale. That is, it checks to see if the inode has been changed by the writer host.

Example 2. If **invalid=30**, the file system checks the inode 30 seconds after the last check. This means that if you issue an **ls**(1) command, you might not see a new file for 30 seconds after it has been created on the writer host.

SUN QFS AND SUN SAM-QFS OPTIONS

The following options are supported only for Sun QFS and Sun SAM-QFS file systems on *ma* Equipment Type file systems. For more information on the **ma** file system Equipment Type, see the **mcf**(4) man page.

qwrite Enables simultaneous reads and writes to the same file from different threads. Specify this option only if users of the file system handle multiple simultaneous transactions to the same file. For example, this is useful for database applications. This option improves I/O performance by queuing multiple requests at the drive level.

By default, **qwrite** is not enabled, and the file system disables simultaneous reads and writes to the same file. This is the mode defined by the UNIX vnode interface standard that gives exclusive access to only one writer and forces other writers and readers to wait.

The **qwrite** option is disabled for NFS reads or writes of the file system.

mm_stripe=*n*
Sets the metadata stripe width for the file system to *n* 16-kilobyte disk allocation units (DAUs). By default, **mm_stripe=1**, which writes one DAU of metadata to one LUN before switching to another LUN. If **mm_stripe=0**, the metadata is round robined across all available metadata LUNs.

FILES

/etc/mnttab	Table of mounted file systems.
/etc/vfstab	List of default parameters for each file system.
/etc/opt/SUNWsamfs/samfs.cmd	List of default and global parameters for Sun SAM-FS and Sun SAM-QFS file systems. For more information, see the **samfs.cmd**(4) man page.

SEE ALSO

release(1), **setfa**(1), **ssum**(1).

mount(1M), **mountall**(1M), **sam-releaser**(1M), **sammkfs**(1M).

mount(2).

sam_setfa(3), **sam_advise**(3), **directio**(3C).

mcf(4), **mnttab**(4), **samfs.cmd(4), vfstab**(4).

NOTES

If the directory upon which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

The mount parameters can be provided in the **samfs.cmd** file, in the **/etc/vfstab** file, and on the **mount**(1M) command. Specifications in the **/etc/vfstab** file override the directives in the **samfs.cmd** file, and options to the **mount**(1M) command override specifications in the **/etc/vfstab** file.

NAME

move − Move a cartridge in a library

SYNOPSIS

/opt/SUNWsamfs/sbin/move *eq***:***src_slot dst_slot*
/opt/SUNWsamfs/sbin/move *mediatype***.***vsn dst_slot*

AVAILABILITY

SUNWsamfs

SUN SAM-FS or SUN SAM-QFS environment

SUN SAM-QFS environment

DESCRIPTION

move will send a request to the library specified by *eq* to move the cartridge in *src_slot* to the slot *dst_slot*. For the form *mediatype*.*vsn*, *eq* and *src_slot* are determined from the catalog entry. All other volumes on the cartridge are moved.

The source slot must be in use and occupied (that is, not loaded in a drive) and the destination slot must not be in use.

Some libraries do not support moving cartridges between storage slots. Generally, if the automated library is SCSI attached, the **move**(1M) command is supported. If the automated library is network attached, the **move**(1M) command is not supported.

If *src_slot* and *dst_slot* are the same, and the cartridge is double-sided, the cartridge will be turned over (flipped).

FILES

mcf	The configuration file for Sun SAM-FS and Sun SAM-QFS environments

SEE ALSO

export(1M), **import**(1M), **mcf**(4), **sam-robotsd**(1M)

NAME

odlabel − Label optical media

SYNOPSIS

odlabel −vsn *vv...* −[**new** | **old** *vv...*] [−**info**] *aa...*] [−**w**] [−**V**] [−**erase**] *eq*[*:slot:side*]

AVAILABILITY

SUNWsamfs

DESCRIPTION

odlabel labels the volume in the optical cartridge specified by *eq***[***:slot:side***].** *eq* is the equipment ordinal. If *eq* is a library, *slot* is the slot in the library containing the cartridge. *side* is the side (1 or 2) of a two-sided cartridge.

A **VOL** (volume) and a **PAR** (partition) label are written. These labels conform to ISO standard IEC13346. The data portion follows ISO standard TC97SC23.

−vsn *vv...* specifies the volume serial name of the optical disk being labeled (up to 31 characters).

If the media being labeled was previously labeled, the VSN must be specified by **−old** *vv...*. The "old" VSN is compared with the VSN on the media to assure that the correct media is being relabeled.

If the media is not labeled (i.e., blank), **−new** must be specified to prevent the previous label comparison from being made.

OPTIONS

−info *aa...* Specifies the "Implementation Use" string in the label (up to 127 characters).

−V Verbose, lists label information written.

−erase Erases the media completely before a label is written. This is a security feature that is normally not necessary. Complete media erasure will take a long time to perform since all data in the media is erased.

−w Wait for the labeling operation to complete. If an error occurs, it will be reported along with a completion code of 1. All labeling errors are also logged. Note: Canceling a command that is waiting for completion will not cause the operation itself to be canceled.

NAME

previewtool − Graphical user interface for displaying pending mount requests associated with Sun SAM-FS and Sun SAM-QFS file systems.

SYNOPSIS

/opt/SUNWsamfs/sbin/previewtool

AVAILABILITY

SUNWsamfs

DESCRIPTION

previewtool presents a graphical user interface for viewing pending mount requests for VSNs associated with Sun SAM-FS or Sun SAM-QFS file systems. Requests for both robot-mounted and operator-mounted VSNs are displayed.

previewtool allows you to select the type of information you want to see. This is controlled by the *Format* menu, located in the upper left corner, and by the *Media* menu, located in the upper middle. Each menu appears when you click the mouse *menu* button on the abbreviated menu button for that menu.

The *Format* menu allows you to choose to see all VSNs waiting for mount, only those waiting for a manual mount, only those waiting for a robot to mount them, or only those associated with a specific robot. When *specific robot* is selected from the *Format* menu, the *Robot* abbreviated menu button appears below the *Format* abbreviated menu button, allowing you to select a specific robot from the list of configured robots; use the mouse *menu* button to do this, just as you selected a format. The *Media* menu allows you to view pending mounts for a specific media type. Selecting *all* in both the *Format* and *Media* menus displays all pending mounts.

previewtool provides a button near the top of its window for clearing individual mount requests. To clear a request, select that request by clicking the mouse *select* button once on that request, and then clicking on the *Clear Request* button.

The display can be immediately updated by clicking on the *Update* button, located in the upper right of the window. Ths display will automatically update when a refresh interval greater than zero is set (in the upper right corner) and refresh is turned on. Refresh is turned on when a checkmark appears in the box immediately to the left of *refresh*; clicking on this box toggles automatic refresh on and off. By default, refresh is turned on and the interval is set to five seconds. To stop automatic refresh, toggle the refresh checkbox or set the interval to zero.

The fields displayed are as follows:

type	Media type. For an explanation of the media type abbreviations, see **mcf** (4).
pid	Process id of requestor.
user	Login name of real user id of requestor.
rb	Equipment ordinal of robot.
flags	The flags displayed have the following meaning:
W	Write access requested.
b	Entry is busy.
C	Clear VSN requested.
f	File system requested.
B	Use block I/O for data transfers.
S	Flipside is already mounted.
s	Stage request flag.
wait	Elapsed time, given in hours:minutes, or in days.
count	If a stage request, the number of requests for this media.

vsn VSN to be mounted.

Operational authority is defined within **defaults.conf**(4). Users with root authority will have full access to the functions within **previewtool**. Users who are part of the sam operator group will have access removed for certain functions. By default, the restrictions include no access to: clear request, media and format menus, and refresh. These restrictions can be changed via **defaults.conf**(4).

SCREEN RESOURCES

The font for preview mount request list can be changed via a resource setting. The following resource can be defined:

fontfamily
Defines the font family to be used for panel lists. An example, define the following line in the .Xdefaults resource file:

```
previewtool.fontfamily: fixed
```

The number of lines displayed in the preview window can be changed using a resource setting. The definition:

displayrows
Defines the number of rows to display in the previewtool window. An example, define the following line in the .Xdefaults resource file:

```
previewtool.displayrows: 2
```

FILES

mcf	The configuration file for Sun SAM-FS and Sun SAM-QFS environments

SEE ALSO

devicetool(1M), **robottool**(1M), **samtool**(1M), **mcf**(4), **defaults.conf**(4)

NAME

qfsdump, qfsrestore − Dump or restore file system data

SYNOPSIS

qfsdump [**-dHqTv**] [**-B** *size* **]** **[-b** *bl_factor* **]** **[-X** *excluded-dir* **]** **-f** *dump_file* **[** *file...* **]**

qfsrestore [**-dilRstTv2**] [**-B** *size* **]** **[-b** *bl_factor* **]** **-f** *dump_file* **[***file...* **]**

DESCRIPTION

qfsdump creates a dump file of the control structures and data of each specified *file* and, if the *file* is a directory, (recursively) its subdirectories. Any *file* specified with an absolute path will be stored in the dump file with an absolute path and any *file* specified with a relative path will be stored in the dump file with a relative path. If no *file* is specified, **qfsdump** creates a dump file of the control structures and data of the current relative directory (referenced as ".") and (recursively) its subdirectories (referenced as "./<subdirectory_name>").

qfsrestore uses the contents of the dump file to restore the control structures and data for all the files in the dump file or each specified *file*. If a *file* is specified, its path and filename must match exactly what exists in the dump file. All files will be restored to the absolute or relative location as each file is described in the dump file, unless the **-s** option is specified. With the **-s** option specified, all filenames with an absolute path in the dump file are restored relative to the current directory, using the entire path as contained in the dump file.

In both **qfsdump** and **qfsrestore**, the dump file must be specified in **-f** *dump_file*, where *dump_file* specifies the name of the dump file to write or read, respectively. If a - (dash) is specified for the *dump_file*, **qfsdump** will write the dump file to **stdout** or **qfsrestore** will read the dump file from **stdin**. The dump file data can be passed through appropriate filters, such as compression or encryption, after being written by **qfsdump** or before being read by **qfsrestore**.

qfsdump and **qfsrestore** require the super-user for execution. Sun Microsystems recommends that a site create **qfsdump** dumps on a periodic basis as part of a disaster recovery plan.

OPTIONS

-d Enable debugging messages. Useful only to Sun Microsystems to trace execution for verification purposes.

-H Specifies the dump file is to be created without a dump header record, or the existing dump file has no header record. This option be used to create control structure dump files which can be concatenated using **cat** (see **cat**(1)).

-i Prints inode numbers of the files when listing the contents of the dump. See also the **-l**, **-t**, and **-2** options.

-l Prints one line per file similar to **sls −l** when listing the contents of the dump. (This option is the lower case letter ‘ell’.) See also the **-i**, **-t**, and **-2** options.

-q Suppresses printing of warning messages during the dump for those files which will be damaged should the dump be restored. By default, such warning messages are displayed.

-R Replaces existing files when restoring control structures.

-s Causes leading slashes to be stripped from filenames prior to restoring them. This is useful if the dump was made with an absolute pathname, and it’s now necessary to restore the dump to a different location. Any directories required for the restoration and not defined in the dump file are automatically created.

-t Instead of restoring the dump, **qfsrestore** will list the contents of the dump file. See also the **-i**, **-l**, and **-2** options.

-T Displays statistics at termination, including number of files and directories processed, number of errors and warnings, etc. An example is:

```
 CSD statistics:
            Files:             52020
            Directories:        36031
            Symbolic links:     0
            Resource files:     8
            File  archives:     0
            Damaged files:      0
            File  warnings:     0
            Errors:             0
            Unprocessed dirs:   0
```

The numbers after "Files", "Directories", "Symbolic links", and "Resource files" are the counts of files, directories and symbolic links whose inodes are contained in the dump.

"File archives" refers to the number of archive images associated with the above Files, Directories, Symbolic links and Resource files. "Damaged files" refers to the number of Files, Directories, Symbolic links, and Resource files which are either already marked damaged (for a qfsdump), or were damaged during a restore because of having no archive image (for a qfsrestore). "File warnings" refers to the number of Files, Directories, Symbolic links and Resource files which would be damaged should the dump be restored (because they had no archive images at the time of the dump). "Errors" refers to the number of error messages which were printed during the dump or restore. These errors are indications of a problem, but the problem is not severe enough to cause an early exit from qfsdump or qfsrestore. Examples of errors during restore are failing to create a symbolic link, failing to change the owner or group of a file. Errors which might occur during a dump include pathname too long, failing to open a directory for reading, failing to read a symbolic link or resource file, or finding a file with an invalid mode. "Unprocessed dirs" refers to the number of directories which were not processed due to an error (such as being unable to create the directory).

-v Prints file names as each file is processed. This option is superseded by options **-l** or **-2**.

-X *excluded-dir*
specifies directory paths to be excluded from the dump. Multiple (up to 10) directories may be excluded by using multiple **-X** parameters. A directory which resolves to . or NULL causes an error message to be issued.

-2 Prints two lines per file similar to **sls −2** when listing the contents of the dump. See also the **-i**, **-l**, and **-t** options.

-B *size* Specifies a buffer size in units of 512 bytes. Note that there are limits on the buffer size, as specified in the error message when the limits have been exceeded. The default buffer size is 512 ∗ 512 bytes.

-b *bl_factor*
Specifies a blocking factor in units of 512 bytes. When specified, all I/O to the dump image file is done in multiples of the blocking factor. There is no blocking done by default.

file... Gives a list of files to be dumped or restored. Note that the names given to restore must match exactly the names as they are stored in the dump; you can use **qfsrestore -t** to see how the names are stored.

NOTES

qfsdump only supports full dumps of specified files and directories. Incremental dump support should be added at a future date.

qfsdump dumps all data of a sparse file, and **qfsrestore** will restore all data. This can lead to files occupying more space on dump files and on restored file systems than anticipated. Support for sparse files should be added at a future date.

ERRORS

"Not a SAM-FS file" means that you are attempting to operate on a file which is not contained in a Sun SAM-FS or Sun SAM-QFS file system.

"*file*: Unrecognised mode (0x..)" means that **qfsdump** is being asked to dump a file which is not a regular file, directory, symbolic link or request file. While Sun SAM-FS and Sun SAM-QFS allow the creation of block special, character special, fifo ... files, these do not function correctly, and **qfsdump** does not attempt to dump them.

"*file*: Warning! File will be damaged." during a **qfsdump** means that the file in question does not currently have any archive copies. The file is dumped to the **qfsdump** file, but if the **qfsdump** file is used to restore this file, the file will be marked damaged.

"*file*: Warning! File is already damaged." during a **qfsdump** means that the file is currently marked damaged. During restore, the file will still be damaged.

"*file*: File was already damaged prior to dump" during a **qfsrestore** means that the file was dumped with the "damaged" flag set.

".: Not a SAM-FS file." means that you are attempting to dump files from a non-SAM-FS file system or restore files from a **qfsdump** dump file into a non-SAM-FS file system.

"*file*: stat() id mismatch: expected: %d.%d, got %d.%d" during a dump indicates one of two things. If the %d. portions match, but the .%d portions differ, then a directory or file was deleted and recreated while **qfsdump** was operating on it. The file is not dumped. If the %d. portions do not match, then a serious error has been encountered; consult your service provider for help.

"Corrupt samfsdump file. name length %d" during a restore means that the pathname of a file to be restored was less than zero, or larger than MAXPATHLEN. This should not occur. **qfsrestore** aborts.

"Corrupt samfsdump file. %s inode version incorrect" during a restore means that a the inode for the indicated file was in an old format. This should not occur. **qfsrestore** aborts.

"*file*: pathname too long" during a dump indicates that the pathname of the indicated file is longer than 1024 characters. The file is not dumped.

EXAMPLES

The following example creates a control structure dump of the entire **/sam** file system:

```
example# cd /qfs1
example# qfsdump -f /destination/of/the/dump/qfsdump.today
```

To restore a file system dump to **/qfs1**:

```
example# cd /qfs1
example# qfsrestore -f /source/of/the/dump/qfsdump.yesterday
```

SEE ALSO

sls(1), **cat**(1)

NAME

rearch − Marks archive entries to be rearchived

SYNOPSIS

rearch [−**f**] [−**M**] [−**o**] −**m** *media* −**v** *vsn filename* . . .

rearch [−**f**] [−**M**] [−**o**] −**c** *n filename* . . .

rearch [−**f**] [−**M**] [−**o**] −**m** *media* −**v** *vsn* −**r** *dirname* [*filename* . . .]

rearch [−**f**] [−**M**] [−**o**] −**c** *n* −**r** *dirname* [*filename* . . .]

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **rearch** command marks archive entries for one or more files or directories to be rearchived. You must specify either a copy number or both a media type and a VSN number. In addition, you must specify either a file name or both a directory name and a file name.

OPTIONS

This command accepts the following options:

−**c** *n* Specifies the archive copy number. If one or more −**c** *n* options are specified, only those archive copies (1 to 4) are marked. The default is all copies.

−**f** Suppresses errors.

−**M** Rearchives metadata only. This includes directories, the segment index, and removable media files. Regular files and symbolic links are not rearchived.

−**m** *media* Specifies the media type. If specified, archive copies on the specified media are marked. This option must be specified in conjunction with the −**v** *vsn* option. For more information on media types, see the **mcf**(4) man page.

−**o** Requires the file to be online before its archive entry is deleted. If the file is offline, the command stages the file onto disk before deleting any entries.

−**v** *vsn* Marks archive copies on VSN *vsn* for rearchiving. This option must be specified in conjunction with the −**m** *media* option.

−**r** *dirname*
Recursively rearchives the archive entries of the specified *dirname* and its subdirectories. The **rearch** flag for archive entries of files in the directories and subdirectories is set. If no −**r** *dirname* option is specified, at least one *filename* must be specified.

filename . . .
Specifies one or more files for rearchiving. If you are using the first form of the command, either a *filename* or an asterisk (∗) is required. If you are using the third or fourth forms of the command, and you do not specify a *filename*, you must use the −**r** option and specify a *dirname*.

SEE ALSO

mcf(4).

NAME

recover.sh − Recovers files archived after last **samfsdump**(1M) was taken

SYNOPSIS

recover.sh *mount_point*

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **recover.sh** script recovers files using the information in the archiver log. This script can be useful in a disaster recovery situation when a file system has been lost and is recovered from a saved **samfsdump**(1M) file. If files were archived for the first time after the dump was taken, there is no record of them in the dump. This script can be used to reload those files from the archive copy by using the **star**(1M) program.

USAGE

Step 1. Edit the archiver log file and extract the relevant portion.

In this editing session, you should eliminate entries for second, third, or fourth archive copies from this file because otherwise the files are recovered multiple times, which wastes time. You should also eliminate directory entries. Directory entries are noted by a **d** in field 12 of the archiver log.

After the file is edited, save the edited file to a temporary file. For example, save this file to **/tmp/arlog.in**.

Step 2. Copy the script from its default location to a temporary location.

Use a command such as the following to copy the script to a temporary location:

```
cp /opt/SUNWsamfs/examples/recover.sh /tmp/recover.sh
```

Step 3. Edit a working copy of the script and modify it for your site.

Edit the copy and change the value of **BLK_SIZE** from **128** to the block size in kilobytes for the VSNs in question.

Step 4. Run the **recover.sh** script.

This creates a new script to actually do the work of recovering the files. In the following example, the Sun SAM-FS mount point is **/sam1**.

```
server# /tmp/recover.sh /sam1 < /tmp/arlog.in > /tmp/recover.out
```

If you have multiple drives and want to recover from more than one VSN at a time, you can split this script into pieces first. The following line appears at the end of the work for each VSN:

"# ----------- end of files for vsn " XXX " ---------"

The **XXX** is replaced with the VSN's bar code label.

Step 5. Create a temporary directory to which the recovered files can be written.

Create this directory in a Sun SAM-FS or Sun SAM-QFS file system. Although this could be your mount point, it's probably better to recover to a temporary directory in the Sun SAM-FS or Sun SAM-QFS file system first, and then move the files to their final location once recovery is complete and everything looks OK. For example:

```
server# mkdir /sam1/recover
```

Step 6. Change to the temporary directory to receive the recovered files.

Use the **cd**(1) command to change to the directory in which you want the files recovered.

```
server# cd /sam1/recover
server# sh -x /tmp/recover.out
```

Step 7. Run the **recover.out** script.

The **/tmp/recover.out** shell script is created in the previous step. It can be used to recover all the files listed in the **/tmp/arlog.in** file.

Run the **recover.out** script. If you have split the scripts, you may have to run it multiple times.

WARNINGS

Improper use of this script may damage user or system data. Please refer to the *Sun QFS, SAM-FS, and SAM-QFS Disaster Recovery Guide* or contact technical support before using this script.

NOTES

If used with the SAM-Remote clients or server, the recovery must be performed on the server to which the tape library is attached.

Do not run multiple recovery scripts at the same time.

FILES

This script resides in the following location:

/opt/SUNWsamfs/examples/recover.sh

SEE ALSO

archiver(1M), **request**(1M), **star**(1M).

NAME

reserve − Reserve a volume for archiving.

SYNOPSIS

/opt/SUNWsamfs/sbin/reserve *mediatype***.***vsn*
asname/*owner*/*fsname* **[***time***]**
/opt/SUNWsamfs/sbin/reserve *eq***:***slot***[:***partition***]**
asname/*owner*/*fsname* **[***time***]**

AVAILABILITY

SUNWsamfs

DESCRIPTION

reserve assigns the volume for archival of specific files. The fields

Normally, the archiver performs reservation of volumes. This command is provided to pre-reserve a volume.

The volume is determined by the specifier *mediatype***.***vsn* , or *eq***:***slot***[:***partition***]**

The reservation is specified by the fields **asname**, **owner**, and **fsname** These fields may be empty depending on the options in the archiver command file.

time is the time the volume is reserved. If not specified, the reserve time is set to the present time. Several formats are allowed for *time*. Examples are:

"2000-09-19"; "2000-07-04 20:31"; 23:05; "Mar 23"; "Mar 23 1994"; "Mar 23 1994 23:05"; "23 Mar"; "23 Mar 1994"; "23 Mar 1994 23:05".

Month names may be abbreviated or spelled out in full. Time-of-day is given in 24-hour format. Years must use all four digits. If the *time* contains blanks, the entire time must be enclosed in quotation marks.

SEE ALSO

archiver(1M), **archiver.cmd**(1M), **unreserve**(1M)

NAME

restore.sh − Restores files online

SYNOPSIS

restore.sh *log_file mount_point*

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **restore.sh** script restores files to their online or partially online status. This script should be used after performing a file system restore using the **samfsrestore**(1M) command.

This script accepts the following arguments:

log_file Specify the name of the log file that was created by the **sammkfs**(1M) or the **samfsrestore**(1M) commands.

mount_point
Specify the mount point of the file system being restored.

USAGE

Step 1. Recreate or restore the file system.
You can do this by using the **samfsrestore**(1M) command with its **-g** option. This creates a log file.

Step 2. Run the **restore.sh** script.
The first argument is the log file created in the previous step, and the second argument is the file system mount point. This script stages back the files that were previously online or partially online at the time the **.inodes** copy or **samfsdump**(1M) was created.

FILES

The **restore.sh** script resides in the following location:

/opt/SUNWsamfs/examples/restore.sh

SEE ALSO

samfsdump(1M), **samfsrestore**(1M).

The *Sun SAM-QFS and Sun SAM-FS Disaster Recovery Guide*.

NAME

robottool − Graphical user interface for displaying information about and managing robot devices associated with Sun SAM-FS or Sun SAM-QFS file system.

SYNOPSIS

/opt/SUNWsamfs/sbin/robottool

AVAILABILITY

SUNWsamfs

DESCRIPTION

robottool presents a graphical user interface for viewing information about and managing the robot devices associated with Sun SAM-FS and Sun SAM-QFS file systems.

Information in **robottool** is displayed in three sections. Robot devices are listed at the top. VSNs associated with a selected robot are displayed in the middle. Devices associated with a selected robot are displayed at the bottom. All information is displayed as a scrolling list.

Initially, the first robot will be selected. When a robot is selected from the list, the VSN catalog and devices associated with that robot are displayed, and the buttons for the commands appropriate to the selected robot and its state become active. Buttons for performing actions on a selected robot are located to the right of the robots display. Deselecting a robot will cause its VSN catalog and devices displays to disappear.

Selecting a VSN from the VSN display allows you to perform actions on that VSN. The buttons for these actions are located to the right of the VSN Catalog display. The possible actions are: *Audit*, *Export*, *Mount*, *Unmount*, *Label*, and *Move*. **robottool** makes available only those actions that make sense for the selected VSN, the users operational authority level, and the device type. To perform an action on a VSN, click the mouse *select* button once on the button corresponding to that action.

The devices for a robot are displayed for information only. To perform actions on a device, use **devicetool**(1M). The device panel list will automatically resize to the number of devices in the list when a robot is selected.

The display can be immediately updated by clicking on the *Update* button, located in the upper right of the window. The display will automatically update when a refresh interval greater than zero is set (in the upper right corner) and refresh is turned on. Refresh is turned on when a checkmark appears in the box immediately to the left of *refresh*; clicking on this box toggles automatic refresh on and off. By default, refresh is turned on and the interval is set to five seconds. To stop automatic refresh, toggle the refresh checkbox or set the interval to zero.

The characters in the robot and device status string, appearing from left to right, have the following meanings:

s	media is being scanned
m	the file system is mounted
M	maintenance mode
E	device received an unrecoverable error in scanning
a	device is in audit mode
l	media has a label
I	device is in idle wait
A	needs operator attention
U	unload has been requested
R	the device is reserved
w	a process is writing on the media

o	the device is open
P	the device is positioning
F	all storage elements occupied
R	device is ready and the media is read-only
r	device is spun up and ready
p	device is mounted

The characters in the VSN catalog status string, appearing from left to right, have the following meanings:

A	volume needs an audit
R	volume is marked for recycling
W	volume is write protected
E	bad media
X	this is an export slot
r	volume is marked read-only
u	element address is unavailable
l	volume is labeled
c	cleaning media
p	element address is occupied

Operational authority is defined within **defaults.conf**(4). Users with root authority will have full access to the functions within **robottool**. Users who are part of the sam operator group will have access removed for certain functions. By default, the restrictions include no access to: full audit, label, element address operations (move, mount, unmount), robot state (except to ON), and refresh. These restrictions can be changed via **defaults.conf**(4).

SCREEN RESOURCES

The font for panel lists, such as the catalog, device and robots lists, can be changed via a resource setting. The following resource can be defined:

fontfamily

Defines the font family to be used for panel lists. An example, define the following line in the .Xdefaults resource file:

```
robottool.fontfamily: fixed
```

FILES

mcf	The configuration file for Sun SAM-FS and Sun SAM-QFS environments

SEE ALSO

devicetool(1M), **previewtool**(1M), **samtool**(1M), **mcf**(4), **defaults.conf**(4)

NAME

sam-archiverd − Sun SAM-FS and Sun SAM-QFS file archive daemon

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-archiverd

AVAILABILITY

SUNWsamfs

DESCRIPTION

The archiver daemon automatically archives Sun SAM-FS or Sun SAM-QFS files when a Sun SAM-FS or Sun SAM-QFS file system is mounted. It is started by sam-fsd, and cannot be executed from a command line. Directives for controlling the archiver are read from the archiver commands file, which is **/etc/opt/SUNWsamfs/archiver.cmd**. This file does not have to be present for the archiver daemon to execute. If the **archiver.cmd** file is present, however, it must be free of errors. Errors in the **archiver.cmd** file prevent the archiver daemon from executing. If the **archiver.cmd** file is not present, all files on the file system are archived to the available removable media according to archiver defaults.

sam-archiverd executes in the directory **/var/opt/SUNWsamfs/archiver**. This is the archiver's working directory. Each **sam-arfind** daemon executes in a subdirectory named for the file system being archived. Each **sam-arcopy** daemon executes in a subdirectory named for the archive file (rm0 - rmxx) being archived to.

ARCHIVING INTERNALS

Archive Sets are the mechanism that the archiver uses to direct files in a **samfs** file system to media during archiving.

All files in the file system are members of one and only one Archive Set. Characteristics of a file are used to determine Archive Set membership. All files in an Archive Set are copied to the media associated with the Archive Set. The Archive Set name is simply a synonym for a collection of media volumes.

Files are written to the media in an Archive File which is written in **tar** format. The combination of the Archive Set and the **tar** format results in an operation that is just like using the command **find(1)** to select files for the **tar** command.

In addition, the file system meta data, (directories, the index of segmented files, and the removable media information), are assigned to an Archive Set to be copied to media. The Archive Set name is the name of the file system. (See **mcf**(4)).**Symbolic**links**are**considered purposes of archiving.

Each Archive Set may have up to four archive copies defined. The copies provide duplication of files on different media. Copies are selected by the Archive Age of a file.

Files in an Archive Set are candidates for archival action after a period of time, the Archive Age, has elapsed. The Archive Age of a file is computed using a selectable time reference for each file. The default time reference is the file's modification time.

For processing files in archive sets with an unarchive age specified, the unarchive age default time reference is the file's access time. But, in this case, two other conditions are recognized: If the modification time is later than the access time, the modification time is used. And, if an archive copy was unarchived, the file will be rearchived only after the file is staged from another copy, i.e the file was offline at the time a read access was made to the file.

Since users may change these time references to values far in the past or future, the time reference will be adjusted by the archiver to keep it in the range: creation_time <= time_ref <= time_now.

Scheduling archive copies.

Finding files to archive.

Each file system is scanned by an individual sam-arfind. The scan is accomplished by one of two algorithms:

1. Recursively descend through the directory tree. "stat()" each file to get an inode to examine. This algorithm is used the first time that sam-arfind executes. This assures that each file gets examined and the file status "archdone" is set if the file does not need archiving.

2. Read the .inodes file. If an inode does not have "archdone" set, determine the file name and examine the inode. This algorithm is used for all other scans.

Determine which Archive Set the file belongs in using the file properties descriptions. If the Archive Age of the file has been met or exceeded, add the file to the archive request (ArchReq) for the Archive Set. The ArchReq contains a 'batch' of files that can be archived together. For segmented files, the segment, not the entire file, is the archivable unit, so the properties (e.g. minimum file size) and priorities apply to the segment. The ArchReq-s are files in separate directories for each filesystem. I.e: **/var/opt/SUNWsamfs/archiver/***file_system***/ArchReq** and you can display them by using the showqueue(1M) command. An ArchReq is removed once the files it specifies have been archived.

The characteristics used for determining which Archive Set a file belongs in are:

directory path portion of the file's name

complete file name using a regular expression

user name of the file's owner

group name of the file's owner

minimum file size

maximum file size

If a file is offline, select the volume to be used as the source for the archive copy. If the file copy is being rearchived, select that volume.

Each file is given a file archive priority. The archive priority is computed from properties of the file and property multipliers associated with the Archive Set. The computation is effectively:

```
ArchivePriority = sum(Pn * Mn)

where:  Pn = value of a file property
        Mn = property multiplier
```

Most property values are 1 or 0 as the property is TRUE or FALSE. For instance, the value of the property 'Copy 1' is 1 if archive copy 1 is being made. The values of 'Copy 2', 'Copy 3' and 'Copy 4' are therefore 0.

Others, such as 'Archive Age' and 'File size' may have values other than 0 or 1.

The archive priority and the Property multipliers are floating point numbers. The default value for all property multipliers is 0.

The file properties used in the priority calculation are:

Archive Age	seconds since the file's Archive Age time reference (time_now - time_ref)
Copy 1	archive copy 1 is being made
Copy 2	archive copy 2 is being made

Copy 3	archive copy 3 is being made
Copy 4	archive copy 4 is being made
Copies made	number of archive copies previously made
File size	size of the file in bytes
Archive immediate	immediate archival requested for file
Rearchive	archive copy is being rearchived
Required for release	archive copy is required before file may be released

All the priorities that apply for a file are added together. The priority of the ArchReq is set to the highest file priority in the ArchReq.

When the filesystem scan is finished, send each ArchReq to sam-archiverd.

Composing archive requests.

If the ArchReq requires automatic ’owner’ Archive Sets, separate the ArchReq by owner.

For ArchReq-s with a ’join’ method required:
Sort the files using the join method property as the key. This collects the files with the same property together. Step through the ArchReq to mark the archive file boundaries where the properties differ. Sort the files within the archive file boundaries according to the ’sort’ method. Each group of joined files is treated as if it were a single file for the remainder of the composing and scheduling processes.

Sort the files according to the ’sort’ method. Sorting the files will tend to keep the files together in the archive files. The default is no sorting so the files will be archived in the order encountered during the file system scan.

Separate the ArchReq into online and offline files. All the online files will be archived together, and the offline files will be together.

The priority of each ArchReq created during this process is set to the highest file priority in the ArchReq. Enter the ArchReq into the scheduling queue in priority order.

Scheduling from the queue.

When an ArchReq is ready to be scheduled to an sam-arcopy, the volumes are assigned to the candidate ArchReq-s as follows:

The volume that has most recently been used for the Archive Set is used if there is enough space for the ArchReq.

If an ArchReq is too big for one volume, files that will fit on the volume are selected for archival to that volume. The remaining files will be archived later.

An ArchReq with a single file that is too large to fit on one volume, and is larger than ’ovflmin’ will have additional volumes assigned as required. The additional volumes are selected in order of decreasing size. This is to minimize the number of volumes required for the file.

For each candidate ArchReq, compute the a scheduling priority by adding the archive priority to the following properties and the associated multipliers:

Archive volume loaded	the first volume to be archived to is loaded in a drive
Files offline	the request contains offline files
Multiple archive volumes	the file being archived requires more than one volume
Multiple stage volumes	the file being archived is offline on more than one volume

Queue wait	seconds that the ArchReq has been queued
Stage volume loaded	the first volume that contains offline files is loaded in a drive

Enter each ArchReq into the archive queue in priority order. Schedule only as many sam-arcopy-s as drives allowed in a robot or allowed by the Archive Set. When all sam-arcopy-s are busy, wait for an sam-arcopy to complete. Repeat the scheduling sequence until all ArchReq-s are processed.

If the Archive Set specifies multiple drives, divide the request for multiple drives.

Assigning an ArchReq to an sam-arcopy.

Step through each ArchReq-s to mark the archive file boundaries so that each archive file will be less than archmax in size. If a file is larger than archmax, it will be the only file in an archive file.

Using priorities to control order of archiving.

By default, all archiving priorities are set to zero. You may change the priorities by specifying property multipliers. This allows you to control the order in which files are archived. Here are some examples (see **archiver.cmd**(4)):

You may cause the files within an archive file to be archived in priority order by using **-sort priority**.

You may reduce the media loads and unloads with: **-priority archive_loaded 1** and **-priority stage_loaded 1**.

You may cause online files to be archived before offline files with: **-priority offline -500**.

You may cause the archive copies to be made in order by using: **-priority copy1 4000**, **-priority copy2 3000**, **-priority copy3 2000**, **-priority copy4 1000**.

OUTPUT FORMAT

The archiver can produce a log file containing information about files archived and unarchived. Here is an example:

```
A 2000/06/02 15:23:41 mo OPT001 samfs1.1 143.1 samfs1 6.6 16384 lost+found d 0 51
A 2000/06/02 15:23:41 mo OPT001 samfs1.1 143.22 samfs1 19.3 4096 seg d 0 51
A 2000/06/02 15:23:41 mo OPT001 samfs1.1 143.2b samfs1 22.3 922337 rmfile R 0 51
A 2000/06/02 15:23:41 mo OPT001 samfs1.1 143.34 samfs1 27.3 11 system l 0 51
A 2000/06/02 15:23:41 mo OPT001 samfs1.1 143.35 samfs1 18.5 24 seg/aa I 0 51
A 2000/06/02 15:23:43 ib E00000 all.1 110a.1 samfs1 20.5 14971 myfile f 0 23
A 2000/06/02 15:23:44 ib E00000 all.1 110a.20 samfs1 26.3 10485760 seg/aa/1 S 0 23
A 2000/06/02 15:23:45 ib E00000 all.1 110a.5021 samfs1 25.3 10485760 seg/aa/2 S 0 23
A 2000/06/02 15:23:45 ib E00000 all.1 110a.a022 samfs1 24.3 184 seg/aa/3 S 0 23
```

Field	Description
1	A for archived. R for re-archived; U for unarchived.
2	Date of archive action.
3	Time of archive action.

4	Archive media.
5	VSN
6	Archive set and copy number.
7	Physical position of start of archive file on media and file offset on the archive file / 512.
8	File system name.
9	Inode number and generation number. The generation number is an additional number used in addition to the inode number for uniqueness since inode numbers get re-used.
10	Length of file if written on only 1 volume. Length of section if file is written on multiple volumes.
11	Name of file.
12	Type of the file. File is of type *c*:
	d directory
	f regular file
	l symbolic link
	R removable media file
	I segment index
	S data segment
13	Section of an overflowed file/segment.
14	Equipment ordinal from the mcf of the device on which the archive copy was made.

SEE ALSO

archiver(1M), **archiver.cmd**(4), **sam-arcopy**(1M), **sam-arfind**(1M)

NAME

sam-arcopy − Sun SAM-FS and Sun SAM-QFS archive copy daemon

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-arcopy

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **sam-arcopy** process is responsible for copying Sun **SAM-FS** or Sun **SAM-QFS** files to removable media. It is executed by **sam-archiverd**(1M). All required information is transmitted to the **sam-arcopy** process through a pipe.

SEE ALSO

sam-archiverd(1M)

NAME

sam-arfind − Sun SAM-FS and Sun SAM-QFS archive find daemon

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-arfind *file_system*

AVAILABILITY

SUNWsamfs

DESCRIPTION

sam-arfind is responsible for finding **samfs** file system files to be archived. It is executed by **sam-archiverd**(1M). The only argument is the name of the file system. All other required information is transmitted to **sam-arfind** through a pipe.

SEE ALSO

sam-archiverd(1M)

NAME

sam-fsd − Initialize Sun SAM-FS and Sun SAM-QFS environments.

SYNOPSIS

/usr/lib/fs/samfs/sam-fsd [**−C**] [**−c** *defaults*] [**−d** *diskvols*] [**−f** *samfs*] [**−m** *mcf*] [**−v**]

AVAILABILITY

SUNWsamfs

DESCRIPTION

sam-fsd initializes Sun SAM-FS and Sun SAM-QFS environments and performs tasks for the filesystem kernel code. These tasks include sending messages to syslog, and starting the archiver, releaser, shared fs, and stager daemons. It is started by **init**(1M) using an entry in **/etc/inittab**

```
sf:23:respawn:/usr/lib/fs/samfs/sam-fsd
```

When started, **sam-fsd** reads the configuration files **defaults.conf**, **diskvols.conf**, **mcf**, and **samfs.cmd** located in the directory **/etc/opt/SUNWsamfs**. These files may be changed at any time while **sam-fsd** is running. The changes will take place when **sam-fsd** is restarted, or sent the signal SIGHUP.

The filesystems are configured and necessary daemons are started. Configuration parameters are set, and table files are written for use by other components of Sun SAM-FS and Sun SAM-QFS environment.

If errors occur in any of the configuration files, **sam-fsd** refuses to run and writes a notification message to **syslog**. The problem must be corrected, and the signal SIGHUP sent to **sam-fsd**. **sam-fsd** then rereads the configuration files. The **syslog** message contains the command necessary to signal **sam-fsd .**
’kill -HUP *sam-fsd-pid*’

Trace files.

Several Sun QFS, SAM-FS, and SAM-QFS daemons write messages to trace files. These messages contain information about the state and progress of the work performed by the daemons. The messages are primarily used by Sun engineers and support personnel to improve performance and diagnose problems. As such, the message content and format are subject to change with bugfixes and feature releases.

The daemons writing trace files are: sam-archiverd, sam-catserver, sam-fsd, sam-ftpd, sam-recycler, sam-sharefsd, and sam-stagerd.

To prevent the trace files from growing indefinitely, sam-fsd monitors the size and age of the trace files and periodically executes the script ’/opt/SUNWsamfs/sbin/trace_rotate.sh’. This script moves the trace files to sequentially numbered copies. The script is executed when the trace file exceeds a specified size, or age. The size and age are specified in ’defaults.conf’. If ’/opt/SUNWsamfs/sbin/trace_rotate.sh’ does not exist, sam-fsd performs no action.

OPTIONS

sam-fsd may be started by direct execution to provide detailed messages about problems in configuration files. In this case, the following options are allowed:

−c *defaults*
Sets an alternate defaults.conf file to check. *defaults* is the path to the alternate defaults configuration file.

−d *diskvols*
Sets an alternate diskvols.conf file to check. *diskvols* is the path to the alternate diskvols configuration file.

−f *fs_name*
Sets a single file system. *fs_name* is the family set name from the mcf file.

−m *mcf*
Sets an alternate mcf file to check. *mcf* is the path to the alternate mcf file.

−v Sets verbose mode.

−C Configure Sun SAM-FS and Sun SAM-QFS if not already configured. Must be the only option.

FILES

/etc/opt/SUNWsamfs	Location of Sun SAM-FS and Sun SAM-QFS configuration files
mcf	The configuration file for Sun SAM-FS and Sun SAM-QFS environments.
samfs.cmd	SAM-FS mount commands file
defaults.conf	Set default values for Sun SAM-FS and Sun SAM-QFS environment.

SEE ALSO

defaults.conf(4), **diskvols.conf**(4), **mcf**(4), **samfs.cmd**(4), and **trace_rotate.sh**(4)

NAME

sam-ftpd − Sun SAM-FS and SAM-QFS file transfer server process

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-ftpd

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **sam-ftpd** process is the file transfer server process for transferring Sun SAM-FS or Sun SAM-QFS files to and from a remote network site. The **sam-ftpd** process is initiated by the **sam-fsd** daemon.

By default, the ftp daemon uses the default behaviors described on the **ftp.cmd**(4) man page.

FILES

If the daemon's command file is present in **/etc/opt/SUNWsamfs/ftp.cmd**, the **sam-ftpd** process reads that file.

SEE ALSO

sam-fsd(1M).

ftp.cmd(4).

NAME

sam-initd − Initialize the Sun SAM-FS and Sun SAM-QFS system daemons

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-initd

AVAILABILITY

SUNWsamfs

DESCRIPTION

sam-initd initializes the Sun SAM-FS and Sun SAM-QFS system daemons. It is typically started by **mount_samfs**(1M) but can be started without mounting the file system. Only the super-user can start **sam-initd**.

FILES

/opt/SUNWsamfs/sbin	Location of Sun SAM-FS or Sun SAM-QFS daemons
/etc/opt/SUNWsamfs	Location of Sun SAM-FS or Sun SAM-QFS daemons configuration files
/etc/opt/SUNWsamfs/mcf	The configuration file for Sun SAM-FS and Sun SAM-QFS environments.

SEE ALSO

mcf(4), **mount**(1M), **mount_samfs**(1M), **archiver**(1M), **generic**(1M), **scanner**(1M), **robots**(1M)

NOTES

sam-initd should only be started by direct execution under unusual circumstances, such as initial installation, when the site wishes to label a robot full of media. To shutdown **sam-initd**, use **ps**(1) to determine the process ID (*sam-init-pid*) of **sam-initd** then enter the following:

```
kill -INT sam-init-pid
```

To shutdown **sam-initd** without releasing shared memory, use:

```
kill -HUP sam-init-pid
```

A handy root alias (csh) is:

```
alias killinit  'kill -INT `/bin/ps -e | grep sam-initd | cut -c1-6`'
```

NAME

sam-logd − Describes the obsolete Sun SAM-FS logging daemon

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-logd

DESCRIPTION

In releases prior to 4.0, the **sam-logd** daemon provided a mechanism for logging staging events from Sun SAM-FS. Logging was controlled by the **/etc/opt/SUNWsamfs/samlogd.cmd** commands file.

As of the 4.0 releases of Sun SAM-FS and Sun SAM-QFS, this functionality is provided through the **stager.cmd** file. For more information, see the **stager.cmd**(4) man page.

This man page will be removed in a future release.

NAME

sam-recycler − Recycles Sun SAM-FS and Sun SAM-QFS volumes

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-recycler [**−c**] [**−C**] [**−d**] [**−E**] [**−n**] [**−s**] [**−v**] [**−V**] [**−x**] [**−X**]
[*family_set* | *archive_set*]

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **sam-recycler** command invokes the recycler. The recycler removes expired archive copies and frees up archive volumes. Often, the recycler is invoked through root's **crontab**(1) file at an off-peak time. However, the recycler can be invoked at any time.

You can specify that only a specific library or archive set be recycled. You can recycle by library only when archiving to tape or magneto optical cartridges in a library. Note that you cannot recycle by library if you are using disk archiving.

If you want to recycle by archive set, you must name the archive sets to be recycled in the **/etc/opt/SUNWsamfs/archiver.cmd** file.

You can provide directives to the recycler through lines entered in the **/etc/opt/SUNWsamfs/recycler.cmd** file and in the **/etc/opt/SUNWsamfs/archiver.cmd** file. If no directives are present and no *family_set* or *archive_set* is specified on the command line, recycling does not occur. The folowing are the default recycler settings:

- The maximum data quantity to recycle (**-dataquantity**) is 1 gigabyte (1G).
- The high water mark (**-hwm**) is 95.
- The VSN gain (**-mingain**) is 50.
- The number of volumes (**-vsncount**) to recycle is 1.
- Automatic email is not sent.

OPTIONS

The following options determine the volumes to be recycled and the content of the recycler log file.

−c Displays the extrapolated capacity of each volume. This is the volume's capacity assuming the compression observed on the volume so far continues for the rest of the volume. This option produces an additional line for each volume with the heading **Alpha:**.

−C Suppresses listing of initial catalog(s).

−d Displays messages during the volume selection phase of processing. These messages indicate why each volume was, or was not, selected for recycling.

−E Specifies that the volume section of the recycler's log file list only volumes that are not 100% free.

−n Prevents any actions from being taken. This option causes **/opt/SUNWsamfs/sbin/sam-recycler** to behave as if **-recycle_ignore** were specified in the **/etc/opt/SUNWsamfs/archiver.cmd** file for all archive sets.

−s Suppresses the listing of individual volumes in the initial catalog section.

−v Displays information about which files are resident on the volume that is marked for recycling. If no path name can be calculated for the inode, it lists the inode. These files are on volumes that are being drained.

−V Suppresses the volume section in the listing.

−x Displays messages for expired archive copies. These are copies that are older than the time the volume upon which the copies reside was labeled. Such copies generate an error message when staged. The data for those copies is irrecoverable. These archive copies must be unarchived. If any such copies are discovered, the recycler stops. This is the default behavior. Also see the **−X** option.

−X Inhibits the messages that indicate the existance of expired archive copies. Typically, if the recycler detects expired archive copies, it stops. Use this options if you want the recycler to continue in the presence of expired archive copies. Also see the **−x** option.

family_set | *archive_set*
Recycles only the named *family_set* or *archive_set*. This is an optional argument. If a *family_set* is specified, the library associated with the family set is recycled. The family set is the fourth field in a server's **mcf** file. If an *archive_set* is specified, that archive set is recycled. The *archive_set* specified must include the copy number, as stated in the **/etc/opt/SUNWsamfs/archiver.cmd** file. For example, **arset.1**.

If no *family_set* or *archive_set* name is specified, the recycler recycles according to specifications in the **/etc/opt/SUNWsamfs/archiver.cmd** and the **/etc/opt/SUNWsamfs/recycler.cmd** files. It examines each library and archive set specified.

Regardless of a specification, only archive sets and family sets that have a current usage that is less than the high-water mark are recycled.

OPERATION

The recycler splits its work into two phases: volume selection and volume recycling.

Phase 1 - Volume Selection
The recycler selects volumes for recycling based on the amount of space used by expired archive copies as a percentage of total space on a volume. For each library or archive set being recycled, the volumes with the highest percentages of expired copies are selected to bring the media utilization in the library or archive set below the configured high-water-mark. This assumes that each volume selected would contribute at least *VSN-minimum-percent-gain* percent of its total space if it were recycled. If no such volumes exist, the library or archive set cannot be recycled. Ties in expired space are resolved by selecting the volumes with the least amount of unexpired space. For more information on setting a high water mark, see the **recycler.cmd**(4) man page.

A few conditions can prevent a volume from being selected. A volume cannot be recycled if it contains data associated with a removable media file created by the **request**(1) command. In addition, it cannot be recycled if it is listed in the **/etc/opt/SUNWsamfs/recycler.cmd** file's **no_recycle** section.

After volumes have been selected, they are recycled.

Phase 2 - Volume Recycling
Volume recycling differs depending upon whether the archive media is a disk volume or whether it is a removable cartridge in a library. Archiving to disk volumes is described first.

When a disk volume is selected for recycling, the volume is not marked for recycling. Additional archive copies can be written to it. Expired archive copies on the disk volume are identified and removed. Valid archive copies are left alone.

When a tape or magneto optical volume is selected for recycling, the system prevents additional archive copies from being written to it. If you are recycling to cartridges in a library, all files with active archive copies in volumes on the cartridges are marked to be re-archived. The archiver moves these copies to other volumes. In subsequent runs, the recycler checks these volumes and post-processes them when all valid archive copies have been relocated.

The recycler checks to see if there are volumes that were selected for recycling that have not yet been post-processed. If such volumes exist, and they are now devoid of active archive copies, the **sam-recycler** command invokes the **/opt/SUNWsamfs/sbin/recycler.sh**(4), which post-processes these volumes with arguments including the generic media type (**tp** or **od**), the VSN, the element address in the library, and the equipment number of the library in which the volume resides. The script can relabel the cartridge using either the original VSN or a new VSN; or it can export the cartridge from the library; or it can perform another user-defined action.

The **/opt/SUNWsamfs/sbin/recycler.sh**(4) script clears the **recycling** flag to indicate that recycling has completed on the volume. The **odlabel**(1M) and **tplabel**(1M) commands clear this flag after the cartridge has been relabeled.

RECYCLER OUTPUT

The recycler log is divided into several sections.

The first section describes each library catalog and archive set. The header contains the family set name or archive set name and the vendor, product, and catalog path name. Then, the capacity and remaining space for each volume appears, in bytes, with suffixes **k**, **M**, **G**, and **T** representing kilobytes, megabytes, gigabytes, and terabytes, respectively. In this log file, a kilobyte=1024 bytes, a megabyte=1024∗1024 bytes, and so on. Then, a summary, containing the total capacity and total space remaining is shown in bytes and as a percentage of space used. The recycling parameters set in the recycler and archiver command files are also shown.

The second section is a series of tables, one for each library and archive set that has associated volumes. The name of the library or archive set is shown just to the right of the **----Percent----** label. A volume can be associated with only one library or archive set. Attempts to assign a volume to multiple archive sets are marked with a **in multiple sets** label. The following fields are displayed:

Field Name	**Meaning**
Status	A phrase giving the volume's recycle status, as follows:

empty VSN	The volume is empty of both expired and current archive images
full VSN	The volume has no free space, but it does have current archive images.
in multiple sets	The volume matches multiple archive sets in the **/etc/opt/SUNWsamfs/archiver.cmd** file.
new candidate	The volume was chosen for recycling during this recycler run.
no-data VSN	The volume contains only expired archive images and free space.
no_recycle VSN	The volume is listed in the **no_recycle** section of the **/etc/opt/SUNWsamfs/recycler.cmd** file.
archive -n files	The volume contains archive images for files now marked as **archive -n**.
old candidate	The volume was already marked for recycling before this recycler run.
request files	The volume contains archive images for removeable media files.
partially full	The volume contains both current archive images and free space.
shelved VSN	The volume is not currently located in any library.

Archives Count	The number of archive copies that are contained on this volume.

Archives Bytes The number of bytes of archive copies contained on this volume.

Percent Use The percentage of space in use on this volume by current archive copies. It is estimated by summing up the sizes of the archive copies on the medium. Because of compression, this value can overstate the amount of space actually used by these images. This is the amount of data that would need to be moved if the volume were selected for recycling.

Percent Obsolete
The percentage of space used on this volume for which no archive copies were found. This is the space that can be reclaimed by recycling this cartridge.

The **Percent Obsolete** value is calculated as follows:

100% - **In Use - Free**

Because **In Use** can overstate the actual space used (because of compression), the sum of **In use + Free** can exceed 100%, which renders **Percent Obsolete** to be a negative value. Although aesthetically unpleasing, this does not cause any problems in the operation of the recycler.

Percent Free The percentage of free space remaining on this volume. This value comes directly from the library catalog. It gives the percent of the volume's total capacity that is available to hold new archive images.

For media that supports data compression, a best-guess value of the average compression is calculated from the ratio of the number of physical tape blocks consumed on the volume (that is, the difference of **capacity - space**) to the logical number of tape blocks written to the volume. The latter value is kept in the catalog. This ratio is then used to adjust the **In Use** value before it is written to the log file.

The first volume to appear in the log file, for each library or archive set, is the one most in need of recycling.

Here is an example recycler log file:

```
========== Recycler begins at Thu Feb  5 13:40:20 1998 ===========
3 catalogs:

0  Family: hy                   Path: /tmp/y
   Vendor: SAM-FS               Product: Historian
   EA                    ty    capacity         space vsn
      (no VSNs in this media changer)
   Total Capacity:  0    bytes, Total Space Available: 0    bytes
   Media utilization 0%, high 0% VSN_min 0%




1  Family: ad40                 Path: /var/opt/SUNWsamfs/catalog/ad40
   Vendor: ADIC                 Product: Scalar DLT 448
   EA                    ty    capacity         space vsn
      0                  lt        19.2G         0    DLT3
      1                  lt        17.7G        17.6G DLT4N
      5                  lt        17.7G        17.6G DLT6
   Total Capacity:  54.6G bytes, Total Space Available: 35.2G bytes
   Media utilization 35%, high 75% VSN_min 50%
```

```
2  Family: arset0.1             Path: /etc/opt/SUNWsamfs/archiver.cmd
   Vendor: SAM-FS               Product: Archive set
   EA                    ty    capacity         space vsn
      0                  lt        0             0    DLT5
      1                  lt       19.2G          0    DLT3
      2                  lt        0             0    DLT2
      3                  lt       17.7G        17.6G DLT4N
      4                  lt       17.7G        17.6G DLT6
   Total Capacity:  54.6G bytes, Total Space Available: 35.2G bytes
   Media utilization 35%, high 80% VSN_min 50%
   Send mail to root when this archive set needs recycling.



6 VSNs:

                    ---Archives---   -----Percent-----
-----Status-----    Count    Bytes   Use Obsolete Free   Library:Type:VSN
shelved VSN          677    648.9M                       <none>:lt:DLT0

                    ---Archives---   -----Percent-----   arset0.1
-----Status-----    Count    Bytes   Use Obsolete Free   Library:Type:VSN
no-data VSN            0      0        0   100      0    ad40:lt:DLT3
empty VSN              0      0        0     0      0  (NULL):lt:DLT2
empty VSN              0      0        0     0    100    ad40:lt:DLT6
full VSN               4     32.1k     0     0      0  (NULL):lt:DLT5
partially full         4     40.8k     0     0    100    ad40:lt:DLT4N



Recycler finished.

========== Recycler ends at Thu Feb  5 13:40:41 1998 ===========
```

Here is the corresponding **archiver.cmd** file:

```
interval = 2m
no_archive .
fs = samfs1
arset0 testdir0
     1 1s
     2 1s
     3 1s
     4 1s
no_archive .
fs = samfs2
no_archive .
vsns
arset0.1 lt DLT3 DLT4N DLT6 DLT1
arset0.2 lt DLT3 DLT4N DLT6 DLT1
arset0.3 lt DLT3 DLT4N DLT6 DLT1
arset0.4 lt DLT3 DLT4N DLT6 DLT1
samfs1.1 lt DLT3
samfs2.1 lt DLT4N
endvsns
```

```
params
arset0.1 -drives 4 -recycle_hwm 80 -recycle_mingain 50
endparams
```

Here is the corresponding **/etc/opt/SUNWsamfs/recycler.cmd** file:

```
logfile = /var/tmp/recycler.log
ad40 75 50
no_recycle mo ^OPT003
```

RECYCLING HISTORIAN CARTRIDGES

The recycler recycles volumes listed in the historian's catalog. The volumes listed in the historian catalog have been exported from a library or have been or are currently in a manually-mounted device.

The **/opt/SUNWsamfs/sbin/recycler.sh**(4) script is passed the name **hy**, signifying volumes that reside in the historian catalog so that it can cope with the possibility of the volumes being recycled residing in an off-site storage facility. Typically, the **/opt/SUNWsamfs/sbin/recycler.sh**(4) script sends email to the administrator when this occurs to remind the administrator to bring the off-site volume back on site so that it can be reused. Volumes do not need to be on site to be drained of archive copies unless such a volume contains the only available archive copy of an off-line file.

RECYCLING BY ARCHIVE SET

When the recycler recycles by archive set, it treats each archive set as a small library that holds just the volumes assigned to the archive set in the **/etc/opt/SUNWsamfs/archiver.cmd** file. The volumes that are identified as belonging to a recycling archive set are removed from the recycler's version of the catalog for the library that physically contains the volume. Thus, only the volumes that are not part of an archive set remain in the library catalog.

To enable recycling for a given archive set, it must have one of the recycling options specified in the **/etc/opt/SUNWsamfs/archiver.cmd** file. For more information, see the **archiver.cmd**(4) man page.

MESSAGES

Consider the following message:

```
Jan 22 10:17:17 jupiter sam-recycler[3400]: Cannot ioctl(F_IDSCF)
      Cannot find pathname for filesystem /samfs1 inum/gen 406/25
```

The preceding message means that the recycler could not set the **rearchive** flag for a file. When this happens, the recycler typically emits a message containing the path name, as follows:

```
Jan 22 10:17:17 jupiter sam-recycler[3400]: Cannot ioctl(F_IDSCF)
      /samfs1/testfile
```

However, in the first message, you see text beginning with **Cannot find pathname...**. This means that the recycler failed in its attempt to convert the inode number (in the preceding example message, it is inode number 406) and generation number (here, 25) into a path name in the **/samfs1** file system.

The most likely reason for this to occur is that the file was deleted between the time that the recycler determined it needed to be rearchived and the time the recycler actually issued the system call to set the rearchive flag.

SEE ALSO

chmed(1M), **odlabel**(1M), **sam-archiverd**(1M), **tplabel**(1M).

archiver.cmd(4), **mcf**(4), **recycler.cmd**(4), **recycler.sh**(4).

NAME

sam-releaser − Sun SAM-FS and Sun SAM-QFS disk space releaser process

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-releaser *file_system low_water_mark weight_size* [*weight_age*]

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **sam-releaser** process controls the activities of the Sun SAM-FS or Sun SAM-QFS releaser. The releaser makes disk cache available by identifying archived files and releasing their disk cache copy. This process is started automatically by the file system when disk cache utilization reaches the high-water mark.

If the releaser command file is present in **/etc/opt/SUNWsamfs/releaser.cmd**, the **sam-releaser** process reads that file to determine whether it contains directives that override command-line arguments. For more information on the releaser command file, see the **releaser.cmd**(4) man page.

OPTIONS

This command accepts the following arguments:

file_system
: This is the file system whose disk space is to be released. The argument may be either the name of the file system, or it's mount_point. The releaser attempts to release the disk space of archived files on the file system mounted on the mount_point until *low_water_mark* is reached.

low_water_mark
: A percentage of the file system that is allowed to be completely occupied with files at all times. Specify an integer number that is at least **0** but no more than **100**. The releaser attempts to release disk space until the file system is at or below this threshold.

weight_size
: A weighting factor that is used to prioritize release candidates. Specify a floating-point value that is at least **0.0** but no more than **1.0**. For more information on *weight_size*, see the PRIORITY WEIGHTS section of this man page.

weight_age
: A weighting factor that is used to prioritize release candidates. Specify a floating-point value that is at least **0.0** but no more than **1.0**. For more information on *weight_age*, see the PRIORITY WEIGHTS section of this man page.

ALGORITHM

The releaser reads the Sun SAM-FS or Sun SAM-QFS **.inodes** file and builds an ordered list of the files that can be released. The position of each file on the list depends on a priority calculated for each inode by the releaser (see the PRIORITY WEIGHTS section of this man page.) Only the top 10,000 files are kept on the list.

Starting with the file with the numerically largest priority, the disk space used by each file is released until the *low_water_mark* has been reached. If the list is exhausted before the *low_water_mark* is reached, the process is repeated. If, while repeating the process, no files are found that can be released, the releaser stops. If the file system is still above high-water mark, the file system restarts the releaser.

PRIORITY WEIGHTS

Each inode is assigned a priority based on its size and age. The size of the file (expressed in units of 4-kilobyte blocks) is multiplied by the *weight_size* parameter. This result is added to the priority calculated for the age of the file to form the file's final priority.

The releaser can use one of the following two methods for determining the contribution of the age of a file to the file's release priority:

- The first method is to take the most recent of the file's access, modification, and residence-change age and multiply by *weight_age*.

- The second method allows specification of weights for each of the access, modification, and residence-change times. These are specified by the **weight_age_access=***float*, **weight_age_modify=***float*, and **weight_age_residence=***float* directives, respectively, in the **releaser.cmd** file. The sum of the product of the weight and corresponding age is the contribution of the age to the file's priority. To specify any of these priority weights, you must use the **releaser.cmd** file. For information on the **releaser.cmd** file, see the **releaser.cmd**(4) man page.

For both methods, the ages are expressed in minutes.

LOG

Within the **releaser.cmd** file, you can specify a log file for each Sun SAM-FS or Sun SAM-QFS file system. If the **releaser.cmd** file does not exist, or if no **logfile=***filename* directive exists in the file, no logging occurs. For more information on the **logfile=***filename* directive, see the **releaser.cmd**(4) man page.

The releaser creates the log file (if it does not exist) and appends the following to it for each run:

```
Releaser begins at Tue Sep 29 15:31:15 1998
inode pathname          /sam1/.inodes
low-water mark          40%
weight_size             1
weight_age              0.5
fs equipment ordinal    1
family-set name         samfs1
started by sam-fsd?     no
release files?          no
release rearch files?   yes
display_all_candidates? no
---before scan---
blocks_now_free:     117312
lwm_blocks:          233750
---scanning---
64122.5 (R: Tue Sep 29 11:33:21 CDT 1998) 237 min, 64004 blks S0 /sam1/250m
5131.5 (R: Tue Sep 22 17:39:47 CDT 1998) 9951 min, 156 blks S0 /sam1/filecq
5095.5 (R: Tue Sep 22 17:39:49 CDT 1998) 9951 min, 120 blks S0 /sam1/filecu
5062 (R: Tue Sep 22 18:38:50 CDT 1998) 9892 min, 116 blks S0 /sam1/filebz
5039.5 (R: Tue Sep 22 17:40:01 CDT 1998) 9951 min, 64 blks S0 /sam1/filedi
5036.5 (R: Tue Sep 22 17:37:34 CDT 1998) 9953 min, 60 blks S0 /sam1/fileio
5035.5 (R: Tue Sep 22 17:40:13 CDT 1998) 9951 min, 60 blks S0 /sam1/filedw
5032.5 (R: Tue Sep 22 17:38:08 CDT 1998) 9953 min, 56 blks S0 /sam1/filejq
5031.5 (R: Tue Sep 22 17:39:56 CDT 1998) 9951 min, 56 blks S0 /sam1/fileda
5024.5 (R: Tue Sep 22 17:38:00 CDT 1998) 9953 min, 48 blks S0 /sam1/filejh
5024 (R: Tue Sep 22 17:38:22 CDT 1998) 9952 min, 48 blks S0 /sam1/fileka
5023.5 (R: Tue Sep 22 17:40:07 CDT 1998) 9951 min, 48 blks S0 /sam1/filedn
5019 (R: Tue Sep 22 17:40:44 CDT 1998) 9950 min, 44 blks S0 /sam1/filefk
5015 (R: Tue Sep 22 17:40:28 CDT 1998) 9950 min, 40 blks S0 /sam1/fileep
5011.5 (R: Tue Sep 22 17:40:14 CDT 1998) 9951 min, 36 blks S0 /sam1/filedx
5011.5 (R: Tue Sep 22 17:39:58 CDT 1998) 9951 min, 36 blks S0 /sam1/filede
5011 (R: Tue Sep 22 17:41:07 CDT 1998) 9950 min, 36 blks S0 /sam1/filegk
5007.5 (R: Tue Sep 22 17:39:51 CDT 1998) 9951 min, 32 blks S0 /sam1/filecw
5007 (R: Tue Sep 22 17:41:10 CDT 1998) 9950 min, 32 blks S0 /sam1/filegr
5007 (R: Tue Sep 22 17:40:42 CDT 1998) 9950 min, 32 blks S0 /sam1/filefg
5007 (R: Tue Sep 22 17:40:30 CDT 1998) 9950 min, 32 blks S0 /sam1/filees
5004.5 (R: Tue Sep 22 17:38:14 CDT 1998) 9953 min, 28 blks S0 /sam1/filejv
5004 (R: Tue Sep 22 17:38:57 CDT 1998) 9952 min, 28 blks S0 /sam1/filelm
```

```
5002 (R: Tue Sep 22 18:38:54 CDT 1998) 9892 min, 56 blks S0 /sam1/filecd
4996.5 (R: Tue Sep 22 17:38:06 CDT 1998) 9953 min, 20 blks S0 /sam1/filejp
4995.5 (R: Tue Sep 22 17:39:57 CDT 1998) 9951 min, 20 blks S0 /sam1/filedc
4992.5 (R: Tue Sep 22 17:37:24 CDT 1998) 9953 min, 16 blks S0 /sam1/fileig
4992 (R: Tue Sep 22 17:39:06 CDT 1998) 9952 min, 16 blks S0 /sam1/filelv
4986 (R: Tue Sep 22 18:38:50 CDT 1998) 9892 min, 40 blks S0 /sam1/fileca
4982 (R: Tue Sep 22 17:36:54 CDT 1998) 9954 min, 5 blks S0 /sam1/filehk
4981 (R: Tue Sep 22 17:41:09 CDT 1998) 9950 min, 6 blks S0 /sam1/filegn
4980.5 (R: Tue Sep 22 17:40:15 CDT 1998) 9951 min, 5 blks S0 /sam1/filedz
---after scan---
blocks_now_free:       0
lwm_blocks:            233750
archnodrop: 0
already_offline: 647
bad_inode_number: 0
damaged: 0
extension_inode: 0
negative_age: 0
nodrop: 0
not_regular: 7
number_in_list: 32
rearch: 1
released_files: 32
too_new_residence_time: 0
too_small: 1
total_candidates: 32
total_inodes: 704
wrong_inode_number: 14
zero_arch_status: 3
zero_inode_number: 0
zero_mode: 0
CPU time: 0 seconds.
Elapsed time: 1 seconds.


Releaser ends at Tue Sep 29 15:31:16 1998
```

The first block of lines shows the arguments with which the releaser was invoked, the name of the **.inodes** file, the low-water mark, the size and age weight parameters, the equipment ordinal of the file system, the family set name of the file system, whether the releaser was started by **sam-fsd** or by the command line, whether files should be released, and whether each inode should be logged as encountered.

The second block of lines begins with the heading **---before scan---**. It shows the number of blocks currently free in the cache and the number that would be free if the file system were exactly at the low-water mark. The goal of the releaser is to increase **blocks_now_free** so that it is equal to or larger than **lwm_blocks**.

The third block of lines begins with the heading **---scanning---**. This block lists the files released by the releaser and contains information for each file in separate fields. The fields are as follows:

Field Number	**Content**
1	This field contains the release priority.
2	This field contains the date and time in the following format: **(***tag***:** *date_and_time***)**. The *tag* is either **A** for access, **M** for modify, or **R** for residency, depending on if the

	date that follows represents the access, modify or residency time. The *date_and_time* is the most recent of the three dates listed.
3	This field contains the age and size of the file. The age of the file is expressed in minutes. The size of the file is expressed in blocks. These two figures are multiplied by their respective weights and the sum taken to yield the release priority.
4	This field contains an **S** followed by the segment number. This is the number of the segment that was released.
5	This field contains the full path name of the released file.

Note that if the **weight_age_access=***float*, **weight_age_modify=***float* or **weight_age_residence=***float* directives are specified in the **releaser.cmd** file, these lines show only the priority, size, and pathname.

The fourth block of lines begins with the heading **---after scan---**. This block shows the statistics accumulated by the releaser during the previous scan pass are shown. These statistics are as follows:

Statistic	**Meaning**
archnodrop	The number of inodes marked **archnodrop**. These files are never released because the archiver is trying to keep them in cache.
already_offline	The number of inodes that were offline.
bad_inode_number	This field is not used and is always zero.
damaged	The number of inodes marked as damaged.
extension_inode	The number of extension inodes found. Used by volume overflow.
negative_age	The number of inodes that had an age in the future. This is usually caused by personal computers with incorrect clock settings acting as NFS clients.
nodrop	The number of inodes marked with **release -n**. For more information on marking files as never release, see the **release**(1) man page.
not_regular	The number of inodes that were not regular files.
number_in_list	The number of inodes that were on the releaser's candidate list when the releaser was finished scanning.
rearch	The number of files with a copy marked for rearchiving.
released_files	The number of files released.
too_new_residence_time	The number of inodes whose residence-change time was within minimum residence age of the current time as specified on the **min_residence_age=***time* directive in the **releaser.cmd** file.
too_small	The number of files that were too small to be released.
total_candidates	The number of inodes found that were viable candidates for releasing.
total_inodes	The total number of inodes scanned.
wrong_inode_number	The number of inodes whose inode number did not match their offset in the inode file. This is usually not a concern, but you should run **samfsck**(1M) to rescue any orphan inodes. If you have already run **samfsck**(1M) and this field remains nonzero, no further action is required. For more information on the **samfsck**(1M) command, see the samfsck(1M) man page.
zero_arch_status	The number of inodes that had no archive copies.
zero_inode_number	The number of inodes that had zero as their inode number.
zero_mode	The number of inodes that were unused.

CPU time	The number of CPU seconds used in the current scan.
Elapsed time	The number of wall-clock seconds used in the current scan.

NOTES

When a file is created, the residency age is set to the creation time. The residency age of a file must be at least the value set by the **min_residence_age=***time* directive before the file is considered for release. This is to prevent a file which was recently staged in from being released. The default *time* is 10 minutes.

If the releaser selects a file as a release candidate, and immediately thereafter the file is accessed, the file might still be released by the file system even though the file has been recently accessed. This can happen because the file system only prohibits release of a file that is currently in use. It does not check the access age of the file again when it is released.

SEE ALSO

release(1).

mount_samfs(1M), **samfsck**(1M).

releaser.cmd(4).

NAME

sam-robotsd, sam-genericd, sam-stkd, sam-ibm3494d, sam-sonyd − Sun SAM-FS and Sun SAM-QFS media changer daemons

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-robotsd *mshmid pshmid*

/opt/SUNWsamfs/sbin/sam-genericd *mshmid pshmid equip*

/opt/SUNWsamfs/sbin/sam-stkd *mshmid pshmid equip*

/opt/SUNWsamfs/sbin/sam-ibm3494d *mshmid pshmid equip*

/opt/SUNWsamfs/sbin/sam-sonyd *mshmid pshmid equip*

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **sam-robotsd** daemon starts and monitors the execution of the media changer library control daemons for Sun SAM-FS and Sun SAM-QFS. The **sam-robotsd** daemon is started automatically by the **sam-initd** daemon if there are any libraries defined in the **mcf** file. The **sam-robotsd** daemon starts and monitors the correct daemon for all defined libraries. For more information on the **mcf** file, see the **mcf**(4) man page.

Each library daemon is responsible for monitoring the preview table for the VSNs that are controlled by that daemon. If a request is found for one of its VSNs, the daemon finds an available drive under its control and moves the cartridge into that drive. When the device is ready, the daemon notifies the Sun SAM-FS or Sun SAM-QFS library daemon, and the device is assigned to the waiting process.

The identifiers are as follows:

mshmid	The identifier of the master shared memory segment created by the **sam-initd** daemon.
pshmid	The identifier of the preview shared memory segment created by the **sam-initd** daemon.
equip	The equipment number of the device.

The **sam-genericd** daemon controls libraries that conform to the SCSI II standard for media changers, and it is the daemon that controls the ADIC/Grau ABBA library through the **grauaci** interface. For more information on this interface, see the **grauaci**(7) man page.

The **sam-stkd** daemon controls StorageTek libraries through the ACSAPI interface and is included in the **SUNWsamfs** package. For more information on this interface, see the **stk**(7) man page.

The **sam-ibm3494d** daemon controls IBM 3494 tape libraries through the **lmcpd** interface and is included in the **SUNWsamfs** package. For more information on this interface, see the **ibm3494**(7) man page.

The **sam-sonyd** daemon controls Sony libraries through the Sony DZC-800S PetaSite Application Interface Library and is included in the **SUNWsamfs** package. For more information on this interface, see the **sony**(7) man page.

FILES

mcf	The master configuration file for Sun SAM-FS and Sun SAM-QFS environments.

SEE ALSO

sam-initd(1M).

mcf(4).

acl2640(7), **acl452**(7), **grauaci**(7), **ibm3494**(7), **sam_remote**(7), **sony**(7), **stk**(7).

NAME

sam-rpcd − Sun SAM-FS and Sun SAM-QFS RPC API server process

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-rpcd

AVAILABILITY

SUNWsamfs

DESCRIPTION

sam-rpcd is the RPC API (Application Programmer Interface) server process. It is initiated by **sam-init**.

sam-rpcd uses the RPC program number that is paired with the RPC program name *samfs*. **sam-rpcd** must run on the same machine as Sun SAM-FS or Sun SAM-QFS file system. You need to make the following entry in /etc/services on the server:

```
samfs       5012/tcp       # SAM-FS API
```

And in /etc/rpc on client and server:

```
samfs       150005
```

Make the equivalent changes in the NIS databases if you run NIS.

SEE ALSO

sam_initrpc(3x)

NAME

sam-scannerd − Sun SAM-FS and Sun SAM-QFS daemon for manually-mounted devices

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-scannerd *mshmid pshmid*

AVAILABILITY

SUNWsamfs

DESCRIPTION

sam-scannerd monitors the manually-mounted devices. It will periodically check each device for newly inserted media. If **sam-scannerd** finds media in the device, it will scan it for a label. If a label is found, it will check the preview table to see if there are any requests for this media. If requests are found, the Sun SAM-FS or Sun SAM-QFS file system is notified and the device is assigned to the request.

sam-scannerd is started automatically by **sam-init** if there are any manually-mounted devices defined in the configuration file. See **mcf**(4).

mshmid is the id of the master shared memory segment created by **sam-init**. *pshmid* is the id of the preview shared memory segment created by **sam-init**.

SEE ALSO

sam-init(1M), **mcf**(4)

NAME

sam-sharefsd − Invokes the Sun QFS or Sun SAM-QFS shared file system daemon

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-sharefsd

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **sam-sharefsd** process establishes connection to the current metadata server in a Sun QFS or Sun SAM-QFS shared file system. The **sam-sharefsd** process on the metadata server opens a listener socket on the port associated with this file system. The shared file system port is defined in **/etc/services** as **samsock.***fs_name*.

The Sun QFS and Sun SAM-QFS shared file system is a distributed file system that can be mounted on Solaris host systems.

The **sam-sharefsd** process is initiated by the **sam-fsd** daemon. The **sam-fsd** daemon starts a shared file system daemon for each configured shared file system.

FILES

Detailed trace information is written to the **sam-sharefsd** trace file.

SEE ALSO

sam-fsd(1M).

sammkfs(1M).

samsharefs(1M).

NAME

sam-stagealld − Sun SAM-FS and Sun SAM-QFS associative staging daemon

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-stagealld

AVAILABILITY

SUNWsamfs

DESCRIPTION

sam-stagealld is responsible for the associative staging feature. It is initiated by **sam-init**. Associative staging is activated when a regular file that has the associative staging attribute set is staged. All files in the same directory that have the associative staging attribute set are staged. If a symbolic link has the associative staging attribute set, the file pointed to by the symbolic link is staged.

SEE ALSO

stage(1), **sam-init**(1M)

NAME

sam-stagerd − Invokes the Sun SAM-FS or Sun SAM-QFS stage daemon

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-stagerd

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **sam-stagerd** process stages files in a Sun SAM-FS or Sun SAM-QFS file system. *Staging* is the process of copying a nearline or offline file from its archive storage back to online storage.

The Sun SAM-FS and Sun SAM-QFS file system staging capability allows you to stage files immediately, to never stage files, and specify other staging actions. The **sam-stagerd** process is initiated by the **sam-fsd** daemon.

By default, the stager uses the default behaviors described on the **stager.cmd**(4) man page.

FILES

If the stager command file is present in **/etc/opt/SUNWsamfs/stager.cmd**, the **sam-stagerd** process reads that file.

SEE ALSO

stage(1).

sam-fsd(1M).

stager.cmd(4).

NAME

sam-stagerd_copy − Invokes the Sun SAM-FS and Sun SAM-QFS stage copy daemon

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-stagerd_copy

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **sam-stagerd_copy** process copies Sun SAM-FS and Sun SAM-QFS files from removable media cartridges. It is executed by the **sam-stagerd**(1M) process.

SEE ALSO

sam-stagerd(1M)

NAME

sambcheck − Lists block usage for a Sun SAM-FS and Sun SAM-QFS file system

SYNOPSIS

sambcheck *fs_name block_id* [*block_id* ...]

AVAILABILITY

SUNWqfs

SUNWsamfs

DESCRIPTION

The **sambcheck** command determines the current usage of each requested *block_id* in a Sun SAM-FS or Sun SAM-QFS file system. This command must be run as root. For accurate results, the file system should be unmounted.

This command accepts the following arguments:

fsname The family set name, as specified in the **mcf** file, for the file system for which the usage list is desired.

block_id Specifies one or more *block_id* identifiers in the following format:

block_number[**.***ordinal*]

For *block_number*, specify a number to identify the blocks for which statistics should be obtained. Use one of the following formats:

- Decimal. Default.
- Octal. The *block_number* must be preceded by **0**.
- Hexadecimal. The *block_number* must be preceded by **0x** or **0X**.

For *ordinal*, specify the partition number upon which the block usage is to be found. If no **.***ordinal* is specified, all partitions are examined. All *ordinal* specifications are assumed to be in decimal.

OUTPUT

The output from this command is one line per requested block number for each explicit or implicit ordinal. The block number is displayed as entered, followed by its decimal form in parentheses, followed by text indicating the usage determined for the *block_id*.

EXAMPLES

```
bilbo# sambcheck samfs1 0x40 0x42.0 0x42.2 0x7a150 0x89cd0.01 512
block 0x40 (64.0) is a data block for .inodes containing 1 - 32
block 0x40 (64.1) is a data block for directory inode 26.1
block 0x40 (64.2) is a data block for inode 934767.1
block 0x40 (64.4) is a data block for inode 934766.1
block 0x42.0 (66.0) is a data block for .inodes containing 1 - 32
block 0x42.2 (66.2) is a free data block
block 0x7a150 (500048.0) is a data block for .inodes containing 999969 - 1000000
block 0x7a150 (500048.1) is a data block for directory inode 787628.1
block 0x7a150 (500048.2) is a data block for inode 934767.1
block 0x7a150 (500048.4) is a free data block
block 0x89cd0.01 (564432.1) is an indirect block for inode 934767.1
block 512 (512.0) is a data block for .inodes containing 897 - 928
block 512 (512.1) is a data block for directory inode 65.1
block 512 (512.2) is a data block for inode 934767.1
block 512 (512.4) is a data block for inode 934766.1
```

NAME

samchaid − change file admin set ID attribute

SYNOPSIS

samchaid [**−fhR**] aid *filename*. . .

AVAILABILITY

SUNWsamfs

SUNWqfs

DESCRIPTION

samchaid sets the admin set ID attribute of files and directories.

If a directory's admin set ID is set, files and directories subsequently created in that directory inherit that admin ID. Only the superuser may set the admin ID.

OPTIONS

−f	Force. Do not report errors.
−h	If the file is a symbolic link, change the admin set ID of the symbolic link. Without this option, the group of the file referenced by the symbolic link is changed.
−R	Recursive. samchaid descends through any directories and subdirectories, setting the specified admin set ID as it proceeds. When a symbolic link is encountered, the admin set ID of the target file is changed (unless the -h option is specified), but no recursion takes place.

SEE ALSO

samquota(1), **sls**(1)

NAME

samcmd − execute Sun SAM-FS and Sun SAM-QFS operator utility command

SYNOPSIS

samcmd *command*

AVAILABILITY

SUNWqfs
SUNWsamfs

DESCRIPTION

samcmd executes a single Sun SAM-FS or Sun SAM-QFS operator utility command. Its purpose is to provide shell script access to the commands and displays available in **samu**(1M).

samcmd uses the first argument as the **samu** command or display name. Succeeding arguments are the arguments for that **samu** command.

COMMANDS

The syntax for the commands is identical to that shown in the COMMANDS section of **samu**(1M). Note that the colon (:) hot key is not required for samcmd to distinguish commands from displays.

DISPLAYS

samcmd can produce displays on standard output similar to those displayed by **samu**. While for **samu** the information is paged to display a screen at a time if there is more than one screen of information available, **samcmd** produces the entire amount of information for a given display. Hence there is no need for equivalents of the control-f, control-b, control-d, and control-u hotkeys. Note that the formatting of the information may be slightly different on the **samcmd** output file than on the **samu** display. Since the format of the display control (single letter) commands can be modified by other hotkeys under **samu**, some equivalents are provided for **samcmd** as follows:

Display	Arguments
a	filesystem
n	mediatype
p	mediatype
r	mediatype
u	mediatype [path]
v	eq [sort] [I | I I]
w	mediatype [path]

The sort selections for the v display are: 1 slot, 2 count, 3 usage, 4 VSN, 5 access time, 6 barcode, 7 label time. Specifying a single I for the v display shows a two-line display with the barcode, blocksize, etc. in the second line. Specifying two I's for the v display shows a two-line display with the archiver volume reservation information in the second line.

EXAMPLES

The following example loads a cartridge from slot 2 in automated library 30:

```
samcmd load 30:2
```

The following example produces a detailed archiver display for filesystem samfs3 on standard output:

```
samcmd a samfs3
```

The following example produces a display, on standard output, of the staging queue restricted to stages from media type "lt", showing the full paths of the files to be staged.

```
samcmd u lt path
```

The following example produces a display of automated library 50’s catalog, with the archiver volume reservation information, on standard output:

```
samcmd v 50 I I
```

SEE ALSO

samu(1M)

NAME

samd − Starts or stops the **sam-initd** daemon

SYNOPSIS

/opt/SUNWsamfs/sbin/samd [config | start | stop]

AVAILABILITY

SUNWqfs

SUNWsamfs

DESCRIPTION

The **samd** utility starts up or shuts down the **sam-initd** daemon. This utility can also be used to reinitialize the Sun QFS, Sun SAM-FS, and Sun SAM-QFS configuration files and allow changes to take effect.

OPTIONS

This command accepts the following options:

config Causes the the **sam-fsd** daemon to (re)configure based on changes to the **/etc/opt/SUNWsamfs/mcf** and **/etc/opt/SUNWsamfs/defaults.conf** files.

start Starts up the **sam-initd** daemon if the **/etc/opt/SUNWsamfs/mcf** file exists and the **sam-initd** daemon is not already running. Implemented only in Sun SAM-FS and Sun SAM-QFS environments. Not a valid argument in a Sun QFS environment.

stop Kills the **sam-initd** daemon. Implemented only in Sun SAM-FS and Sun SAM-QFS environments. Not a valid argument in a Sun QFS environment.

SEE ALSO

sam-fsd(1M), **sam-initd**(1M).

defaults.conf(4), **mcf**(4).

NAME

samdev − Adds **/dev/samst** entries for media changers and optical disks attached to the system

SYNOPSIS

/opt/SUNWsamfs/sbin/samdev [-d]

AVAILABILITY

SUNWsamfs

OPTIONS

−d	Print descriptive messages when creating each symbolic link.

DESCRIPTION

samdev creates symbolic links in the **/dev/samst** directory pointing to the actual device special files under the **/devices** directory tree. It performs the following steps:

1. The **/dev/samst** directory is checked for device entries − that is, symbolic links with names of the form **c***N***t***X***u***N*, where *N* represents a decimal number, and *X* represents a hexadecimal number.

   c*N* is the logical controller number, an arbitrary number assigned by this program to designate a particular controller. The first controller found on the first occasion this program is run on a system, is assigned number 0. This number may not be the same number as the number assigned by the **disks**(1M) program. **t***X* is the SCSI **target** for the device. **u***N* is the SCSI logical unit number of the device at this SCSI target. Each entry's symbolic link is read and placed in an internal pathname table.

2. Searches the **/devices** directory searching for devices that are supported by **samst**(7). It adds the pathnames for the devices to the table built in step 1.

3. **samdev** then traverses the pathname table and builds symbolic links in **/dev/samst** for each device in the table.

samdev should be executed each time the system is reconfiguration-booted. An entry can be added to the **/etc/init.d/devlinks** file after the entry for /usr/sbin/tapes to automate this process. **samdev** can only be run after **drvconfig**(1M) is run, since **drvconfig**(1M) builds the **/devices** tree.

This utility does not apply to Fibre Channel devices.

FILES

/dev/samst/∗	entries for general use
/devices/∗	device nodes

SEE ALSO

disks(1M), **devlinks**(1M), **drvconfig**(1M), **ports**(1M), **tapes**(1M)

NAME

samfsck − Check and repair a Sun SAM-FS and Sun SAM-QFS file system

SYNOPSIS

samfsck [**−s** *scratch_dir*] [**−F** [**−R**]] [**−G**] [**−V**] *fs_name*

AVAILABILITY

SUNWsamfs

DESCRIPTION

samfsck checks and optionally repairs a SAM-FS file system from the disk partitions that belong to the family set *fs_name*. *fs_name* is the family set name from the **mcf** file. One or more disk partitions are specified in the **mcf** file. If no options are specified, **samfsck** checks and reports, but does not repair, all the blocks that belong to inodes and lists inodes which have duplicate blocks. samfsck also checks inodes which have blocks that are free blocks. If only one inode is listed in the duplicate list, that inode contains a block that is also free. The file system must be *unmounted.*

If there are files encountered that are not attached to a parent directory, they will be moved to the /*mount_point*/lost+found directory. If this directory does not exist, you must create this directory first and make it sufficently large to hold the expected number of disconnected files if you wish this to happen. Here is how to do this in the Bourne shell for a SAM file system mounted on /sam:

```
/bin/mkdir /sam/lost+found
cd /sam/lost+found
N=0
while [ $N -lt 1024 ]; do
    touch TMPFILE$N
    N=‘expr $N + 1‘
done
rm TMPFILE*
```

OPTIONS

−s *scratch_dir*
: is the scratch directory. If specified, this directory is used for the scratch files that are used. The default scratch directory is /tmp.

−F
: Check and repair the file system. For all inodes that have duplicate blocks, mark those inodes offline if they have been archived.

−G
: Generate directory entry hash. In SAM-FS 3.5.0 and above, a hash code was added to directory entries to speed up directory searches. This is particularly useful for longer file names. The **-G** option, when used in conjunction with the **-F** option, will modify directory entries which do not have a proper hash value to have a hash. When the **-G** option is used without the **-F** option, the number of directory entries which could be hashed is reported. The presence of a hash value has no effect on versions of SAM-FS prior to 3.5.0.

−V
: Turns on a verbose diplay of DEBUG information. This information is useful to Sun Microsystems analysts.

−R
: Rename the file system. When specified along with the -F option, the -R option will rewrite the super block with the disk cache family set name found in /etc/opt/SUNWsamfs/mcf. No action will be taken if the -R option is used without the -F option.

EXIT STATUS

The following exit values are returned:

0	The filesystem is consistent.
4	Nonfatal: Filesystem block counts need to be reconciled.
5	Nonfatal: Filesystem blocks can be reclaimed.

10	Nonfatal: Orphan inodes can be moved to lost+found.
20	Fatal: invalid directory blocks exist, overlapping blocks mapped to 2 inodes exist. Files/directories will be marked offline if an archive copy exists or damaged if no archive copy exists.
30	Fatal: I/O Errors occurred, but samfsck kept processing. Filesystem is not consistent.
35	Fatal: Argument errors terminated samfsck.
36	Fatal: Malloc errors terminated samfsck.
37	Fatal: Device errors terminated samfsck.
40	Fatal: Filesystem superblock is invalid.
45	Fatal: Filesystem .inodes file is invalid.
50	Fatal: I/O Errors terminated samfsck.

FILES

/etc/opt/SUNWsamfs/mcf
The configuration file for **samfs**

SEE ALSO

mcf(4)

NAME

samfsconfig − Recovers configuration information

SYNOPSIS

/opt/SUNWsamfs/sbin/samfsconfig [**−d**] [**−h**] [**−s**] [**−v**] *device* [*device*] ...

AVAILABILITY

SUNWqfs
SUNWsamfs

DESCRIPTION

The **samfsconfig** utility opens the *device*(s) listed on the command line, attempts to read the Sun SAM-FS or Sun SAM-QFS file system superblock on each, and generates output in a format similar to an editable **mcf**(4) file. A *Sun SAM-FS or Sun SAM-QFS file system superblock* is a record that the **sammkfs**(1M) utility writes to the beginning of every device in a Sun SAM-FS or Sun SAM-QFS file system. This record identifies the devices to the file system.

By default, the output is written to **stdout**, but the output can be redirected to a file and edited to regenerate the file system portions of the **mcf** file in the event of a system reconfiguration or disaster.

OPTIONS

This command accepts the following options:

−d	Generates detailed information about all the Sun SAM-FS and Sun SAM-QFS superblocks found, including the content of each superblock.
−h	Generates a usage message and exits.
−s	Print the host file contents of SAM shared filesystems.
−v	Generates messages regarding the disposition of each *device*.
device	One or more *device* identifiers from which configuration information is to be recovered. Use a space character to separate multiple *device* identifiers on the command line. It can be desireable to save a list of *device* identifiers to a file and use this file for command line input to the program.

The **samfsconfig** utility generates information about all the Sun SAM-FS and Sun SAM-QFS file systems and file system components it finds. It flags irregularities as follows:

- It prefixes any incomplete file system components with a pound sign (#) to indicate problems.
- It prefixes any duplicate devices discovered with a greater-than sign (>). This is common if, for instance, multiple paths exist or whole disk partitions are specified on the command line.

EXAMPLES

Example 1.

```
ceres# samfsconfig /dev/dsk/*

#
# Family Set ’samfs2’ Created Thu Jun 29 11:55:45 2000
#
# Missing slices
# Ordinal 3
# /dev/dsk/c3t0d0s5    44    md   samfs2  -
# Ordinal 4
# /dev/dsk/c3t0d0s6    45    md   samfs2  -
# Ordinal 6
# /dev/dsk/c3t1d0s5    47    md   samfs2  -
# Ordinal 7
# /dev/dsk/c3t1d0s6    48    md   samfs2  -
```

Example 2. Another example, this from a saved list of devices:

```
ceres# samfsconfig -v ‘cat /tmp/dev_files‘
Device ’/dev/dsk/c1t0d0s0’ has a SAM-FS superblock.
Device ’/dev/dsk/c1t0d0s1’ has a SAM-FS superblock.
Couldn’t open ’/dev/dsk/c1t0d0s3’;  errno=5.
Device ’/dev/dsk/c1t2d0s0’ has a SAM-FS superblock.
Device ’/dev/dsk/c1t2d0s1’ has a SAM-FS superblock.
Device ’/dev/dsk/c1t4d0s0’ has a SAM-FS superblock.
Device ’/dev/dsk/c1t4d0s1’ has a SAM-FS superblock.
Couldn’t open ’/dev/dsk/c1t4d0s3’;  errno=5.
Couldn’t open ’/dev/dsk/c1t4d0s4’;  errno=5.
Couldn’t open ’/dev/dsk/c1t4d0s5’;  errno=5.
Device ’/dev/dsk/c2t7d0s0’ has a SAM-FS superblock.
Device ’/dev/dsk/c2t7d0s6’ has a SAM-FS superblock.
Device ’/dev/dsk/c3t0d0s3’ has a SAM-FS superblock.
9 SAM-FS devices found.

#
# Family Set ’samfs1’ Created Tue Aug  1 16:57:24 2000
#
# Missing slices
# Ordinal 2
# /dev/dsk/c1t4d0s1    23    mr   samfs1  -

#
# Family Set ’qfs1’ Created Thu Aug 10 19:04:56 2000
#
# Missing slices
# Ordinal 0
# /dev/dsk/c1t4d0s0    11    mm   qfs1  -

#
# Family Set ’samfs1’ Created Wed Feb 21 20:21:04 2001
#
samfs1 ma 10 samfs1
/dev/dsk/c2t7d0s0    11    mm   samfs1  -
/dev/dsk/c2t7d0s6    12    mr   samfs1  -

#
# Family Set ’qfs1’ Created Thu Feb 22 12:49:10 2001
#
qfs1 ma 20 qfs1
/dev/dsk/c3t0d0s3    21    mm   qfs1  -
/dev/dsk/c1t0d0s0    24    g0   qfs1  -
/dev/dsk/c1t0d0s1    25    g0   qfs1  -
/dev/dsk/c1t2d0s0    26    g2   qfs1  -
/dev/dsk/c1t2d0s1    27    g2   qfs1  -
```

SEE ALSO

sammkfs(1M)

mcf(4)

NAME

samfsdump, samfsrestore − Dumps or restores Sun SAM-FS or Sun SAM-QFS file control structure data

SYNOPSIS

samfsdump [**−b** *bl_factor*] [**−d**] **−f** *dump_file* [**−n**] [**−q**] [**−u**] [**−v**] [**−B** *size*] [**−H**] [**−T**] [**−W**] [**−X** *excluded_dir*] [*file* ...]

samfsrestore [**−b** *bl_factor*] [**−d**] **−f** *dump_file* [**−g** *log_file*] [**−i**] [**−l**] [**−s**] [**−t**] [**−v**] [**−B** *size*] [**−R**] [**−T**] [**−2**] [*file* ...]

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **samfsdump** command creates a dump file containing control structure information for each specified *file*. This command must be entered after you have used the **cd**(1) command to change to the mount point of a Sun SAM-FS or Sun SAM-QFS file system.

The **samfsdump** command creates a dump file, as follows:

- If nothing is specified for *file*, the **samfsdump** command creates a dump file containing the control structures for every file in the current directory and also for every file in the current directory's subdirectories.
- If an individual file is specified for *file*, the **samfsdump** command creates a dump file containing the control structures for that individual file.
- If a directory is specified for *file*, the **samfsdump** command creates a dump file containing the control structures for every file in that directory and also for every file in that directory's subdirectories.

Any *file* specified with an absolute path is stored in the dump file with an absolute path. Any *file* specified with a relative path is stored in the dump file with its relative path.

The **samfsrestore** command uses the contents of the dump file to restore control structures for all the files in the dump file or for each specified *file*. If a *file* is specified, its path and file name must match exactly what exists in the dump file. By default, all files are restored to the absolute or relative location as each file is described in the dump file. If the **−s** option is specified, however, all file names with an absolute path in the dump file are restored relative to the current directory, using the entire path as contained in the dump file.

The **samfsdump** command does not create a dump of any data associated with the files, so no data can be restored from this dump file. It is assumed that the data associated with the dumped files has been archived in some way. If a file for which no archive copy is available is dumped, a warning message is issued noting that this file will be marked as damaged when restored. When that file is restored from the dump file, it is marked as damaged by **samfsrestore**. Note that this warning can be explicitly suppressed by using the **−q** option.

You must be logged in as superuser (**root**) in order to execute the **samfsdump** and **samfsrestore** commands. Sun Microsystems recommends that a site create **samfsdump** dumps on a periodic basis as part of a disaster recovery plan.

OPTIONS

This command accepts the following options:

−b *bl_factor*
: Specifies a blocking factor in units of 512 bytes. When specified, all I/O to the dump image file is done in multiples of the blocking factor. There is no blocking done by default.

−d
: Enables debugging messages. This option is useful only to Sun Microsystems and is used to trace execution for verification purposes.

−f *dump_file*
: Names the file to which the control structure data dump is written to (by **samfsdump**) or

read from (by **samfsrestore**). You must specify a *dump_file*.

If a dash character (−) is specified for the *dump_file*, **samfsdump** writes the dump file to **stdout** and **samfsrestore** reads the dump file from **stdin**.

The dump file data can be passed through appropriate filters, such as compression or encryption, after being written by **samfsdump** or before being read by **samfsrestore**.

−g *log_file* Generates a file of online directories and files. For information on the format of this file, see the NOTES section of this man page.

−i Prints the inode numbers of the files when listing the contents of the dump. For more listing options, see **−l**, **−t**, and **−2** options.

−l Prints one line per file. This option is similar to the **sls**(1M) command's **−l** option when listing the dump contents. Note that this option is identified by the lowercase letter 'l', not a number '1'. For more listing options, see the **−i**, **−t**, and **−2** options.

−n Forces the **samfsdump** file to use the newest header format available. The new header is incompatable with **samfsrestore** prior to the 3.5.0 release level. Using the **−u** argument automatically enables the **−n** argument.

−q Suppresses warning messages for damaged files. By default, **samfsdump** writes warning messages for each file that would be considered damaged if the dump were restored.

−s Removes leading slashes from file names prior to restoring them. This is useful if the dump was made with an absolute path name and the dump is being restored to a different location. Any directories required for the restoration and not defined in the dump file are automatically created.

−t Lists the content of the dump file rather than restoring the dump. For more listing options, see the **−i**, **−l**, and **−2** options.

−u Dumps the data portions of files without at least one archive copy. This option can considerably increase the size of the dump file, as data and metadata are both being dumped. You must take care to manage the increased size of the dump.

−v Prints file names as each file is processed. This option is superseded by the **−l** or **−2** options.

−B *size* Specifies a buffer size in units of 512 bytes. Note that there are limits on the buffer size, as specified in the error message when the limits have been exceeded. The default buffer size is 512 ∗ 512 bytes.

−H For **samfsdump**, creates the dump file without a dump header record. For **samfsrestore**, declares that the existing dump file has no header record. This option can be used to create control structure dump files that can be concatenated using the **cat** command. For more information on this command, see the **cat**(1) man page.

−R Replaces existing files when restoring control structures.

−T Displays statistics at command termination. These statistics include the number of files and directories processed, the number of errors and warnings, and other information. Example:

```
CSD statistics:
              Files:              52020
              Directories:        36031
              Symbolic links:     0
              Resource files:     8
              File segments:      0
              File archives:      0
```

```
Damaged files:      0
File warnings:      0
Errors:             0
Unprocessed dirs:   0
```

The numbers after the **Files**, **Directories**, **Symbolic links**, and **Resource files** keywords are the counts of files, directories, symbolic links, and removable-media files whose inodes are contained in the dump.

File segments refers to the number of data segments associated with segmented files from the dump.

File archives refers to the number of archive images associated with the preceding **Files**, **Directories**, **Symbolic links**, and **Resource files**.

Damaged files refers to the number of **Files**, **Directories**, **Symbolic links**, and **Resource files** that that are either already marked damaged (for a **samfsdump**) or were damaged during a restore because they had no archive image (for a **samfsrestore**).

File warnings refers to the number of **Files**, **Directories**, **Symbolic links**, and **Resource files** that would be damaged should the dump be restored because they had no archive images at the time of the dump.

Errors refers to the number of error messages that were printed during the dump or restore. These errors indicate a problem, but the problem is not severe enough to cause an early exit from **samfsdump** or **samfsrestore**. Examples of errors during a restore are failing to create a symbolic link and failing to change the owner or group of a file. Errors that might occur during a dump include having a path name too long, failing to open a directory for reading, failing to read a symbolic link or resource file, or finding a file with an invalid mode.

Unprocessed dirs refers to the number of directories that were not processed due to an error, such as being unable to create the directory.

-W (Obsolete.) Writes warning messages during the dump process for files that would be damaged if the dump were restored. This option is retained for compatibility. By default, these warning messages are now issued automatically. For more information on controlling this behavior, see the **−q** option, which suppresses warning messages.

−X *excluded_dir*
Specifies directory paths to be excluded from the dump. Multiple (up to 10) directories can be excluded by using multiple **−X** options. A directory that resolves to . or **NULL** generates an error message.

−2 Writes two lines per file, similar to the **sls**(1) command's **−2** option, when listing the contents of the dump. For more listing options, see the **−i**, **−l**, and **−t** options.

file ... Lists files to be dumped or restored. Note that the names given to restore must match exactly the names as they are stored in the dump. You can use **samfsrestore −t** to see how the names are stored.

NOTES

A **samfsrestore** should not be attempted on a Sun QFS shared file system client.

The **samfsdump** output files compress to less than 25% of their original size.

If the **−g** option is used, a log file is generated during file system restoration. This file contains one line per file that was online, or partially online, at the time the file was dumped. This line is divided into fields and contains the following information:

Field	Description
1	The file type, which is indicated by one of the following letters: • **d** indicates a directory. • **f** indicates a regular file. • **l** indiactes a symbolic link. • **R** indicates a removable media file. • **I** indicates a segment index. • **S** indicates a data segment.
2	The media type and Volume Serial Name (VSN) in *media_type***.***vsn* format.
3	The position on the media.
4	Either **online** or **partial**.
5	The path relative to the file system mount point.

After a **samfsrestore** command is issued, it is possible to restore files that were online, prior to the dump, back to their online state. You do this by using the script in **/opt/SUNWsamfs/examples/restore.sh.**

EXAMPLES

The following example creates a control structure dump of the entire **/sam** file system:

```
example# cd /sam
example# samfsdump -f /destination/of/the/dump/samfsdump.today
```

To restore a control structure dump to **/sam**:

```
example# cd /sam
example# samfsrestore -f /source/of/the/dump/samfsdump.yesterday
```

SEE ALSO

cat(1), **sls**(1).

DIAGNOSTICS

You may encounter messages while using the **samfsdump** or **samfsrestore** command. The following list shows several possible messages and their explanations:

Message	Explanation
*file***: Unrecognised mode (0x..)**	**samfsdump** is being asked to dump a file that is not a regular file, directory, symbolic link, or removable media file. The Sun SAM-FS and Sun SAM-QFS file systems allow the creation of block special, character special, fifo, and other special files, but they do not function correctly. **samfsdump** does not attempt to dump them.
*file***: Warning! File will be damaged.**	If received during a **samfsdump**, this means that the file in question does not currently have any archive copies. The file is dumped to the **samfsdump** file, but if the **samfsdump** file is used to restore this file, the file will be marked damaged.
*file***: Warning! File is already damaged.**	If received during a **samfsdump**, means that the file is currently marked damaged. During restoration, the file will still be damaged.
*file***: File was already damaged prior to dump**	If received during a **samfsrestore**, this means that the file was dumped with the **damaged** flag set.

file: **File is now damaged**
If received during a **samfsrestore**, this means that the file was dumped when it had no archive images. **samfsdump** and **samfsrestore** do not dump file data. They rely on the file's data having been archived. Because the file no longer has any data associated with it, it is marked **damaged**.

.: Not a SAM-FS file. You are attempting to dump files from a file system that is not a Sun SAM-FS or Sun SAM-QFS file system, or you are attempting to restore files from a **samfsdump** dump file into a file system that is not a Sun SAM-FS or Sun SAM-QFS file system.

file: **stat() id mismatch: expected: %d.%d, got %d.%d**
If received during a dump, this indicates one of two things. If the **%d.** portions match, but the **.%d** portions differ, then a directory or file was deleted and recreated while **samfsdump** was operating on it. The file is not dumped. If the **%d.** portions do not match, then a serious error has been encountered; consult your service provider for help.

Corrupt samfsdump file. name length %d
If received during a restore, this means that the path name of a file to be restored was less than zero or larger than **MAXPATHLEN**. This should not occur. **samfsrestore** aborts.

Corrupt samfsdump file. %s inode version incorrect
During a restore, this means that a the inode for the indicated file was in an old format. This should not occur. **samfsrestore** aborts.

file: **pathname too long**
If received during a dump, this indicates that the path name of the indicated file is longer than 1024 characters. The file is not dumped.

NAME

samgrowfs − Adds disk partitions to an existing Sun SAM-FS or Sun SAM-QFS file system

SYNOPSIS

samgrowfs [**−V**] *fsname*

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **samgrowfs** command adds disk partitions to an Sun SAM-FS and Sun SAM-QFS file system and allows the file system to grow.

The following procedure uses the **samgrowfs** command to increase the size of a Sun SAM-FS or Sun SAM-QFS file system:

1. Unmount all the file systems you want to grow.
2. In a Sun SAM-FS or Sun SAM-QFS environment, idle all drives by entering a **samcmd idle** *eq* and a **samd stop** command. For more information on these commands, see the **samcmd**(1M) and **samd**(1M) man pages.
3. Edit the **mcf** file, save the changes, and quit the editor. Up to 252 disk partitions can be specified in the **mcf** file for a Sun SAM-FS or Sun SAM-QFS file system. The new partitions must be placed after the existing partitions for the specified family set *fsname*.
4. Run the **samgrowfs**(1M) command on the *fsname* file system.
5. Mount the *fsname* file system.

For more information on this procedure, see the *Sun QFS, SAM-FS, and SAM-QFS File System Administrator's Guide*.

OPTIONS

This command accepts the following arguments:

−V	Lists configuration information but does not execute the command.
fsname	Specifies the existing family set name of the file system that is to grow. This is the family set name as specified in the **mcf** file.

EXAMPLE

The following example adds 2 partitions to an existing 1-partition SAM-FS file system. The **mcf** file for the existing 1-partition file system with a family set name of **samfs1** is as follows:

```
samfs1  10  ms  samfs1
/dev/dsk/c0t3d0s7  11  md  samfs1  -  /dev/rdsk/c0t3d0s7
```

The procedure is as follows:

1. Unmount the **samfs1** file system.

   ```
   server# umount samfs1
   ```

2. Kill the **sam-initd** process:

   ```
   server# samd stop
   ```

3. Edit the **mcf** file and add the 2 new partitions for the file system with family set name of **samfs1**:

   ```
   samfs1  10  ms  samfs1
   /dev/dsk/c0t3d0s7  11  md  samfs1  -  /dev/rdsk/c0t3d0s7
   /dev/dsk/c2t3d0s2  12  md  samfs1  -  /dev/rdsk/c2t3d0s2
   /dev/dsk/c2t4d0s2  13  md  samfs1  -  /dev/rdsk/c2t4d0s2
   ```

4. Grow and mount the file system by entering the following commands:

```
server# samgrowfs samfs1
server# mount samfs1
```

FILES

/etc/opt/SUNWsamfs/mcf The configuration file for Sun SAM-FS and Sun SAM-QFS file systems.

SEE ALSO

samcmd(1M), **samd**(1M), **sammkfs**(1M).

mcf(4).

See the Sun QFS, SAM-FS, and SAM-QFS File System Administrator's Guide.

WARNINGS

Be sure that the **/dev/dsk** and **/dev/rdsk** names for each **md** device reference the same **c***n***t***n***d***n***s***n* partition.

As with creating any type of file system, if you specify the wrong partition names, you risk damaging user or system data. Be sure to specify partitions which are otherwise unused on your system. Do not use overlapping partitions.

To grow a QFS file system, you must add a metadata partition (**mm**) prior to issuing a **samgrowfs** command. Data partitions can be added as well as metadata partitions. The added metadata partition contains block reservation information for all added partitions. When adding a small metadata partition with large data partitions, the small metadata partition may be too small to hold the block reservation as well as other information, depending on total storage added and DAU size. This condition may cause an error, or a very full metadata partition after samgrowfs.

NAME

sammkfs, samfsinfo − Constructs or displays information for a Sun QFS, Sun SAM-FS, or Sun SAM-QFS file system

SYNOPSIS

/opt/SUNWsamfs/sbin/sammkfs [**−a** *allocation_unit*] [**−i** *inodes*] [**−P**] [**−S**] [**−V**] *fs_name*

/opt/SUNWsamfs/sbin/samfsinfo *fs_name*

AVAILABILITY

SUNWqfs

SUNWsamfs

DESCRIPTION

The **sammkfs** command creates a Sun QFS, Sun SAM-FS, or Sun SAM-QFS file system from the disk partitions that belong to the family set *fs_name*, where *fs_name* is the family set name as defined in the **mcf** file. Up to 252 disk partitions can be specified in the **mcf** file for a Sun QFS, Sun SAM-FS, or Sun SAM-QFS file system. The **sammkfs** command can also be used to recreate a file system after a disaster.

The **samfsinfo** command displays the structure of an existing Sun QFS, Sun SAM-FS, or Sun SAM-QFS file system. The output is similar to that obtained by using the **−V** option to the **sammkfs** command.

OPTIONS

These commands accept the following options:

−a *allocation_unit*

Specifies the disk allocation unit (DAU). The DAU is the basic unit of online storage. When you specify a DAU size, you specify the number of 1024-byte (1 kilobyte) blocks to be allocated for a file.

The DAU size you can specify depends on the type of file system being initialized, as follows:

- The Sun SAM-FS file system is an **ms** file system. The disk devices in it are all **md** devices. Both data and metadata are written to the **md** devices. The *allocation_unit* specifies the DAU to be used for the **md** devices. Possible *allocation_unit* specifications are **16** (the default), **32**, or **64**.
- The Sun QFS and Sun SAM-QFS file systems are **ma** file systems. The metadata in these file systems is written to **mm** devices. The disk devices in these file systems are specified as either **md**, **mr**, or **g***XXX* devices, as follows:
  - For the **md** devices, possible *allocation_unit* specifications are **16**, **32**, or **64** (the default). A single file system cannot have **md** devices mixed among the **mr** and **g***XXX* devices.
  - For **mr** devices, the DAU is fully adjustable. Specify an *allocation_unit* that is a multiple of 8 in the following range for **mr** devices: 16 ≤ *allocation_unit* ≤ 65536. The default is **64**.
  - For **g***XXX* devices, which specify striped groups, the DAU is fully adjustable. If the file system contains striped groups, the minimum unit of disk space allocated is the DAU multiplied by the number of members in the striped group. Specify an *allocation_unit* that is a multiple of 8 in the following range for **g***XXX* devices: 16 ≤ *allocation_unit* ≤ 65536. The default is **256**.

You can mix **mr** and **g***XXX* devices in a single Sun QFS or Sun SAM-QFS file system. If these device types are mixed, the *allocation_unit* specified is used for both device types. If no *allocation_unit* is specified, the DAU size used for each type of device is **256**.

−i *inodes* Specifies the number of inodes to be allocated for this file system. This is the total number of user inodes that can be used for the life of this file system. In Sun QFS, Sun SAM-FS, and Sun SAM-QFS version 2 superblock file systems, a number of inodes are reserved for file system usage, and are unavailable to the user. This number is in addition to the specified number of user inodes. The actual number of inodes available vary from that specified, due to rounding to metadata DAU size.

NOTE: By specifying this option, you eliminate the possibility of ever increasing the number of inodes for the file system. Therefore, Sun does not recommend the use of this option.

When this option is specified, later use of the **samgrowfs**(1M) command increases the size of the file system, but it cannot increase the number of allowable inodes. For more information on enlarging file systems, see the WARNINGS section of this man page and the **samgrowfs**(1M) man page.

−P Specifies that a previous version of the **sammkfs** command be invoked, which creates a file system compatable with that version. The current previous version creates a version 1 superblock, which was created for Sun QFS, Sun SAM-FS, and Sun SAM-QFS releases prior to 4.0. Without the **−P** parameter, a version 2 superblock is created, and it is unmountable using software at a release level prior to 4.0.

This option cannot be specified in conjunction with the **−S** option.

−S Indicates that this file system is shared. In order to mount the file system as a Sun QFS shared file system, you must also create a **hosts.***fs_name* configuration file. For more information on this configuration file and other aspects of the Sun QFS shared file system, see the *Sun QFS, Sun SAM-FS, and Sun SAM-QFS File System Administrator's Guide*. For information on configuring a hosts file, see the **hosts.fs**(4) man page.

This option cannot be specified in conjunction with the **−P** option.

−V Writes configuration information to standard output but does not execute the **sammkfs** command. This information can be used to create a new file system.

The **samfsinfo** command should be used to generate configuration information for an existing file system.

EXAMPLES

Example 1. The following command creates a Sun SAM-QFS file system with a DAU size of 128 kilobytes:

```
server#  sammkfs -a 128 samfs1
```

FILES

/etc/opt/SUNWsamfs/mcf The configuration file for a Sun QFS, Sun SAM-FS, or Sun SAM-QFS file system.

WARNINGS

Be sure that the **/dev/dsk** and **/dev/rdsk** names for each **md** device reference the same **c***n***t***n***d***n***s***n* partition. As with creating any type of file system, if you specify the wrong partition names, you risk damaging user or system data. Be sure to specify partitions that are otherwise unused on your system. Do not use overlapping partitions.

SEE ALSO

samgrowfs(1M), **undamage**(1M).

mcf(4).

Sun QFS, Sun SAM-FS, and Sun SAM-QFS File System Administrator's Guide.

Sun SAM-FS and Sun SAM-QFS Storage and Archive Management Guide.

WARNINGS

Be careful when using the **−i** *inodes* option for this command. By using this option, you dictate the maximum number of inodes allowed for the life of this file system. This eliminates the possibility of ever using the **samgrowfs**(1M) command to increase the number of files in this file system. After a file system is made with **−i** specified, the **samgrowfs**(1M) command can only be used to increase the size of the file system in terms of bytes.

NOTES

Data alignment refers to matching the allocation unit of the RAID controller with the *allocation_unit* of the file system. A mismatched alignment causes a read-modify-write operation for I/O that is less than the block size. The optimal alignment formula is as follows:

allocation_unit = RAID_stripe_width ∗ *number_of_data_disks*

For example, if a RAID-5 unit has a total of 8 disks with 1 of the 8 being the parity disk, the number of data disks is 7. If the RAID stripe width is 64 kilobytes, then the optimal *allocation_unit* is 64 ∗ 7 = 448.

NAME

samncheck − generate pathnames vs. i-numbers for Sun SAM-FS and Sun SAM-QFS file systems

SYNOPSIS

samncheck *mount_point i-number* [*i-number* ...]

AVAILABILITY

SUNWsamfs

DESCRIPTION

samncheck generates a pathname in the Sun SAM-FS or Sun SAM-QFS filesystem mounted on *mount_point* for each *i-number* listed in the command line. **samncheck** must be run with root permissions.

The output from **samncheck** is one line per *i_number* which represents an existing inode in the file system. The *i_number* followed by the current generation number for that inode is displayed, followed by a tab and a pathname. Note that there may be many pathnames to a given *i_number;* **samncheck** reports just one.

Nonexistant *i_numbers* are silently ignored.

EXAMPLES

```
bilbo# samncheck /sam 1 2 3 4 5 18
1.1     /sam/.inodes
2.2     /sam/
4.4     /sam/.ioctl
5.5     /sam/.archive
18.3    /sam/file
```

SEE ALSO

ncheck(1)

NAME

samquota − Reports, sets, or resets quota information

SYNOPSIS

samquota [−**a**] [−**b** *count***:***type*] [−**e**] [−**f** *count***:***type*] [−**g**] [−**h**] [−**i**] [−**k**] [−**p**] [−**t** *interval*] [−**u**] [−**w**] [−**x** *action*] [−**z**] [−**A** *adminsetID*] [−**G** *groupID*] [−**U** *userID*] [*file*]

AVAILABILITY

SUNWqfs

DESCRIPTION

The **samquota** command displays quota usage statistics and can be used to edit quotas, grace periods, and usages for users, groups, and admin sets. This command supports file counts and online block counts.

Only a superuser can use this command to change quotas. End users can use a subset of this command's options to display quota usage and to display limit information. For more information on the end-user version of this command, see the **squota**(1) man page.

By default, **samquota**(1M) writes the user's applicable GID/UID quotas and usages on all mounted Sun QFS file systems to **stdout**.

ADMIN SETS AND DIRECTORY/PROJECT QUOTAS

An admin set quota applies to all files and directories on a file system that have their admin set attribute set to the given value. The main use of admin set quotas is to effect directory or project quotas. They can be used to effect directory quotas by setting a directory's admin set ID to a unique value and using **samquota**(1M) to set quotas for that value. All subdirectories and files subsequently created beneath the directory then inherit the value, and the admin set's quota limits apply to them. Conversely, a project quota can be effected by choosing a set of project directories, setting their admin set ID values to a single unique value, and using **samquota**(1M) to set quotas for that ID. Note in either case that newly created files inherit an admin set ID from the directory in which they are created; the admin set IDs do not change if the file is moved to a new directory with a different admin set ID.

You can use the **samchaid**(1M) command to set admin set IDs. The **samchaid**(1M) command allows system administrators to assign files and directories to individual admin sets. Admin set IDs are not tied to any set of permissions associated with the user. That is, a user can have a set of directories and files on one Sun QFS file system with a particular admin set ID, and the same user can have another set of directories and files on another file system (or even the same one) with a completely different admin set ID. A writable file is therefore used as a surrogate to determine that a user has permission to view an admin set's quota values.

OPTIONS

This command accepts the following options:

−**a** Specifies admin set quota statistics for *file*.

−**b** *count***:***type*

Sets soft, hard, or in-use block allocation limits. This setting can pertain to either online files or to the total number of files. Note that a colon (**:**) is used to separate each component.

count specifies the number of blocks for the limit and must be an integer number in the following range:
$0 \leq$ *count* $\leq$ (2∗∗63) -1.

By default, the *count* specification indicates a number of 512-byte blocks. If the **-k** option is also specified, the *count* specification is interpreted as a number of 1024-byte blocks.

By default, the integer specified for *count* is interpreted as it is written. You can append a unit multiplier to the *count* value, however, to force the system to interpret *count* as a larger number. These unit multipliers are as follows:

Multiplier	**Interpretation**
k or **K**	Specifies 1000. For example, specifying **2k** is interpreted as 2000.
m or **M**	Specifies 1,000,000. For example, specifying **80M** is interpreted as 80,000,000.
g or **G**	Specifies 1,000,000,000.
t or **T**	Specifies 10∗∗12.
p or **P**	Specifies 10∗∗15.

type specifies the type of limit. Possible *type* specifications are as follows:

type	**Interpretation**
s or **soft**	Specifies that the **samquota** command is being used to reset a soft limit.
h or **hard**	Specifies that the **samquota** command is being used to reset a hard limit,
u or **use**	Specifies that the **samquota** command is being used to reset the in-use counter. Typically, this is set only by the **samfsck**(1M) command and other system administration tools.

Example. The following command line sets a soft limit of 120,000 512-byte blocks to be occupied by user **george**'s files in file system **qfs22**:

samquota -b 120k:s -U george /qfs22

−e Writes the quota information from this command line in an executable format. You can use this option if you want the system to put the information from this command into a file for editing.

```
server# samquota -eG sam /qfs1
# Type  ID
#                    Limits
#            soft            hard
# Files
# Blocks
# Grace Periods
#
samquota -G 101 \
      -f     1000:s -f     1200:h \
      -b   100000:s -b   120000:h \
                 -t  1d   /qfs1
```

−f *count***:***type*

Sets soft, hard, or in-use file limits for a file system. Note that a colon (**:**) is used to separate each component.

count specifies the number of files for the limit and must be an integer number in the following range:
0 ≤ *count* ≤ (2∗∗63) -1.

If the **-k** option is also specified, any *count* specification referring to blocks is interpreted in 1024-byte blocks instead of 512-byte blocks (by multiplying by 2).

By default, the integer specified for *count* is interpreted as it is written. You can append a unit multiplier to the *count* value, however, to force the system to interpret *count* as a larger number. These unit multipliers are as follows:

Multiplier **Interpretation**

k or **K**	Specifies 1000. For example, specifying **2k** is interpreted as 2000.
m or **M**	Specifies 1,000,000. For example, specifying **80M** is interpreted as 80,000,000.
g or **G**	Specifies 1,000,000,000.
t or **T**	Specifies 10**12.
p or **P**	Specifies 10**15.

type specifies the type of limit. Possible *type* specifications are as follows:

type	**Interpretation**
s or **soft**	Specifies that the **samquota** command is being used to reset a soft limit.
h or **hard**	Specifies that the **samquota** command is being used to reset a hard limit,
u or **use**	Specifies that the **samquota** command is being used to reset the in-use counter. Typically, this is set only by the **samfsck**(1M) command and other system administration tools.

Example. The following command line sets a soft limit of 120 files for user **martha** in file system **qfs222**:

samquota -U martha -b 120:s /qfs222

−g Returns group quota statistics for *file*.

−h Provides a brief usage summary.

−i Zeros all limits. This option reinitializes the quota specifications by clearing all fields in the quota records except the in-use fields. It then resets the fields to conform to the new specifications on the command line.

−k Specifies that the command interpret or display all storage units (block quantities) in units of 1024-byte blocks. When specified, all information on the command line is assumed to be in units of 1024 bytes, and all information is returned in multiples of 1024 bytes.

Example 1. The following command line specifies a hard quota limit of 256,000 1024-byte blocks (or, equivalently, 512,000 512-byte blocks) for group **adm**, in file system **qfs4**:

samquota -G adm -k -b 256k:hard /qfs4

Example 2. The following command line sets a soft limit of 120 1024-byte blocks (or, equivalently, 240 512-byte blocks) to be occupied by the files for user **fred** in file system **qfs2**:

samquota -U fred -k -b 120:soft /qfs2

−p Writes updated quota statistics to **stdout** if you are changing preestablished quota values or limits.

−t *interval* Specifies the time to be used for the soft limit grace period.

interval specifies the interval to use for the grace period. By default, the integer specified for *interval* is interpreted in units of seconds. You can append a unit multiplier to the *interval* value, however, to force the system to interpret *interval* as a larger unit. These unit multipliers are as follows:

Multiplier	**Interpretation**
w	Specifies weeks. For example, specifying **10w** is interpreted as ten weeks.
d	Specifies days.
h	Specifies hours.
m	Specifies minutes.

s (default)	Specifies seconds.

The *interval* must be an integer number in the following range:
0 ≤ *interval* ≤ (2∗∗31) - 1.

Note that (2∗∗31) - 1 = 2,147,483,647, which means that the maximum specification, in seconds, would be 2147483647, which is about 68 years.

Example. The following command line specifies an interval of 7 days and 12 hours for the grace period of user **adele** in the **myqfs** file system:

samquota -U adele -t 7d12h /myqfs

−u Returns user quota statistics for the owner of *file*.

−w Suppresses messages. By default, **samquota** generates warning messages and requests confirmation before changing any quota values maintained by the system. When this option is specified on the command line in conjunction with the **-b**, **-f**, **-x**, or **-Z** options, it suppresses both the warning messages and the confirmation requests.

−x *action* Adjusts the soft limit grace period timer. After a user reaches a soft limit, a certain amount of time can elapse before a user is not allowed to create any more files in the file system. This option allows you to override the existing quota mechanism and temporarily respecify the consequences of having reached the soft limit.

action specifies what to do with the grace period timer. Note that the soft limit grace period is set with the **-t** option. Possible *action* specifications are as follows:

action	**Interpretation**
clear	Specifies that the current grace period be ended and the grace period counter be reset to zero. The grace period counter is restarted the next time a file or block is allocated.
reset	Specifies that the current grace period be ended and that the grace period counter be restarted immediately.
expire	Specifies that the current grace period be ended and that no new files or blocks be allocated until the user, group, or admin set frees blocks and/or files and is again under the soft limit.
interval	*interval* specifies the interval to use for the grace period. Specifying an *interval* sets the grace period to expire at a new time. The *interval* must be an integer number in the following range: 0 ≤ *interval* ≤ (2∗∗31) - 1.

Note that (2∗∗31) - 1 = 2,147,483,647, which means that the maximum specification, in seconds, would be 2147483647, which is about 68 years.

The timer is set to the given value, and starts counting immediately. If the quota goes under the soft limit, it will be reset to zero at that time.

By default, the integer specified for *interval* is interpreted in units of seconds. You can append a unit multiplier to the *interval* value, however, to force the system to interpret *interval* as a larger unit, and can concatenate these units. These unit multipliers are as follows:

Multiplier	**Interpretation**
w	Specifies weeks (times 7∗24∗60∗60). For example, specifying **10w** is interpreted as ten weeks or 10∗7∗24∗60∗60 seconds.
d	Specifies days (times 24∗60∗60).

h	Specifies hours (times 60∗60).
m	Specifies minutes (times 60).
s (default)	Specifies seconds.

Example. Admin set **pubs** is over its soft limit on file system **qfs50**, and its grace period has expired. You can reset the grace period by using the following command:

samquota -x 1d2h -A pubs /qfs50

If the preceding command is executed at 1100 on Thursday, the grace period for **pubs** is reset to expire at 1300 on Friday.

−z Reinitializes the quota specifications by clearing all fields in the quota records, including the in-use fields. This option also resets the fields to conform to the new specifications on the command line.

−A *adminsetID*
Generates a quota report for an admin set, or, when specified in conjunction with options that reset values, resets the values for the admin set specified. Specify an integer for the *adminsetID*.

−G *groupID*
Generates a quota report for a group, or when specified in conjunction with options that reset values, resets the values for the group specified. Specify an integer identifier or a group name for the *groupID*.

−U *userID* Generates a quota report for a user, or, when specified in conjunction with options that reset values, resets the values for the user specified. Specify an integer identifier or a user name for the *userID*.

file Specifies that the quota information pertain to a specific file. A user is allowed to examine the group, user, or admin set quotas of any file for which the user has write permissions. The information displayed differs depending on whether or not the command is issued by a user who has write permission to *file*, as follows:

- If the user issuing this command has write permission to *file*, the command generates information on the applicable admin set, group, and user quotas that apply to *file*.
- If the user issuing this command does not have write permission to *file*, the command generates information for only the user’s user ID and group ID quotas for the file system on which *file* resides.

EXAMPLES

Example 1. The following command initializes a quota for group **sam** on the file system mounted on **/qfs1**:

```
server# samquota -G sam -f 1000:s -f 1200:h -b 100k:s -b 120k:h -t 1d /qfs1
```

The group is given the following:

- Soft limits of 1000 files and 100,000 512-byte blocks (about 50 megabytes)
- Hard limits of 1200 files and 120,000 512-byte blocks
- A grace period of 1 day (24 hours)

Example 2. The following example initializes a quota for admin set 17 on the file system that **/qfs1/sol** is part of:

```
server# samquota -A 17 -k -f 10k:s -f 20k:h -b 10m:s -b 15m:h -t 1w /qfs1/sol
```

The admin set is given the following:

- Soft limits of 10,000 files and 10,000,000 1024-byte blocks (10.24 gigabytes)
- Hard limits of 20,000 files and 15,000,000 1024-byte blocks (15.36 gigabytes)
- A grace period of 1 week (168 hours)

EXIT STATUS

This command returns the following:

- 0 on successful completion.
- 1 on a usage or argument error.
- 10 on an execution error.

FILES

filesytem/**.quota_a**	Admin set quota information
filesystem/**.quota_g**	Group quota information
filesystem/**.quota_u**	User quota information

SEE ALSO

squota(1)

samfsck(1M)

passwd(4) - User ID information

group(4) - Group ID information

DIAGNOSTICS

No user quota entry.
User quotas are not active on the file system.

No group quota entry.
Group quotas are not active on the file system.

No admin quota entry.
Admin set quotas are not active on the file system.

NOTES

File system quotas are supported on Sun QFS file systems only.

NAME

samquotastat − Reports on active and inactive file system quotas

SYNOPSIS

samquotastat [−**a**] [−**g**] [−**h**] [−**u**] *file*

AVAILABILITY

SUNWqfs

DESCRIPTION

The **samquotastat** command reports whether user, group, or admin set quotas are enabled on the file system that contains *file*. If only the *file* argument is specified, output is generated as if the **-a**, **-g**, and **-u** arguments had all been specified. This command accepts the following arguments:

−**a**	Generates information on admin set quotas.
−**g**	Generates information on group quotas.
−**h**	Generates a brief usage summary.
−**u**	Generates information on user quotas.
file	Specify either a specific file name, a path to a file, or the file system mount point. If a file name or path to a file is specified, the command generates the report for the file system in which the file resides.

EXAMPLES

```
server% samquotastat /qfs1
admin quota enabled
group quota enabled
user quota disabled
```

EXIT STATUS

This command exits with a status of zero if any queried quota types are enabled.

SEE ALSO

squota(1).

samquota(1M), **samfsck**(1M).

NOTES

File system quotas are supported on Sun QFS file systems only.

NAME

samset − Change the Sun SAM-FS or Sun SAM-QFS environment

SYNOPSIS

samset *[keyword [parameter...]]*

AVAILABILITY

SUNWsamfs

DESCRIPTION

samset is used to change or display variables that control Sun SAM-FS or Sun SAM-QFS operation. Without any arguments, **samset** displays current settings to stdout. If **samset** is executed with a *keyword* but with no *parameter...*, then the current value for just that *keyword* is displayed to stdout.

The *keywords* all have values assigned to them at startup. These values come from the **defaults.conf** file. **samset** allows you to change *keywords* while sam-fsd is running. Any changes made remain effective only during the current instance of sam-fsd; values revert to the defaults in **defaults.conf** at the next startup.

The following *keywords* are supported:

attended yes

attended no

attended tells Sun SAM-FS or Sun SAM-QFS library daemon if an operator is available to manually mount media. Regardless of the **attended** setting, requests for media which are mounted in a drive, or present in a media changer, will be satisfied as soon as possible. **attended** affects the behavior of Sun SAM-FS or Sun SAM-QFS library daemon when a medium is requested which is not currently present in either a manually-mounted drive, or in a library. The usual action taken by the library daemon when such a request occurs is to place it into the preview display (see **previewtool** (1M)), and await manual intervention (but see **stale_time**, below). However, if either **attended** is set to **no**, or the medium is marked "unavailable" in the historian catalog, then the request will not go into the preview display, and will fail with an ESRCH error. If other archive copies are available, they will be tried. If no further copies are available, ENXIO will be returned to the requester.

exported_media +u *eq...*

exported_media -u *eq...*

This option controls the flagging of media exported (see **export**(1M)) from the listed libraries as unavailable (**+u**) or available (**-u**) in the historian's catalog. See **attended**, above, for the effect of this flag. The setting of the flag for a given medium may be changed after export using **chmed**.

idle_unload

This is the time (in seconds) that a media changer controlled device may be idle before the media in that device is unloaded. A value of zero will disable this feature.

labels *label-option*

This option applies only to barcode-reader equipped tape libraries.

The media daemon can obtain the tape label from the upper-cased characters of the tape's barcode. *label-option* may be: **barcodes**, to use the first six characters of the barcode as label; **barcodes_low**, to use the trailing six characters; or **read**, to disable barcode processing and to read the magnetic label from the tape.

When **labels** is set to **barcodes** or **barcodes_low**, any tape robotically mounted for a write operation that is write enabled, unlabeled, has never been mounted before, and has a readable barcode will have a magnetic label written before the write is started.

stale_time *minutes*

Sets the amount of time (in minutes) that a request for media will wait in the preview table before being canceled with an ETIME. The file system will react to an ETIME error in the same way as an ESRCH error (see **attended**, above).

timeout *seconds*
Sets the time (in seconds) that will be allowed to elapse between I/O requests for direct access to removable media (see **request**(1)). If a process fails to issue the next I/O to the device within this time, the device will be closed and, on the next I/O, the process will receive an ETIME error. A value of 0 implies no timeout will occur.

debug debug manipulates the debug/trace flags within Sun SAM-FS or Sun SAM-QFS environents to produce expanded logging. Unless otherwise specified, the debug messages are logged to the syslog facility at the LOG_DEBUG priority. *parameter...* is a space separated list of flags. To set a flag, give its name. To clear a flag, give its name prefixed with a '-'. The flags are:

all Turn on all debug flags (except trace_scsi and robot_delay).

none Turn off all debug flags.

default Set all debug flags to the default as defined by **defaults.conf**.

logging File system requests to the daemons and the daemons response to the requests are logged to files. These files are used only by Sun Microsystems support.

debug This is catch-all for messages that might be of interest but generally do not show a problem.

moves Log move-media commands issued to media changers.

events This should only be used by Sun Microsystems analysts to trace the flow of events used by the media changer daemons. These messages are coded and of little use in the field. These messages are logged to syslog at LOG_NOTICE priority.

timing This setting has been replaced by the device log timing event **devlog** *eq [event ...]*. This is described in more detail under the **devlog** keyword.

od_range For optical disk media, log the range of sectors allowed for writing.

labeling Log the VSN, blocksize (for tape media only), and label date when a label is read from a medium following the media's being mounted. These messages are logged to syslog at LOG_INFO priority.

canceled Log when the stage process detects a canceled stage request.

disp_scsi Display the current SCSI cdb being executed by a device. This information is appended to any existing message. If the length of the existing message and the cdb would overflow the message area, the cdb is not displayed. The message area for a device can be viewed with samu (see **samu**(1M)) in the "s" or "r" displays.

messages This is used by Sun Microsystems analysts to trace the flow of messages used by the media changer daemons. These messages are coded and of little use to customers. These messages are logged to syslog at LOG_NOTICE priority.

migkit Log events connected with the Sun Sam Migration Toolkit.

mounts Log media mount requests.

opens Log open and close of removable media devices.

remote Log events connected with the Sun Sam Remote daemon.

trace_scsi This option may only be set by the super user through the samset command. It causes all scsi commands issued through the user_scsi interface to be written to a file named /tmp/sam_scsi_trace_*xx* (where *xx* is the equipment number of either the media changer to which this device belongs or the device itself if it does not belong to a media changer.) The trace file is opened with O_APPEND and O_CREAT on the next I/O to each device after this flag is set. It is closed when the option is cleared and the next I/O to that device occurs. Sun Microsystems does not recommend running with this option for long periods. The format of the trace information is:

```
struct {
  int    eq;      /* equipment number */
  int    what;    /* 0 - issue, 1 - response */
  time_t now;     /* unix time */
  int    fd;      /* the fd the ioctl was issued on */
  char   cdb[12]; /* the cdb */
  char   sense[20]; /* returned sense(valid if what=1) */
}cdb_trace;
```

Sun Microsystems does not recommend setting this option indiscriminately, as large output files are quickly produced.

stageall This should be used only by Sun Microsystems analysts to trace stageall processing.

devlog *eq [event ...]*
devlog manipulates the device log event flags for device *eq*. *eq* is either an equipment ordinal or "all"; if "all", then the flags are set or listed for all devices. These flags control which events get written to the device log files. *[event ...]* is a space separated list of event names. To set an event flag, give its name. To clear a flag, give its name prefixed with a '-'. The events are:

all	Turn on all events.
none	Turn off all events.
default	Set the event flags to the default which are: err, retry, syserr, and date.
detail	events which may be used to track the progress of operations.
err	Error messages.
label	Labeling operations.
mig	Migration toolkit messages.
msg	Thread/process communication.
retry	Device operation retries.
syserr	System library errors.
time	Time device operations.
module	Include module name and source line in messages.
event	Include the event name in the message.
date	Include the date in the message.

SEE ALSO
request(1), **chmed**(1M), **export**(1M), **samu**(1M), **defaults.conf**(4), **mcf**(4).

NAME

samsharefs − Manipulates the Sun QFS shared file system configuration information

SYNOPSIS

samsharefs [**−h**] [**−q**] [**−R**] [**−s** *host*] [**−u**] *fs_name*

AVAILABILITY

SUNWsamfs
SUNWqfs

DESCRIPTION

The **samsharefs** command prints and modifies the host configuration for a Sun QFS shared file system. The printed hosts file identifies the metadata server and the client hosts included in the Sun QFS shared file system configuration. This command is only valid from the metadata server or potential metatdata server.

Initially, you create the hosts file using **vi**(1) or another text editor. The **sammkfs**(1M) command reads the hosts file when the Sun SAM-QFS shared file system is created, and it initializes the file system host configuration.

To subsequently change the host configuration you must use the **samsharefs** command. Typically, you use an editor to edit an ASCII hosts file as printed by the **samsharefs** command and use the **samsharefs** command to update the file system host configuration.

OPTIONS

This command accepts the following options:

−h Writes a short usage message to **stdout**.

−q Suppresses host configuration output. By default, the command writes the file system host configuration, possibly modified, to **stdout**.

−R Specifies that the file system's host configuration should be manipulated using the raw disk device associated with the file system, rather than the file system interfaces. This option can be used to change hosts information when the file system is not or cannot be mounted. This option can also be used to change hosts information when the filesystem is mounted, but the active metadata server is down.

CAUTION: This option **must not** be executed on a potential metadata server to change the metadata server host wihout first stopping, disabling, or disconnecting the active metadata server. Doing so will cause file system corruption.

−s *host* Sets the **server** flag for the specified host in the system configuration. This option declares *host* to be the new metadata server host. All other hosts' **server** flags are cleared.

If this option is used in conjunction with the **−u** option, file system host configuration is generated from **/etc/opt/SUNWsamfs/hosts.***fs_name*. the server is set to the specified host; and the file system's host configuration is updated.

−u Specifies that the hosts file is to be updated from **/etc/opt/SUNWsamfs/hosts.***fs_name*.

fs_name Specifies the family set name of the Sun QFS shared file system.

EXAMPLES

Example 1. The following example shows how to use the **samsharefs** to examine the hosts information on a mounted Sun QFS shared file system:

```
tethys# samsharefs share1
#
# Host file for family set 'share1'
#
# Version: 2    Generation: 14    Count: 3
# Server = host 0/titan, length = 112
#
titan titan.xyzco.com 1 0
tethys tethys.xyzco.com 2 0
mimas mimas.xyzco.com 0 0
```

Example 2. The following example shows how the hosts file can be modified to add a new server to the shared filesystem. The administrator has edited /etc/opt/SUNWsamfs/hosts.share1 and added new hosts for the shared filesystem as shown. **samsharefs** is then run with the **-u** option to update the (mounted) filesystem's configuration.

```
titan# cat /etc/opt/SUNWsamfs/hosts.share1
#
# New share1 config, adds dione and rhea
#
titan   titan.xyzco.com 1 0 server
tethys tethys.xyzco.com 2 0
mimas   mimas.xyzco.com 0 0
dione   dione.xyzco.com 0 0
rhea     rhea.xyzco.com 0 0

titan# samsharefs -u share1
#
# Host file for family set 'share1'
#
# Version: 2    Generation: 15    Count: 5
# Server = host 0/titan, length = 162
#
titan titan.xyzco.com 1 0
tethys tethys.xyzco.com 2 0
mimas mimas.xyzco.com 0 0
dione dione.xyzco.com 0 0
rhea rhea.xyzco.com 0 0
```

Example 3. The following example shows how the hosts file can be modified to specify a new Sun QFS shared file system server when the file system is mounted.

```
tethys# samsharefs -s tethys share1
#
# Host file for family set 'share1'
#
# Version: 2    Generation: 16    Count: 5
# Server = host 1/tethys, length = 162
#
titan titan.xyzco.com 1 0
tethys tethys.xyzco.com 2 0
mimas mimas.xyzco.com 0 0
dione dione.xyzco.com 0 0
rhea rhea.xyzco.com 0 0
```

FILES

The hosts file for a Sun QFS shared file system is initialized from:

/etc/opt/SUNWsamfs/hosts.*filename*

CAUTION

The **−R** option **must not** be used on a mounted file system to change the metadata server host without first stopping, disabling, or disconnecting the active metadata server and ensuring that it is restarted before accessing the filesystem again. Doing so will cause file system corruption.

SEE ALSO

sammkfs(1M).

NAME

samsnoop − SAM version of snoop

SYNOPSIS

samsnoop [-*aCDNPSvV*] [-*t [r | a | d*] [-c maxcount]
[-d device] [-i filename] [-n filename]
[-o filename] [-p first [, last]] [-s snaplen]
[-x offset [, length]] [expression]

AVAILABILITY

SUNWsamfs

DESCRIPTION

samsnoop is a version of **snoop**(1M) modified to capture and display packets from the SAM-FS shared file system. The arguments are identical to those of **snoop**(1M).
Note that **samsnoop** only recognizes packets on ports 7105-7109 as belonging to the SAM-FS shared file system.

EXAMPLES

Example 1: A sample output of the snoop command.

Capture all packets on port 7107 and display them as they are received:

```
example# /opt/SUNWsamfs/unsupported/samsnoop port 7107
```

Capture packets on port 7107 and save them to a file:

```
example# /opt/SUNWsamfs/unsupported/samsnoop -o /tmp/snoopy port 7107
```

Capture packets on port 7105 and save them to a
file. Then inspect the packets using times (in seconds)
relative to the first captured packet:

```
example# /opt/SUNWsamfs/unsupported/samsnoop -o cap port 7105
example$ /opt/SUNWsamfs/unsupported/samsnoop -i cap  -t r | more
```

Look at selected packets in another capture file:

```
example$ /opt/SUNWsamfs/unsupported/samsnoop -i pkts  -p99,108
```

Packet 101 Looks interesting. Take a look in more detail:

```
example$ /opt/SUNWsamfs/unsupported/samsnoop -i pkts -v -p101
```

SEE ALSO

For information about snoop, type "man snoop"

NAME

samtool − Graphical user interface for starting Sun SAM-FS and Sun SAM-QFS administrative tools

SYNOPSIS

/opt/SUNWsamfs/sbin/samtool

AVAILABILITY

SUNWsamfs

DESCRIPTION

samtool contains programs that help you manage and view information about the robots, devices, and pending mount requests associated with Sun SAM-FS and Sun SAM-QFS environments.

samtool displays three icons, one for each of the available Sun SAM-FS or Sun SAM-QFS administrative tools: **robottool**, **devicetool**, and **previewtool**. **robottool** is the tool that manages robots (see **robottool**(1M)). **devicetool** displays information about and manages individual devices (see **devicetool**(1M)). **previewtool** displays pending mount requests (see **previewtool**(1M)).

A brief description of each tool, including *samtool*, is displayed when selecting that tool from the *samtool* help menu. This menu appears when you click the mouse *menu* button on the *Help* button in *samtool*.

To start one of the programs in *samtool*, click once on its icon.

FILES

mcf	The configuration file for Sun SAM-FS and Sun SAM-QFS environments

SEE ALSO

devicetool(1M), **previewtool**(1M), **robottool**(1M), **mcf**(4)

NAME

samtrace − Dumps the Sun QFS, Sun SAM-FS, or Sun SAM-QFS trace buffer

SYNOPSIS

samtrace [**−d** *corefile*] [**−n** *namelist*] [**−s**] [**−v**] [**−V**]

AVAILABILITY

SUNWqfs

SUNWsamfs

DESCRIPTION

samtrace dumps the contents of the trace buffer for the mounted file system.

OPTIONS

−d *corefile*
The name of the corefile containing an image of the system memory. If no *corefile* is specified the default is to use the **/dev/mem** or **/dev/kmem** file from the running system.

−n *namelist*
The name of the namelist file corresponding to the corefile. If none is specified the default is to use **/dev/ksyms** from the running system.

−s Dumps the sam-init command queue. Includes -v output.

−v Verbose option, excluding inode free and hash chains.

−V Verbose option, including inode free and hash chains. Includes -v output.

NOTE

samtrace is a utility that is used to provide Sun Microsystems analysts with troubleshooting information. It is not intended for general use at your site.

FILES

/dev/kmem	Special file that is an image of the physical memory of the computer.
/dev/mem	Special file that is an image of the kernel virtual memory of the computer.
/dev/ksyms	Character special file of kernel symbols.

NAME

samu − Sun QFS, SAM-FS and SAM-QFS operator utility

SYNOPSIS

samu [−**d** *c*] [−**r** *i*] [−**c** *string*] [−**f** *cmd-file*]

AVAILABILITY

SUNWqfs
SUNWsamfs

DESCRIPTION

samu is a full screen operator interface for Sun QFS, SAM-FS, and SAM-QFS environments. It has a number of displays that show the status of file systems and devices and allows the operator to control file systems and removable media devices.

OPTIONS

−**d** *c*	Specifies the initial display when **samu** starts execution. See DISPLAYS below.
−**r** *i*	Specifies the time interval in seconds for refreshing the display window.
−**c** *string*	Specifies an initial command string that should be executed when **samu** starts execution.
−**f** *cmd-file*	Specifies a file from which to read samu commands. Each line in the file is a command.

CONTROL KEYS

The following "hot" keys are available for all displays:

q	Quit
:	Enter command
space	Refresh display
control-l	Refresh display (clear)
control-r	Enable/disable refresh (default is enabled)

The following keys perform the listed functions for each of the displays shown:

Key	Function	Display
control-f	Next file system	:a,a
	Page forward	c,h,o,p,s,t,u,v,w,A,J,M
	Next inode	I
	Next sector	S
	Next equipment	T,U
	Next filesystem	N
control-b	Previous file system	:a,a
	Page backward	c,h,o,p,s,t,u,v,w,A,J,M
	Previous inode	I
	Previous sector	S
	Previous equipment	T,U
	Previous filesystem	N
control-d	Half-page forward	c,p,s,u,w,A,J,M
	Next robot catalog	v
	Page forward	h,S
	Page arcopies forward	a
	Page partitions forward	N
control-u	Half-page backward	c,p,s,u,w,A,J,M
	Previous robot catalog	v
	Page backward	h,S

	Page arcopies backward	a
	Page partitions backward	N
control-k	Advance display format	A,I,S
	Select (manual,robotic,both,priority)	p
	Advance sort key	v
	Toggle path display	n,u,w
control-i	Detailed (2-line) display format	v
1-7	Select sort key	v
/	Search for VSN	v
%	Search for barcode	v

The sort selections for the v display are: 1 slot, 2 count, 3 usage, 4 VSN, 5 access time, 6 barcode, 7 label time.

DISPLAYS

The following displays are available. Those displays marked with '*' are the only ones available for Sun QFS. Those displays marked with '@' are additionally available under Sun SAM-FS/SAM-QFS. All others are available under Sun SAM-FS/SAM-QFS only if "samd start" has been executed.

a@	Display archiver status		
c	Display configuration	C	Memory
d*	Display tracing info.		
f*	Display filesystem info.	F	Optical disk label
h*	Display help information		
l@	Display license information	I*	Inode
m*	Display mass-storage status	J	Preview shared memory
n@	Display staging activity	L	Shared memory tables
o	Display optical disk status	M	Shared memory
p	Display mount request preview	N*	File system parameters
r	Display removable media	R	SAM-Remote info
s	Display device status summary	S	Sector data
t	Display tape status	T	SCSI sense data
u	Display stage queue	U	Device table
v	Display robot VSN catalog		
w	Display pending stage queue		

COMMANDS

The following commands may be entered after a colon (:).

Archiver commands:

aridle [dk | rm | fs.fsname]	Idle archiving
arrestart	Restart archiver
arrun [dk | rm | fs.fsname]	Start archiving
arstop [dk | rm | fs.fsname]	Stop archiving

Display control commands:

refresh i	Set refresh time
a filesystem	Select detailed "a" display
n media	Set n display media selection
p media	Set p display media selection
r media	Set r display media selection

u media	Set u display media selection
v eq	Set v display robot catalog
w media	Set w display media selection

Device commands:

devlog	eq [option ...]	Set device logging options
down	eq	Mark equipment down
idle	eq	Idle equipment
off	eq	Off equipment
on	eq	On equipment
readonly	eq	Mark equipment read-only
ro	eq	Mark equipment read-only
unavail	eq	Mark equipment unavailable
unload	eq	Unload mounted media/magazine

File System commands:

meta_timeo	eq interval	Set shared fs meta cache timeout
notrace	eq	Turn off file system tracing
partial	eq size	Set default size in kB left online after release
readahead	eq contig	Set maximum readahead in 1k blocks
thresh	eq high low	Set high and low release thresholds
trace	eq	Turn on file system tracing
writebehind	eq contig	Set maximum writebehind in 1k blocks

Robot commands:

audit	[-e] eq[:slot[:side]]	Audit slot or library
import	eq	Import cartridge from mailbox
export	eq:slot	Export cartridge to mailbox
export	mt.vsn	Export cartridge to mailbox
load	eq:slot[:side]	Load cartridge in drive
load	mt.vsn	Load cartridge in drive

Miscellaneous commands:

clear	vsn [index]	Clear load request
dtrace	daemon[.variable] value	Set daemon trace controls
mount	mntpt	Select a mount point (I, N displays)
open	eq	Open device (F, S displays)
read	addr	Read device
snap	file	Snapshot screen to file
stidle		Idle staging
strun		Start staging
!shell-command		Run shell command

SEE ALSO

curses(3), **mcf**(4).

NAME

samunhold − Releases SANergy file holds

SYNOPSIS

/opt/SUNWsamfs/sbin/samunhold *mntpoint*

AVAILABILITY

SUNWqfs

SUNWsamfs

DESCRIPTION

The **samunhold** command can be used to release SANergy file holds. These holds can be detected when attempts are made to unmount a file system with the **umount**(1M) command. If holds are present, the **umount**(1M) command generates log messages such as the following:

```
Inode XXXX: held by SAN, refcnt = N
```

SANergy File Sharing uses the following two types of leases, both of which require holds:

- Read leases, which typically expire within a few seconds.
- Write leases, which can extend for as long as an hour.

It is preferable to allow SANergy File Sharing to clean up the leases, but in an emergency, or in case of a SANergy File Sharing system failure, the administrator can use the **samunhold** command to avoid a reboot.

The **samunhold** command should only be run when SANergy File Sharing has held inodes and is preventing a file system from being unmounted. Prior to executing this command, the administrator should ensure the following:

- There are no SANergy applications running on any client, possibly including the server itself.
- The file system in question is not fused on any SANergy clients.
- The file system is not NFS mounted.

OPTIONS

The **samunhold** command releases all held inodes (files) on the file system whose root directory is the named *mntpoint* argument. The **samunhold** command must be run as root.

EXAMPLES

The following example shows the **samunhold** command:

```
bilbo# samunhold /sam1
bilbo# umount /sam1
```

SEE ALSO

samgetmap(1M), **samgetvol**(1M), **umount**(1M).

NAME

sefreport − Displays the content of the System Error Facility (SEF) log

SYNOPSIS

/opt/SUNWsamfs/sbin/sefreport [**−v**] [**−d**] *file*

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **sefreport** command reads the content of a Sun SAM-FS or Sun SAM-QFS SEF log file and writes its output to **stdout** in a human-readable format. By default, the log file is **/var/opt/SUNWsamfs/sef/sefdata**. The SEF log file contains the data gathered from the log sense pages of peripheral tape devices used by Sun SAM-FS and Sun SAM-QFS file systems. For more information on the SEF log file, including its content and format, see the **sefdata**(4) man page.

The **sefreport** command reads the input file specified by the *file* argument. If no other options are specified, the **sefreport** command examines the SEF log file and generates the following information for each record contained in *file*:

- The first header line states the record number, which is its ordinal position in the file.
- The second header line contains the timestamp of the record, the vendor name of the device from which the log sense data was received, the product name of the device, the revision level of the device's firmware, the string **VSN**, and the Volume Serial Name (VSN) of the volume mounted in the device when the log sense data was generated.

Following the header lines, the log sense data for each page in the record is printed. For each log sense page, a line identifying the page code is printed, followed by a line of column headings. The data is then printed in three columns per line with the following headings: parameter code, control, and parameter value. All data is generated in hexadecimal notation. For the meanings of the parameter codes, control bits, and parameter values, see your vendor documentation for the specific device.

OPTIONS

This command accepts the following options:

−d Includes additional device information. For each record, the command generates a third header line that identifies the equipment number of the device as configured in the **mcf** file and the path name of the device.

−v Generates verbose output. On each line of log sense data output, an additional string containing the equipment number, page code, and VSN is printed. This string is enclosed in parentheses and the items are colon-separated.

OPERANDS

This command accepts the following operand, which must be specified:

file Specifies the SEF log file. The SEF log file can be read from its default location (**/var/opt/SUNWsamfs/sef/sefdata**) or it can be redirected to another file for SEF processing.

EXAMPLES

Example 1. Assume that your system is set up to write SEF values to file **/var/opt/SUNWsamfs/sef/sefdata.mid**. You have entered the following command to write the SEF data using the report formatter:

```
srvr# sefreport /var/opt/SUNWsamfs/sef/sefdata.mid > ~mydir/sef.short
```

The file ˜**mydir/sef.short** is as follows:

```
Record no. 1
Mon Mar 26 11:17:48 2001  STK      9840             1.25 VSN 002981
```

```
     PAGE CODE 2
     param code  control   param value
         00h       74h     0x0
         01h       74h     0x0
         02h       74h     0x0
         03h       74h     0x0
         04h       74h     0x0
         05h       74h     0x40050
         06h       74h     0x0


     PAGE CODE 3
     param code  control   param value
         00h       74h     0x0
         01h       74h     0x0
         02h       74h     0x0
         03h       74h     0x0
         04h       74h     0x0
         05h       74h     0x140
         06h       74h     0x0


     PAGE CODE 6
     param code  control   param value
         00h       74h     0x0


  Record no. 2
  Mon Mar 26 11:30:06 2001  STK      9840             1.25 VSN 002999

     PAGE CODE 2
     param code  control   param value
         00h       74h     0x0
         01h       74h     0x0
         02h       74h     0x0
         03h       74h     0x0
         04h       74h     0x0
         05h       74h     0x1400a0
         06h       74h     0x0


     PAGE CODE 3
     param code  control   param value
         00h       74h     0x0
         01h       74h     0x0
         02h       74h     0x0
         03h       74h     0x0
         04h       74h     0x0
         05h       74h     0x190
         06h       74h     0x0


     PAGE CODE 6
```

```
   param code  control   param value
       00h       74h     0x0

<<<NOTE:  This output has been truncated for inclusion on this
man page.>>>
```

Example 2: Assume that you also need to produce a report with additional data. You can use the same log file as in Example 1, but you want this report to contain more information than **sef.short**, so you invoke **sefreport** with the **−d** and **−v** options. The following command is entered:

```
srvr# sefreport -d -v /var/opt/SUNWsamfs/sef/sefdata.mid > ~mydir/sef.long
```

The file ˜**mydir/sef.long** is as follows:

```
Record no. 1
Mon Mar 26 11:17:48 2001  STK      9840             1.25 VSN 002981
   Eq no. 32   Dev name /dev/rmt/1cbn

 rec  pg cd   param code  control   param value
   1    2         00h       74h     0x0          (32:2:002981)
   1    2         01h       74h     0x0          (32:2:002981)
   1    2         02h       74h     0x0          (32:2:002981)
   1    2         03h       74h     0x0          (32:2:002981)
   1    2         04h       74h     0x0          (32:2:002981)
   1    2         05h       74h     0x40050      (32:2:002981)
   1    2         06h       74h     0x0          (32:2:002981)


 rec  pg cd   param code  control   param value
   1    3         00h       74h     0x0          (32:3:002981)
   1    3         01h       74h     0x0          (32:3:002981)
   1    3         02h       74h     0x0          (32:3:002981)
   1    3         03h       74h     0x0          (32:3:002981)
   1    3         04h       74h     0x0          (32:3:002981)
   1    3         05h       74h     0x140        (32:3:002981)
   1    3         06h       74h     0x0          (32:3:002981)


 rec  pg cd   param code  control   param value
   1    6         00h       74h     0x0          (32:6:002981)


Record no. 2
Mon Mar 26 11:30:06 2001  STK      9840             1.25 VSN 002999
   Eq no. 31   Dev name /dev/rmt/0cbn

 rec  pg cd   param code  control   param value
   2    2         00h       74h     0x0          (31:2:002999)
   2    2         01h       74h     0x0          (31:2:002999)
   2    2         02h       74h     0x0          (31:2:002999)
   2    2         03h       74h     0x0          (31:2:002999)
   2    2         04h       74h     0x0          (31:2:002999)
   2    2         05h       74h     0x1400a0     (31:2:002999)
   2    2         06h       74h     0x0          (31:2:002999)
 rec  pg cd   param code  control   param value
   2    3         00h       74h     0x0          (31:3:002999)
   2    3         01h       74h     0x0          (31:3:002999)
   2    3         02h       74h     0x0          (31:3:002999)
   2    3         03h       74h     0x0          (31:3:002999)
   2    3         04h       74h     0x0          (31:3:002999)
   2    3         05h       74h     0x190        (31:3:002999)
   2    3         06h       74h     0x0          (31:3:002999)


 rec  pg cd   param code  control   param value
   2    6         00h       74h     0x0          (31:6:002999)

<<<NOTE:  This output has been truncated for inclusion on this
man page.>>>
```

FILES

/var/opt/SUNWsamfs/sef/sefdata
The default system error facility log file for Sun SAM-FS and Sun SAM-QFS file systems.

SEE ALSO

mcf(4), **sefdata**(4).

NAME

set_admin.sh − set administrator privileges for Sun SAM-FS and Sun SAM-QFS commands

SYNOPSIS

set_admin.sh [*sam-admin-group*]

AVAILABILITY

SUNWsamfs

DESCRIPTION

set_admin.sh changes the group and permissions of many of the Sun SAM-FS and Sun SAM-QFS administrator commands so they may be executed by users in a selected administrator group. You must be root to execute this command. *sam-admin-group* is the administrator group for the Sun SAM-FS and Sun SAM-QFS administrator commands. If you wish to change the administrator group back to the default, specify "bin" as the *sam-admin-group*. If *sam-admin-group* is not specified, the user will be prompted to enter it.

NAME

set_state − Set device state

SYNOPSIS

/opt/SUNWsamfs/sbin/set_state [**-w**] *state eq*

AVAILABILITY

SUNWsamfs

DESCRIPTION

set_state will change the state of a removable media device *eq* to *state*. If **−w** is specified, the command will wait for the operation to complete before terminating.

The valid states are:

on The device is usable by Sun SAM-FS or Sun SAM-QFS file systems. A device moving to the **on** state will be unloaded if there is media mounted.

idle The device will not be selected for use by either Sun SAM-FS or Sun SAM-QFS file systems. Any existing activity will be allowed to complete. Once there is no more activity, the device will be placed in the **off** state.

unavail The device is unavailable for use by Sun SAM-FS and Sun SAM-QFS file systems and most Sun SAM-FS and Sun SAM-QFS commands. The only valid commands for a device in this state are **load**(1M), **unload**(1M), and **set_state**(1M). A device moving to the **unavail** state will be unloaded if there is media mounted.

off The device is unusable by Sun SAM-FS and Sun SAM-QFS file systems. A device moving to the **off** state from **on**, **idle** or **unavail** will be unloaded if there is media mounted.

down The device is unusable by Sun SAM-FS and Sun SAM-QFS file systems. No attempt will be made to unload media from a device moving to the **down** state. The only state a **down** device may be moved to is **off**.

FILES

mcf The configuration file for Sun SAM-FS and Sun SAM-QFS environments.

SEE ALSO

load(1M), **unload**(1M), **mcf**(4), **sam-robotsd**(1M)

NAME

showqueue − Display content of an archiver queue files

SYNOPSIS

/opt/SUNWsamfs/sbin/showqueue [**−v**] [*filesystem*[*archreq* ...]]

AVAILABILITY

SUNWsamfs

DESCRIPTION

showqueue reads the archreq files named in the argument list and prints the information.

If there are no names in the argument list, all ArchReq files are printed for all mounted filesystems.

If there is only one name in the argument list, all ArchReq files are printed for that filesystem.

Otherwise, print only the listed ArchReq files.

OPTIONS

−v Print information about each file to be archived in the ArchReq files.

Example output for: *showqueue -v samfs4*

```
Archive request files for: samfs4
ArchReq:AllFileTypes.1.1 compose 2001/07/04 17:07:41
    count:300 space:29.4M (min: 100.9k) flags: sort
    size:17080, alloc:17080
    Archive file 1:
      ino: 111 flags:0 space:100.9k time:936909718 priority:1000
        s3719/t/file000
      ino: 109 flags:0 space:100.9k time:936909718 priority:1000
        s3719/t/file001
      ino: 108 flags:0 space:100.9k time:936909718 priority:1000
        s3719/t/file002
      ino: 107 flags:0 space:100.9k time:936909718 priority:1000
        s3719/t/file003
      ino: 106 flags:0 space:100.9k time:936909718 priority:1000
        s3719/t/file004
      ino: 105 flags:0 space:100.9k time:936909718 priority:1000
        s3719/t/file005
```

NAME

star − Creates tape archives and adds or extract files

SYNOPSIS

star [*options*] ... [*file*] ...

AVAILABILITY

SUNWsamfs

DESCRIPTION

This **man**(1) page describes the GNU version of the **tar**(1) command as extended by Sun Microsystems. Sun Microsystems has enhanced the **tar**(1) command to support the Sun SAM-FS and Sun SAM-QFS file systems. The **star** command saves many files together into a single tape or disk archive, and it can be used to restore individual files from the archive.

OPTIONS

This command accepts options in both single-character and multicharacter equivalent option formats.

Main Operation Mode Options

−**t**, −−**list**	Lists the content of an archive.
−**x**, −**extract**, −**get**	Extracts files from an archive.
−**c**, −−**create**	Creates a new archive.
−**d**, −−**diff**, −−**compare**	Finds differences between archive and file system.
−**r**, −−**append**	Appends files to the end of an archive.
−**u**, −−**update**	Only appends files newer than the copy in archive.
−**A**, −−**catenate**, −−**concatenate**	Appends **tar**(1) files to an archive.
−−**delete**	Deletes from the archive (not on mag tapes!).

Operation Modifier Options

−**W**, −−**verify**	Attempts to verify the archive after writing it.
−−**remove−files**	Removes files after adding them to the archive.
−**k**, −−**keep−old−files**	Does not overwrite existing files when extracting.
−**U**, −−**unlink−first**	Removes each file prior to extracting over it.
−−**recursive−unlink**	Empties hierarchies prior to extracting directory.
−**S**, −−**sparse**	Handles sparse files efficiently.
−**O**, −−**to−stdout**	Extracts files to standard output.
−**G**, −−**incremental**	Handles old GNU-format incremental backup.
−**g**, −−**listed−incremental**	Handles new GNU-format incremental backup.
−−**ignore−failed−read**	Does not exit with nonzero on unreadable files.

File Attribute Handling Options

−−**owner=***name*	Forces *name* as the owner for added files.
−−**group=***name*	Forces *name* as the group for added files.
−−**mode=***changes*	Forces (symbolic) mode *changes* for added files.
−−**atime−preserve**	Does not change access times on dumped files.
−**m**, −−**modification−time**	Does not extract file modified time.
−−**same−owner**	Tries extracting files with the same ownership.
−−**numeric−owner**	Specifies to always use numbers for user/group names.
−**p**, −−**same−permissions, −−preserve−permissions**	Extracts all protection information.
−**s**, −−**same−order**, −−**preserve−order**	Sorts names to extract to match archive.
−−**preserve**	Same as specifying both −**p** and −**s**.

Device Selection and Switching Options

−**f=***archive*, −−**file=***archive*	Uses archive file or device *archive*. The *archive* can be *file*, *host***:***file* or *user@host***:***file*.
−−**force−local**	Specifies that archive file is local even if has a colon.

−−**rsh**−**command=***command*	Specifies to use remote *command* instead of **rsh**.
−[**0-7**][**lmh**]	Specifies drive and density.
−**M**, −−**multi**−**volume**	Creates/lists/extracts multivolume archive.
−**L=***num*, −−**tape**−**length=***num*	Changes tape after writing *num* x 1024 bytes.
−**F=***file*, −−**info**−**script=***file*, −−**new**−**volume**−**script=***file*	Runs script in *file* at the end of each tape (implies −**M**).
−−**volno**−**file=***file*	Uses/updates the volume number in *file*.

Device Blocking Options

−**b=***blocks*, −−**blocking**−**factor=***blocks*	Specifies *blocks* x 512 bytes per record.
−−**record**−**size=***size*	Specifies *size* bytes per record, multiple of 512.
−**i**, −−**ignore**−**zeros**	Ignores zeroed blocks in archive (means EOF).
−**B**, −−**read**−**full**−**records**	Specifies to reblock as the file is being read (for 4.2BSD pipes).

Archive Format Selection Options

−**V=***name_or_pattern*, −−**label=***name_or_pattern*	Creates archive with volume name *name* or globbing pattern *pattern* at list/extract time.
−**o**, −−**old**−**archive**, −−**portability**	Writes a V7 format archive.
−−**posix**	Writes a POSIX-conformant archive (GNU). Support for POSIX is only partially implemented. The **star** command cannot read, nor can it produce, −−**posix** archives. If the **POSIXLY_CORRECT** environment variable is set, GNU extensions are disallowed with −−**posix**.
−**z**, −−**gzip**, −−**ungzip**	Filters the archive through **gzip**(1).
−**Z**, −−**compress**, −−**uncompress**	Filters the archive through **compress**(1).
−−**use**−**compress**−**program=***prog*	Filters through *prog* (must accept −**d**).

Local File Selection Options

−**C=***dir*, −−**directory=***dir*	Changes to directory *dir*.
−**T=***name*, −−**files**−**from=***name*	Gets names to extract or create from file *name*.
−−**null**	Instructs **star** to expect file names terminated with **NUL** characters so **star** can work correctly with file names that contain newline characters. Must be specified in conjunction with the −**t** or the −**files**−**from=***name* option. Disables the −**C** option.
−−**exclude=***pattern*	Excludes files, given as a globbing *pattern*.
−**X=***file*, −−**exclude**−**from=***file*	Excludes globbing patterns listed in *file*.
−**P**, −−**absolute**−**names**	Does not strip leading slash characters (/) from file names.
−**h**, −−**dereference**	Dumps instead the files to which symlinks point.
−−**no**−**recursion**	Avoids descending automatically in directories.
−**l**, −−**one**−**file**−**system**	Stays in local file system when creating archive.
−**K=***name*, −−**starting**−**file=***name*	Begins at file *name* in the archive.
−**n**, −−**newer_than_existing**	Only restores files newer than the existing copy.
−**N=***date*, −−**newer=***date*, −−**after**−**date=***date*	Only restores files newer than *date*.
−−**newer**−**mtime**	Compares date and time when data changed only.
−−**backup**[=*control*]	Backs up before removal, chooses version control. You can use the **VERSION_CONTROL** environment variable or the *control* argument to specify version control. The possible values for *control* are as follows:

control **Values**	**Version**
t, **numbered**	Makes numbered backups.
nil, **existing**	Makes numbered if numbered backups exist, simple otherwise.

	never, **simple** Specifies to always make simple backups.
−−**suffix=***suffix*	Backs up before removal. Overrides usual suffix. By default, the backup suffix is a tilde character (˜). You can use this option or the **SIMPLE_BACKUP_SUFFIX** environment variable to specify an alternative *suffix*.

Informative Output Message Options

−−**help**	Writes help text (which is this **man**(1) page), then exits.
−−**version**	Writes the **tar**(1) program version number, then exits.
−**v**, −−**verbose**	Lists files processed verbosely.
−−**checkpoint**	Writes directory names while reading the archive.
−−**totals**	Writes total bytes written while creating archive.
−**R**, −−**block**−**number**	Shows block number within archive with each message.
−**w**, −−**interactive**, −−**confirmation**	Prompts for confirmation for every action.

Input File Option

file	The *file* can be a file or a device.

NOTES

The **star**(1) command defaults to −**f**− and −**b20**.

Be careful when combining options. The **star**(1) command supports old-style tar combined options without the leading "-", e.g.

```
/opt/SUNWsamfs/sbin/star tvbf 128 file
```

sets the blocksize to 64K and uses "file" as the archive. However,

```
/opt/SUNWsamfs/sbin/star -tvbf 128 file
```

sets the blocksize to "f" and uses "128" as the archive. If you want to use the leading "-" you should separate the options as follows:

```
/opt/SUNWsamfs/sbin/star -tv -b 128 -f file
```

SEE ALSO

For more information about the **star**(1) command, enter the following command:

```
/opt/SUNWsamfs/sbin/star --help
```

tar(1)

NAME

tplabel − Label tape

SYNOPSIS

tplabel −vsn *vvvvvv* −[**new** | **old** *vv...*] [−**b** *blksize*] [−**w**] [−**V**] [−**erase**] *eq*

tplabel −vsn *vvvvvv* −[**new** | **old** *vv...*] [−**b** *blksize*] [−**w**] [−**V**] [−**erase**] *eq:slot*

tplabel −vsn *vvvvvv* −[**new** | **old** *vv...*] [−**b** *blksize*] [−**w**] [−**V**] [−**erase**] *eq:slot*

DESCRIPTION

tplabel labels the tape volume specified by *eq:slot.* *eq* is the equipment ordinal. If *eq* is a library, *slot* is the slot in the library containing the tape cartridge.

The following sequence of labels is written:

VOL1
HDR1
HDR2
tapemark
EOF1
tapemark
tapemark

The labels conform to ANSI X3.27-1987 File Structure and Labeling of Magnetic Tapes for Information Interchange.

−**vsn** *vvvvvv* specifies the volume serial name (VSN) of the tape being labeled. The VSN must be one to six characters in length. All characters in the VSN must be selected from the 26 upper-case letters, the 10 digits, and the following special characters: !"%&'()*+,-./:;<=>?_.

If the media being labeled was previously labeled, the VSN must be specified by −**old** *vv...*. The "old" VSN is compared with the VSN on the media to assure that the correct media is being relabeled.

If the media is not labeled (i.e., blank), −**new** must be specified to prevent the previous label comparison from being made.

The media type must be specified by −**media** *mm*.

OPTIONS

−**V** Verbose, lists label information written.

−**b** *blksize* specifies the blocksize for this tape. The value must be one of 16, 32, 64, 128, 256, 512, 1024 or 2048 and represents the size of the tape block in units of 1024. This option overrides the default blocksize.

−**erase** Erases the media completely before a label is written. This is a security feature that is normally not necessary. Complete media erasure will take a long time to perform since all data in the media is erased. On the AMPEX D2 this option will perform an initialize format of the tape prior to writing the labels.

−**w** Wait for the labeling operation to complete. If an error occurs, it will be reported along with a completion code of 1. All labeling errors are also logged. Note: Canceling a command that is waiting for completion will not cause the operation itself to be canceled.

NAME

unarchive − Delete archive entries

SYNOPSIS

unarchive [−**f**] [−**M**] [−**o**] −**c** *n* [−**m** *media*] [−**v** *vv...*] *filename*. . .

unarchive [−**f**] [−**M**] [−**o**] [−**c** *n*] −**m** *media* [−**v** *vv...*] *filename*. . .

unarchive [−**f**] [−**M**] [−**o**] −**c** *n* [−**m** *media*] [−**v** *vv...*] −**r** *dirname*. . . [*filename*. . .]

unarchive [−**f**] [−**M**] [−**o**] [−**c** *n*] −**m** *media* [−**v** *vv...*] −**r** *dirname*. . . [*filename*. . .]

AVAILABILITY

SUNWsamfs

DESCRIPTION

unarchive deletes archive entries for one or more files or directories. Selection is controlled by the the archive copy and/or the media type and VSN.

If one or more −**c** *n* are specified, only those archive copies (1 to 4) are deleted. The default is all copies. Either a −**c** or a −**m** option must be specified.

If −**m** *media* is specified, archive copies on the specified media are deleted. See **mcf**(4). Either a −**c** or a −**m** option must be specified.

If −**v** *vv...* is specified, −**m** must also be specified. In this case, archive copies on the specified VSN *vv...* are deleted.

OPTIONS

−**M** Unarchive meta data only. This includes directories, the segment index, and removable media files. Regular files and symbolic links are not unarchived.

−**o** Require the file to be on-line before its archive entry is deleted. If the file is off-line, **unarchive** will stage the file on to disk before deleting any entries.

−**f** Do not report errors.

−**r** Recursively delete the archive entries of the specified *dirname* and its subdirectories. The archive entries of files in the directories and subdirectories are deleted.

NOTES

If the last (undamaged) copy of a file would be unarchived, unarchive will report "Last undamaged offline copy" and that copy will not be unarchived.

SEE ALSO

mcf(4)

NAME

undamage − Undamage and unstale archive entries

SYNOPSIS

/opt/SUNWsamfs/sbin/undamage [**−f**] [**−c** *n*] [**−M**] [**−m** *media*] [**−v** *vv...*] *filename*. . .

/opt/SUNWsamfs/sbin/undamage [**−f**] [**−c** *n*] [**−M**] [**−m** *media*] [**−v** *vv...*] −r *dirname*. . . [*filename*. . .]

AVAILABILITY

SUNWsamfs

DESCRIPTION

undamage marks archive entries for one or more files or directories undamaged. **undamage** also unstales archive entries if offline. Selection is controlled by the archive copy and/or the media type and VSN.

If one or more **−c** *n* are specified, only those archive copies (1 to 4) are marked undamaged and unstale.

If **−m** *media* is specified, archive copies on the specified media are marked undamaged and unstale. See **mcf**(4).

If **−v** *vv...* is specified, **−m** must also be specified. In this case, archive copies on the specified vsn *vv...* are marked undamaged and unstale.

If all archive entries are undamaged, the file or directories is undamaged.

OPTIONS

−f Do not report errors.

−M Undamage meta data only. This includes directories, the segment index, and removable media files. Regular files and symbolic links are not undamaged.

−r Recursively mark the archive entries of the specified *dirname* and its subdirectories undamaged and unstale. The archive entries of files in the directories and subdirectories are marked undamaged and unstale.

EXAMPLE

Undamage all archive copies of "myfile".

```
undamage -c1 -c2 -c3 -c4 myfile
```

SEE ALSO

mcf(4)

NAME

unload − Unload media from a device

SYNOPSIS

/opt/SUNWsamfs/sbin/unload [**−w**] *eq*

AVAILABILITY

SUNWsamfs

DESCRIPTION

Unload the media mounted on device *eq*. The device specified by *eq* must be a removable media device or a media changer.

If *eq* is a removable media device controlled by a media changer, the medium will be moved into storage. This command is used when a shutdown of a Sun SAM-FS or Sun SAM-QFS file system is required and a tape is still in a drive. This command is also used in situations where the system administrator wishes to remove a tape from a drive that is currently in the **unavail** state.

If *eq* is a media changer, **unload** moves catalog entries from the media changer's catalog to the historian's catalog. The device state for device *eq* is set to **off**. When the device state for the media changer is set to **on** and the media changer has bar codes, then the catalog information for that media changer is retrieved from the historian. If the media changer does not have bar codes, an audit invoked by the administrator will recover the historian information. This command is useful for moving tapes in to and out of media changers which do not have import/export capabilities, or sense capability for open door. By first issuing the **unload** command, the system administrator can safely open the door to the media changer, add or remove tapes, close the door, and re-audit the media changer.

If **−w** is specified, the command will wait for the operation to complete before terminating.

FILES

mcf	The configuration file for Sun SAM-FS or Sun SAM-QFS environment

SEE ALSO

auditslot(1M), **historian**(7), **load**(1M), **set_state**(1M), **mcf**(4), **sam-robotsd**(1M)

NAME

unrearch − Marks archive entries to be not rearchived

SYNOPSIS

unrearch [−**f**] [−**M**] [−**c** *n*] [−**m** *media*] [−**v** *vv...*] *filename*. . .

unrearch [−**f**] [−**M**] [−**c** *n*] [−**m** *media*] [−**v** *vv...*] −**r** *dirname*. . . [*filename*. . .]

AVAILABILITY

SUNWsamfs

DESCRIPTION

unrearch marks archive entries for one or more files or directories to be not rearchived. Selection is controlled by the archive copy and/or the media type and VSN.

If one or more −**c** *n* are specified, only those archive copies (1 to 4) are marked. The default is all copies.

If −**m** *media* is specified, archive copies on the specified media are marked. See **mcf**(4).

If −**v** *vv...* is specified, −**m** must also be specified. In this case, archive copies on the specified VSN *vv...* are marked.

OPTIONS

−**f** Do not report errors.

−**M** Unrearchive meta data only. This includes directories, the segment index, and removable media files. Regular files and symbolic links are not unrearchived.

−**r** Recursively unrearchive the archive entries of the specified *dirname* and its subdirectories. The rearch flag for archive entries of files in the directories and subdirectories is cleared.

SEE ALSO

mcf(4)

NAME

unreserve − Unreserve a volume for archiving.

SYNOPSIS

/opt/SUNWsamfs/sbin/unreserve *mediatype***.***vsn*
/opt/SUNWsamfs/sbin/unreserve *eq***:***slot***[:***partition***]**

AVAILABILITY

SUNWsamfs

DESCRIPTION

unreserve removes the assignment of the volume for archival of specific files.

Normally, relabeling a volume will remove the reservation of a volumes. This command is provided to unreserve a volume without re-labeling.

The volume is determined by the specifier *mediatype***.***vsn* , or *eq***:***slot***[:***partition***]**

SEE ALSO

archiver(1M), **archiver.cmd**(1M), **reserve**(1M)

NAME

intro_libsam, **intro_libsamrpc** − Introduces the Sun SAM-FS Application Programmer Interface (API) routines

AVAILABILITY

Sun SAM-FS or SAM-QFS File System
Sun SAM-FS or SAM-QFS Environment

DESCRIPTION

The SAM-FS API allows an SAM-FS file to be requested from within an application program. The aplication program can reside either on the machine upon which the Sun SAM-FS or SAM-QFS file system is running or on another machine on the network. This man page provides an introduction to the Sun SAM-FS and SAM-QFS API routines.

The following topics are presented:

- API overview
- API library routines
- Using **libsam**
- Using **libsamrpc**

API OVERVIEW

When a request is made, the process or program making the request is the client process or program, running on the client machine. The requests are received and processed by the server, running on the server, or host, machine. For the API routines, the server machine is always the machine upon which the Sun SAM-FS or SAM-QFS file system is running.

In the simplest case, the client and server machines are the same, and no network communication is necessary. In other cases, however, he application programmer needs to allow for the client program to run on a machine where the Sun SAM-FS and SAM-QFS file system] is not running. In this case, networked library calls must be used.

The two API libraries available with the Sun SAM-FS and SAM-QFS file systems are as follows:

- **libsam**. The library calls in **libsam** do not perform network communication. They only make local requests. In this case, each library call makes a system call, and the server is the local operating system.
- **libsamrpc**. The library calls in **libsamrpc** use Remote Procedure Calls (RPCs) to communicate with a special server process, **sam-rpcd**. Because of the RPC mechanism, the client and server can exist on the same machine or on different machines in the network. The server process always runs on the machine upon which the Sun SAM-FS and SAM-QFS file system is running.

Both **libsam** and **libsamrpc** are released in shared object (**.so**) and archive (**.a**) format for Solaris platforms. **libsam.so** and **libsam.a** are installed in **/opt/SUNWsamfs/lib**. **libsamrpc.so** and **libsamrpc.a** are installed in **/opt/SUNWsamfs/client/lib**, with symbolic links to them in **/opt/SUNWsamfs/lib**.

API LIBRARY ROUTINES

The library calls included in the Sun SAM-FS and SAM-QFS API include calls for the Sun QFS, SAM-FS, and SAM-QFS environments. In addition, some are supported in **libsam** and some are supported in **libsamrpc**.

Table 1 lists the API library routines and indicates the environments in which they are supported. In addition, table 1 indicates the libraries in which they are included:

Table 1. Library routine availability

Routine	**Description**
sam_advise	Sets file attributes.

Availability: Sun QFS, SAM-FS, and SAM-QFS environments.
Libraries: **libsam**.

sam_archive Sets archive attributes on a file.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsam** and **libsamrpc**.

sam_cancelstage Cancels a pending or in-progress stage on a file.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsam**.

sam_closecat Ends access to the catalog for an automated library.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsam**.

sam_closerpc Closes down the RPC connection.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsamrpc**.

sam_devstat, **sam_ndevstat**
Gets device status. **sam_ndevstat** accepts a longer device name.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsam**.

sam_devstr Translates numeric device status into a character string.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsam**.

sam_getcatalog Obtains a range of entries from the catalog for an automated library.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsam**.

sam_initrpc Initializes the RPC connection.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsamrpc**.

sam_opencat Accesses the VSN catalog for an automated library.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsam**.

sam_readrminfo Gets information for a removable media file.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsam**.

sam_release Releases and sets release attributes on a file.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsam** and **libsamrpc**.

sam_request Creates a removable media file.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsam**.

sam_restore_copy
Creates an archive copy for a file.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsam**.

sam_restore_file Creates an offline file.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsam**.

sam_segment Sets segment attributes on a file or directory. Requires a Sun SAM-FS Segment

license.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsam** and **libsamrpc**.

sam_setfa Sets file attributes.
Availability: Sun QFS, SAM-FS and SAM-QFS environments.
Libraries: **libsam** and **libsamrpc**.

sam_ssum Sets checksum attributes on a file.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsam**.

sam_stage Stages and sets stage attributes on a file.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsam** and **libsamrpc**.

sam_stat, **sam_lstat**
sam_stat obtains file information and follows symbolic links to the file. **sam_lstat** obtains file information, and if that file is a link, it returns information about the link.
Availability: Sun QFS, SAM-FS, and SAM-QFS environments.
Libraries: **libsam** and **libsamrpc**.

sam_vsn_stat Creates an archive copy for a file.
Availability: Sun SAM-FS and SAM-QFS environments.
Libraries: **libsam**.

All APIs in **libsam**, except for **sam_closecat**, **sam_getcatalog**, and **sam_opencat**, are available for use with 64-bit programs. Sun Microsystems, Inc. does not support a 64-bit version of **libsamrpc**.

The APIs for manipulating archive copies are supported for archiving to cartridges in libraries. They are not supported for disk archive copies.

For more details about each library routine, see the individual corresponding man page for that routine. Library routines contained in **libsam** are found in section 3 of the online man pages. Library routines contained in **libsamrpc** are found in section 3X of the online man pages.

USING libsam

No special initialization or configuration is required prior to using the API library routines in **libsam**. The application program must be linked with **libsam**, however. For information on the routines, see the individual **libsam** man pages, all of which are listed in the **SEE ALSO** section of this man page.

USING libsamrpc

The source code for **libsamrpc** is included in the release for customers who wish to write and run application programs on platforms that do not run the Solaris operating system. In these cases, the library must be ported to the client machine. The source code is located in **/opt/SUNWsamfs/client/src**. Example application programs are located in **/opt/SUNWsamfs/client/examples**.

Specifying the Server Machine

A call to **sam_initrpc** is required before any other RPC client API calls can be executed successfully. Only one **sam_initrpc** call is required, followed by any number of other client API calls (other than **sam_closerpc**). The **sam_initrpc** call accepts one argument: a pointer to a character string that specifies the name of the server machine. If this pointer is **NULL**, **sam_initrpc** checks for an environment variable named **SAMHOST**. If this environment variable is set, that name is used for the server machine. If there is no **SAMHOST** environment variable, the default server name **samhost** is used.

In summary, the name of the server machine can be specified in any of three ways, which are checked by **sam_initrpc** in the following order:

1. As an argument to the **sam_initrpc** call.

2. As the environment variable **SAMHOST**.

3. By accepting the default server name, **samhost**.

RPC Server Process

The RPC API server process receives and processes requests from the client. This server process, **/opt/SUNWsamfs/sbin/sam-rpcd**, must be run on the same machine as the file system. The **sam-rpcd** daemon must be running for client requests to execute successfully.

The **sam-rpcd** daemon is started automatically by **sam-initd** if the appropriate entry is made in the **defaults.conf** file. For information on editing the **defaults.conf** file, see **Configuring the API** later in this man page.

The **sam-rpcd** daemon can also be started manually. It should be run as superuser. The **sam-rpcd** command accepts no arguments.

The **sam-rpcd** daemon services the requests it receives by making the appropriate system call on the server machine and then returning the output or result to the client. For more information on this daemon, see the **sam-rpcd**(1M) man page.

Configuring the API

The following steps describe setting up the API server and clients. These steps assume that your software is properly configured and running.

Step 1: Configure the API Server

For the server portion of the API to run successfully, the following conditions must be present:

- The RPC program name and number pair must be known on the server machine
- The RPC program name and number pair must be the same as the pair used on the API client machines.

Make an entry for the RPC program name and number. The RPC program number is a number chosen by you. The RPC program name is **samfs**. The name and number pair must be the same on the server and all clients. The **/etc/nsswitch.conf** file determines where you should specify the RPC program name and number pair. For more information on this, see the **nsswitch.conf**(4) man page.

In **/etc/rpc** (or the NIS database), add the following line:

```
samfs       150005
```

In **/etc/services** (or the NIS database), add the following line:

```
samfs       5012/tcp    # Sun SAM-FS API
```

The API server is started automatically by the **sam-initd** daemon if the following entry is made in the **defaults.conf** file (note that changes to the **defaults.conf** file do not take effect until the next time the **sam-initd** daemon is initialized):

```
samrpc = on
```

The **sam-rpcd** daemon is not automatically started if no entry for it appears in the **defaults.conf** file or if the following entry appears in the file:

```
samrpc = off
```

For more information about the **defaults.conf** file, see the **defaults.conf**(4) man page.

Step 2: Configure the API Client Machines

The following two configuration components must be present on the client machine for the RPC communication to be successful:

- The name of the server machine.
- The RPC program name and number pair.

Make an entry for the RPC program name and number on all client machines, as you did on the API server machine previously. Again, the RPC program name must be **samfs**. The RPC program number is a number chosen by you, but it must be the same on the server and client machines.

In **/etc/rpc** (or the NIS database), add the following line:

```
samfs        150005
```

The host name of the server machine must be known on the client machine. For default cases, the host name samhost must be listed as an alias for the Sun SAM-FS and SAM-QFS file system server machine. For more information, see the **sam_initrpc**(3X) man page.

Authentication and libsamrpc

Authentication information is generated at the time of the **sam-initrpc** call. This information consists of the user identification (**uid**) and group identification (**gid**) of the calling process. It is associated with the connection made to the RPC server process.

Subsequent **libsamrpc** calls have this information associated. When the request is received by the RPC server process on the server machine, the uid and gid information is used. File access and operations are granted or denied based on this information.

It is important that the server machine have a common **uid** and **gid** space with the client machines.

SEE ALSO

sam_advise(3), **sam_archive**(3), **sam_cancelstage**(3), **sam_closecat**(3), **sam_devstat**(3), **sam_devstr**(3), **sam_getcatalog**(3), **sam_lstat**(3), **sam_ndevstat**(3), **sam_opencat**(3), **sam_readrminfo**(3), **sam_release**(3), **sam_request**(3), **sam_restore_copy**(3), **sam_restore_file**(3), **sam_segment**(3), **sam_setfa**(3), **sam_ssum**(3), **sam_stage**(3), **sam_stat**(3).

sam_archive(3X), **sam_closerpc**(3X), **sam_initrpc**(3X), **sam_lstat**(3X), **sam_release**(3X), **sam_stage**(3X), **sam_stat**(3X).

NAME

sam_advise − provide advice to the filesystem

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsam [** *library* **...]**

#include "**/opt/SUNWsamfs/include/lib.h**"

int sam_advise(const int *fildes***, const char** **ops***);**

DESCRIPTION

sam_advise( ) provides advice about expected behavior of the application when accessing data in the file associated with the open file descriptor, *fildes.* **sam_advise( )** provides advice for a file using a Sun SAM-FS or SAM-QFS ioctl call. The last caller of sam_advise() sets the advice for all applications using the file. The last close of the file sets the advice back to the default mode.

ops is the character string of options, for example: "dw". Individual options are described below.

OPTIONS

d Return the advice on the file to the default, i.e. the ***qwrite*** is reset to the mount setting. When this option is specified, the advice is reset to the default. If it is used, it should be the first character in the string.

r Advises the system to use direct (raw) I/O (see directio(3C) for Solaris 2.6 and above). The default I/O mode is buffered (uses the page cache). At the last close, the type of I/O is set back to paged or direct based on the mount option **forcedirectio** or the **directio** attribute set by the **setfa** command.

p Advises the system that the I/O buffers used for direct I/O are locked by the user (see mlock(3C)). This means the system does not lock them. The user must be superuser to lock buffers. The buffers must be initialized before issuing mlock. If the buffers are not initialized, an EIO error is returned for each I/O. The default means the system locks and unlocks the I/O buffers used in direct I/O. The **p** option is only supported by the *ma* equipment type filesystem. (See man **mcf**(4)).

w Advises the system to enable simultaneous reads and writes to the same file from different threads. See the ***qwrite*** parameter on the mount command. The **w** option is only supported by the *ma* equipment type filesystem. (See man **mcf**(4)).

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

sam_advise( ) fails if one or more of the following are true:

EINVAL	An invalid option was specified, or the file is not a regular file.
EPERM	Not the owner or super-user.
EFAULT	*path* or *ops* points to an illegal address.
EINTR	A signal was caught during the **sam_advise( )** function.
ELOOP	Too many symbolic links were encountered in translating *path*.
EMULTIHOP	Components of *path* require hopping to multiple remote machines and the file system does not allow it.
ENAMETOOLONG	The length of the *path* argument exceeds {***PATH_MAX***}, or the length of a *path* component exceeds {***NAME_MAX***} while {***_POSIX_NO_TRUNC***} is in effect.
ENOENT	The named file does not exist or is the null pathname.

ENOLINK	*path* points to a remote machine and the link to that machine is no longer active.
ENOTDIR	A component of the path prefix is not a directory.

SEE ALSO

setfa(1), **sam_setfa**(3), **directio**(3C), **mlock**(3C), **mount_samfs**(1M), **mcf**(4)

NAME

sam_archive − Set archive attributes on a file or directory

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsam [** *library* **...]**

#include "**/opt/SUNWsamfs/include/lib.h**"

int sam_archive(const char **path*, **const char** **ops***);**

DESCRIPTION

sam_archive( ) sets archive attributes on a file or directory using a Sun SAM-FS or SAM-QFS system call. *path* is the file on which to set the attributes. *ops* is the character string of options, for example: "dn". Individual options are described below.

OPTIONS

C Specifies concurrent archiving for this file. This file can be archived even if opened for write. The archive time is regulated by the modification time. Note, nfs files are not opened and are by default concurrently archived. Concurrent archiving is useful for databases, however caution is advised since archiving can occur while the file is being modified and this can result in wasted media. The default is to disallow archiving while the file is opened for write.

d Return the archive attributes on the file to the default, i.e. archive the file according to the archiver rules. When this option is specified, the attributes are reset to the default. If it is used, it should be the first character in the string.

i Specifies that the file be immediately archived if it is not already archived.

w Wait for the file to have at least 1 archive copy before completing. Not valid with *d* or *n*.

Note that it may take a long time for the file to be archived.

n Specifies that this file never be archived. Not valid with either of the checksum *g* (*generate*) or *u* (*use*) attributes. (See **ssum**(1) or **sam_ssum**(3)).

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

sam_archive( ) fails if one or more of the following are true:

EINVAL	An invalid option was specified, or the file is neither a regular file nor a directory.
EPERM	Not the owner or super-user.
EFAULT	*path* or *ops* points to an illegal address.
EINTR	A signal was caught during the **sam_archive( )** function.
ELOOP	Too many symbolic links were encountered in translating *path*.
ENAMETOOLONG	The length of the *path* argument exceeds {***PATH_MAX***}, or the length of a *path* component exceeds {***NAME_MAX***} while {***_POSIX_NO_TRUNC***} is in effect.
ENOENT	The named file does not exist or is the null pathname.
ENOLINK	*path* points to a remote machine and the link to that machine is no longer active.
ENOTDIR	A component of the path prefix is not a directory.

SEE ALSO

archive(1), **ssum**(1), **sam_ssum**(3)

NAME

sam_cancelstage − Cancel a file stage

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsam [** *library* **...]**

#include "/opt/SUNWsamfs/include/lib.h"

int sam_cancelstage(const char **path***)**

DESCRIPTION

sam_cancelstage() cancels the stage of the file pointed to by *path*. Only the file owner or superuser can perform this operation on the file.

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

sam_cancelstage() fails if one or more of the following are true:

EPERM	Caller is not the file owner or superuser.
EFAULT	*buf* or *path* points to an illegal address.
EINTR	A signal was caught during the **sam_cancelstage()** function.
ELOOP	Too many symbolic links were encountered in translating *path*.
EMULTIHOP	Components of *path* require hopping to multiple remote machines and the file system does not allow it.
ENAMETOOLONG	The length of the *path* argument exceeds {***PATH_MAX***}, or the length of a *path* component exceeds {***NAME_MAX***} while {***_POSIX_NO_TRUNC***} is in effect.
ENOENT	The named file does not exist or is the null pathname.
ENOLINK	*path* points to a remote machine and the link to that machine is no longer active.
ENOTDIR	A component of the path prefix is not a directory.

SEE ALSO

sam_stage(3), **sam_stat**(3)

NAME

sam_closecat − Close a catalog handle

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsam [** *library* **...]**

#include "**/opt/SUNWsamfs/include/catalog.h**"

int sam_closecat(int *cat_handle***);**

AVAILABILITY

32-bit programs only

DESCRIPTION

sam_closecat( ) deallocates the catalog handle indicated by *cat_handle*. *cat_handle* is a catalog "handle" obtained from a previous call to **sam_opencat( ).** Deallocation of the catalog handle ends access to the automated library catalog that it referred to, and makes the catalog handle available for return by subsequent calls to **sam_opencat( ).**

RETURN VALUES

Upon successful completion, a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

sam_closecat( ) fails if one or more of the following error conditions are true:

EBADFILE	*cat_handle* is invalid.

SEE ALSO

sam_opencat(3), **sam_getcatalog**(3)

NAME

sam_devstat, sam_ndevstat − Get device status

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsam [** *library* **...]**

#include "**/opt/SUNWsamfs/include/devstat.h**"

int sam_devstat(ushort_t *eq***, struct sam_devstat** ∗*buf***, size_t** *bufsize***);**

int sam_ndevstat(ushort_t *eq***, struct sam_ndevstat** ∗*buf***, size_t** *bufsize***);**

DESCRIPTION

sam_devstat( ) and sam_ndevstat( ) obtain information about the device identified by the equipment number, *eq*.

buf is a pointer to a **sam_devstat( ) or sam_ndevstat( )** structure into which information is placed concerning the device.

bufsize is the length of the user's buffer to which *buf* points. This should be equal to or greater than sizeof(struct sam_devstat) or sizeof(struct sam_ndevstat).

The contents of the structure pointed to by *buf* include the following members for **sam_devstat( )** :

```
u_short   type;          /* Media type */
char      name[32];      /* Device name */
char      vsn[32];       /* VSN of mounted volume, 31 characters */
dstate_t  state;         /* State - on/ro/idle/off/down */
uint_t    status;        /* Device status */
uint_t    space;         /* Space left on device */
uint_t    capacity;      /* Capacity in blocks */
```

The contents of the structure pointed to by *buf* include the following members for **sam_ndevstat( )** :

```
u_short   type;          /* Media type */
char      name[128];     /* Device name */
char      vsn[32];       /* VSN of mounted volume, 31 characters */
dstate_t  state;         /* State - on/ro/idle/off/down */
uint_t    status;        /* Device status */
uint_t    space;         /* Space left on device */
uint_t    capacity;      /* Capacity in blocks */
```

type	The type of the media. Masks for interpreting media type are defined in devstat.h.
name	The name of the device, such as /dev/rmt/3cbn.
vsn	The VSN of the mounted volume, if any. This is a null-terminated string with a maximum of 31 characters.
state	The state of the device. This field is an enumerated type defined in devstat.h.
status	The status of the device. Status bits are defined in the file devstat.h. Also, the library routine **sam_devstr**(3) is available to translate the numeric status field into a character string as displayed in the Sun SAM-FS or SAM-QFS graphical user interfaces and in the Sun SAM-FS or SAM-QFS administrative tool **samu**(1M).
space	The space left on the device, in blocks.
capacity	The capacity of the device, in blocks.

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

sam_devstat( ) or sam_ndevsat( ) fail if one or more of the following are true:

ENODEV	The equipment number supplied is not a valid number, or no device is configured with that equipment number.
EACCES	The program does not have permission to access the Sun SAM-FS or SAM-QFS shared memory segment.
EINVAL	The size of the Sun SAM-FS or SAM-QFS shared memory segment is incorrect. You may need to recompile your program with the current version of Sun SAM-FS or SAM-QFS.
ENOENT	Access to the Sun SAM-FS or SAM-QFS shared memory segment has failed; possibly Sun SAM-FS or SAM-QFS isn't running.
EMFILE	The Sun SAM-FS or SAM-QFS shared memory segment could not be accessed because the number of shared memory segments attached to the calling process would exceed the system-imposed limit.
ENOMEM	The available data space is not large enough to accommodate access to the Sun SAM-FS or SAM-QFS shared memory segment.
E2BIG	For **sam_devstat( )** only. The name field of the device was too long to fit in the name field of the sam_dev structure. Use **sam_ndevstat( ).**

SEE ALSO

samu(1M), **sam_devstr**(3)

NAME

sam_devstr − Translate numeric device status into character string

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsam [** *library* **...]**

#include "**/opt/SUNWsamfs/include/lib.h**"

char *∗***sam_devstr(uint_t** *status***)**

DESCRIPTION

sam_devstr( ) translates an unsigned integer, *status*, into a character status string based on the bits set in *status*, returning a pointer to the character string.

The meanings of the characters in the string returned are as follows:

s---------	Volume is being scanned.
m---------	If a filesystem, the filesystem is mounted.
M---------	The device is in maintenance mode.
-E--------	Device received an unrecoverable error.
-a--------	Device is auditing. If a filesystem, it is being archived.
--l-------	Volume has a label.
---I------	Device is in wait-idle mode.
---A------	Device requires operator attention.
----U-----	Unload has been requested.
-----R----	The device has been requested.
------w---	The device is open for writing.
-------o--	The device is open.
--------P-	The device is positioning (tape only).
--------F-	All storage slots are occupied (robotic devices only).
---------r	Device is ready. If a filesystem, its disk space is being released.
---------R	Device is ready and the volume is read-only.
---------p	Device is present.

RETURN VALUES

A pointer to the character status string is returned.

SEE ALSO

sam_devstat(3)

NAME

sam_getcatalog − Get catalog entries

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsam [** *library* **...]**

#include "**/opt/SUNWsamfs/include/catalog.h**"

int sam_getcatalog(int *cat_handle***, uint** *start_entry***, uint** *end_entry***, struct sam_cat_ent** ∗*buf***, size_t** *entbufsize***);**

AVAILABILITY

32-bit programs only

DESCRIPTION

sam_getcatalog( ) obtains a range of entries from the catalog of an automated library or the historian. The catalog from which entries will be obtained is indicated by *cat_handle*. *cat_handle* is similar to a file descriptor, and is returned from a previous call to **sam_opencat( ).** The range of entries is indicated by *start_entry* and *end_entry*. *start_entry* must be less than or equal to *end_entry*, and must be in the range of valid slot numbers for the automated library (or historian). *buf* is a pointer to an array of **sam_cat_ent** structures, into which the catalog entry information is placed. This array should be large enough to hold the number of entries requested. *entbufsize* is the size of a single **sam_cat_ent** structure, usually indicated by sizeof(struct sam_cat_ent).

The contents of a **sam_cat_ent** structure include the following members:

```
/* catalog table entry */
uint_t    type;             /* OBSOLETE */
uint_t    status;           /* Catalog entry status */
char      media[4];         /* Media type */
char      vsn[32];          /* VSN */
int       access;           /* Count of accesses */
uint_t    capacity;         /* Capacity of volume */
uint_t    space;            /* Space left on volume */
int       ptoc_fwa;         /* First word address of PTOC */
int       reserved[3];      /* Reserved space */
time_t    modification_time;/* Last modification time */
time_t    mount_time;       /* Last mount time */
uchar_t   bar_code[BARCODE_LEN + 1];/* Bar code (zero filled) */
```

RETURN VALUES

Upon successful completion the number of catalog entries obtained is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

sam_getcatalog( ) fails if one or more of the following are true:

EBADF	The catalog handle provided is invalid.
EFAULT	**buf** is an invalid address.
EOVERFLOW	The catalog library software returned more information than was requested.
ENOENT	This is no longer an active catalog.
EINVAL	The buffer size provided is invalid, or **start_entry** or **end_entry** is invalid. (Either **start_entry** is less than zero, **end_entry** is greater than the number of entries in the catalog, or **start_entry** is greater than **end_entry**.)

SEE ALSO

sam_opencat(3), **sam_closecat**(3)

NAME

sam_opencat − Access an automated library's catalog to read entries

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsam [** *library* **...]**

#include "/opt/SUNWsamfs/include/catalog.h"

int sam_opencat(const char ∗*path***, struct sam_cat_tbl** ∗*buf***, size_t** *bufsize***);**

AVAILABILITY

32-bit programs only

DESCRIPTION

sam_opencat( ) initiates access to the automated library catalog pointed to by *path*. The string which *path* points to is limited to 127 characters. It returns a **sam_cat_tbl** structure in the area pointed to by *buf* . *bufsize* is the length of the user's buffer to which *buf* points. This should be equal to or greater than sizeof(struct sam_cat_tbl).

The user may have access to at most MAX_CAT catalogs at any one time.

The contents of a **sam_cat_tbl** structure include the following members:

```
/* catalog table */
time_t    audit_time;    /* Audit time */
int       version;       /* Catalog version number */
int       count;         /* Number of slots */
char      media[4];      /* Media type, if entire catalog is one */
```

Following the call to **sam_opencat( ),** entries in the library catalog are obtained using **sam_getcatalog( ).**

RETURN VALUES

Upon successful completion, a catalog "handle" is returned, which is an integer equal to or greater than zero.

This "handle" is used on subsequent calls to **sam_getcatalog( )** to specify the catalog to access, and is also used by **sam_closecat( )** to deallocate the "handle" and end access to the catalog.

If the call to **sam_opencat( )** fails, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

sam_opencat( ) fails if one or more of the following error conditions are true:

EMFILE	The user already has access to MAX_CAT catalogs , or the process has too many open files.
EINVAL	*bufsize* is set to an invalid value, or either *path* or *buf* is a null pointer.
ER_UNABLE_TO_INIT_CATALOG	This process was unable to initialize the catalog data.
ENOENT	There is no active catalog file with the name given.

SEE ALSO

sam_closecat(3), **sam_getcatalog**(3)

NAME

sam_readrminfo − Get removable-media file status

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -R/opt/SUNWsamfs/lib -lsam [** *library* **...]**

#include "/opt/SUNWsamfs/include/rminfo.h"

int sam_readrminfo(const char **path***, struct sam_rminfo** **buf***, size_t** *bufsize***);**

DESCRIPTION

sam_readrminfo( ) returns information about a removable-media file. The removable media file is pointed to by *path*.

buf is a pointer to a **sam_rminfo( )** structure into which information is placed concerning the file.

bufsize is the length of the user's buffer to which *buf* points. This should be equal to or greater than sizeof(struct sam_rminfo). The maximum number of overflow vsns is 256. The following macro can be used to calculate the size of the sam_rminfo structure for n vsns.

#define SAM_RMINFO_SIZE(n) (sizeof(struct sam_rminfo) + ((n) - 1) ∗ sizeof(struct sam_section))

The contents of the structure pointed to by *buf* is documented in sam_request(3).

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

sam_readrminfo( ) fails if one or more of the following are true:

EACCES	Search permission is denied for a component of the path prefix.
EFAULT	*buf* or *path* points to an illegal address.
EINTR	A signal was caught during the **sam_readrminfo( )** function.
ELOOP	Too many symbolic links were encountered in translating *path*.
EMULTIHOP	Components of *path* require hopping to multiple remote machines and the file system does not allow it.
ENAMETOOLONG	The length of the *path* argument exceeds {***PATH_MAX***}, or the length of a *path* component exceeds {***NAME_MAX***} while {***_POSIX_NO_TRUNC***} is in effect.
ENOENT	The named file does not exist or is the null pathname.
ENOLINK	*path* points to a remote machine and the link to that machine is no longer active.
ENOTDIR	A component of the path prefix is not a directory.
EOVERFLOW	A component is too large to store in the structure pointed to by *buf*.

SEE ALSO

sam_request(3)

NAME

sam_release − Set release attributes on a file or directory

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsam [** *library* **...]**

#include "/opt/SUNWsamfs/include/lib.h"

int sam_release(const char ∗*path***, const char** ∗*ops***);**

DESCRIPTION

sam_release( ) sets release attributes on a file or directory using a Sun SAM-FS or SAM-QFS system call. *path* is the file on which to set the attributes. *ops* is the character string of options, for example: "dn". Individual options are described below.

OPTIONS

a Set the attribute that specifies that a file's disk space be released when at least one archive copy of the file exists. Not valid with **n**.

d Return the release attributes on the file to the default, i.e. the file space is released according to archiver rules or as needed by the releaser. When this option is specified, the attributes are reset to the default. If the partial attribute is reset, all blocks are released for an offline regular file. If it is used, it should be the first character in the string.

i Specifies that the file's disk space be released immediately. This will occur if the file is currently not staging, has at least one archive copy, and has some data in it.

n Specifies that the disk space for this file never be released. Only the super-user can set this attribute on a file.

p Set the *partial* attribute on the file so that, when the file's disk space is released, the first portion of that disk space will be retained. Not valid with sam_release *n* option. Also not valid with either of the checksum *g* (*generate*) or *u* (*use*) attributes. (See **ssum**(1) or **sam_ssum**(3)). The *partial* attribute is mutually exclusive with the sam_stage *n* attribute unless enabled by the data base license key.

If the *partial* attribute is set when the file is offline, the partial blocks are not on the disk and the entire file will be staged if accessed. The **sam_stage p** attribute can be used to stage the partial blocks to the disk.

s n Set the *partial* attribute on the file so that, when the file's disk space is released, the first *n* kilobytes of that disk space will be retained. Not valid with release **−n**. Also not valid with either of the checksum *generate* or *use* attributes (**ssum −g** or **ssum −u**). The *partial* attribute is mutually exclusive with the stage **−n** attribute unless enabled by the data base license key. The minimum value is 8 kbytes and the maximum value is the *maxpartial* value set for this file system by the mount command (see **mount_samfs (1M)**).

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

sam_release( ) fails if one or more of the following are true:

EINVAL	An invalid option was specified, or the file is neither a regular file nor a directory.
EPERM	Not the owner or super-user.
EFAULT	*path* or *ops* points to an illegal address.
EINTR	A signal was caught during the **sam_release( )** function.
ELOOP	Too many symbolic links were encountered in translating *path*.

EMULTIHOP	Components of *path* require hopping to multiple remote machines and the file system does not allow it.
ENAMETOOLONG	The length of the *path* argument exceeds {***PATH_MAX***}, or the length of a *path* component exceeds {***NAME_MAX***} while {***_POSIX_NO_TRUNC***} is in effect.
ENOENT	The named file does not exist or is the null pathname.
ENOLINK	*path* points to a remote machine and the link to that machine is no longer active.
ENOTDIR	A component of the path prefix is not a directory.
ENOTSUP	License does not support option.

SEE ALSO

release(1), **ssum**(1), **sam_ssum**(3)

NAME

sam_request − Create a removable-media file

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -R/opt/SUNWsamfs/lib -lsam [** *library* **...]**

#include "**/opt/SUNWsamfs/include/rminfo.h**"

int sam_request(const char *∗path***, struct sam_rminfo** *∗buf***, size_t** *bufsize***);**

DESCRIPTION

sam_request( ) creates a removable-media file allowing access to either tape or optical disk. The removable media file is pointed to by *path*.

buf is a pointer to a **sam_rminfo( )** structure into which information is placed concerning the file.

bufsize is the length of the user's buffer to which *buf* points. This should be equal to or greater than sizeof(struct sam_rminfo). The maximum number of overflow vsns is 256. The following macro can be used to calculate the size of the sam_rminfo structure for n vsns.

```
#define SAM_RMINFO_SIZE(n) (sizeof(struct sam_rminfo) + ((n) - 1) * sizeof(struct sam_section))
```

The contents of the structure pointed to by *buf* include the following members:

```
                                     /* POSIX rminfo structure. */
ushort_t          flags;             /* File mode (see mknod(2)) */
char              media[4];          /* Media type */
ulong_t           creation_time;     /* Creation time of removable media file */
uint_t            block_size;        /* Block size of file in bytes */
U_longlong_t      position;          /* Position of file on the removable media */
U_longlong_t      required_size;     /* Required size for optical only */

                                     /* optical information only. */
char              file_id[32];       /* File identifier */
int               version;           /* Version number */
char              owner_id[32];      /* Owner identifier */
char              group_id[32];      /* Group identifier */
char              info[160];         /* Information */

                                     /* all media information. */
short             n_vsns;            /* Number of vsns containing file */
short             c_vsn;             /* Current vsn ordinal -- returned */
struct sam_section   section[1];     /* VSN array - n_vsns entries */
```

flags The access flags for the file.

RI_blockio uses block I/O for data transfers. Each request buffer is a block on the device.

RI_bufio uses buffered I/O for data transfers. The block size is defined by **block_size.**

RI_foreign uses block I/O for data transfers. The tape is not written by Sun SAM-FS or SAM-QFS, is bar-coded, write protected and is opened for read access only. This option requires the foreign tape license option.

media The left adjusted string which identifies the media type. See mcf(4).

creation_time

Specifies the time the file was created. This value is not used on entry.

block_size The length of the block in bytes. The block_size is set in the volume labels when the removable media is labeled. This value is returned.

position This field can be set by superuser to specify an initial starting position for the file.

required_size
This size in bytes may be specified. If set, this space must be left on the removable-media.

file_id The file's ID. It is written into the optical file label.

version The version number of the file. It is returned.

owner_id The file's owner ID. It is written into the optical file label.

group_id The file's group ID. It is written into the optical file label.

info The file's optional information field. It is written into the optical file label by .

n_vsns Specified the number of removable media cartridges containing the file.

c_vsn Specifies the current removable media ordinal.

sam_section
Specifies the array of removable media cartridges. The contents of the sam_section structure includes the following members:

```
                              /* POSIX sam_section structure. */
char          vsn[32];        /* Volume serial name */
U_longlong_t  length;         /* Length of this section in bytes */
U_longlong_t  position;       /* Position of this section */
U_longlong_t  offset;         /* Byte offset of this section */
```

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

sam_request() fails if one or more of the following are true:

EACCES	Search permission is denied for a component of the path prefix.
EFAULT	*buf* or *path* points to an illegal address.
EINTR	A signal was caught during the **sam_request()** function.
ELOOP	Too many symbolic links were encountered in translating *path*.
EMULTIHOP	Components of *path* require hopping to multiple remote machines and the file system does not allow it.
ENAMETOOLONG	The length of the *path* argument exceeds {***PATH_MAX***}, or the length of a *path* component exceeds {***NAME_MAX***} while {***_POSIX_NO_TRUNC***} is in effect.
ENOENT	The named file does not exist or is the null pathname.
ENOLINK	*path* points to a remote machine and the link to that machine is no longer active.
ENOTDIR	A component of the path prefix is not a directory.
EOVERFLOW	A component is too large to store in the structure pointed to by *buf*.
ENOTSUP	License does not support foreign tapes.

SEE ALSO

request(1), **mcf**(4)

NAME

sam_restore_copy − Creates an archive copy for a Sun SAM-FS or SAM-QFS file.

SYNOPSIS

cc [*flag* ...] *file* ... **-L/opt/SUNWsamfs/lib -lsamut** [*library* ...]

#include "**/opt/SUNWsamfs/include/stat.h**"

int sam_restore_copy(const char ∗*path*, **int** *copy*, **struct sam_stat** ∗*buf*, **size_t** *bufsize*, **struct sam_section** ∗*vbuf*, **size_t** *vbufsize*);

DESCRIPTION

The **sam_restore_copy()** library routine creates an archive copy for an existing Sun SAM-FS or SAM-QFS file. The file must already exist and the archive copy must not exist. The **sam_restore_copy()** library routine creates an archive copy using information supplied by the user and obtained from a source such as the **archiver.log**. This library routine accepts the following arguments:

path	The path name to the file where the archive copy is being created. It may be an absolute or relative path name, but it must be no longer than **PATH_MAX** (see the **/usr/include/limits.h** file).
copy	The copy number (0, 1, 2, or 3) of the archive copy that is being created.
buf	A **sam_stat** structure. See **sam_stat**(3).
bufsize	The size of the *sam_stat* structure. See **sam_stat**(3).
vbuf	A **sam_section** structure for the array of VSNs if the archive copy overflowed volumes, that is, **n_vsns** > 1. If **n_vsns** = 1, *vbuf* should be set to NULL. See **sam_stat**(3).
vbufsize	The size of the **sam_section** structure. If **n_vsns = 1,** *vbufsize* should be set to 0. See **sam_stat**(3).

The following members in the **sam_stat** structure must exist. All other fields are ignored.

```
ulong_t      st_mode      /∗ File mode (see mknod(2)) ∗/
ulong_t      st_uid       /∗ User ID of the file’s owner ∗/
ulong_t      st_gid       /∗ Group ID of the file’s owner ∗/
u_longlong_t st_size      /∗ File size in bytes ∗/
ulong_t      st_atime     /∗ Time of last access     ∗/
ulong_t      st_ctime     /∗ Time of last file status change ∗/
ulong_t      st_mtime     /∗ Time of last data modification  ∗/
```

The following members in the **sam_copy_s** structure must exist for the copy. All other fields are ignored.

```
u_longlong_t position;     /∗ Position of the file on the media. ∗/
time_t       creation_time; /∗ Time the archive copy was created ∗/
uint_t       offset;       /∗ Location of the copy in the archive file ∗/
short        n_vsns;       /∗ Number of volumes used by the archive ∗/
char         media[4];     /∗ Two character media type. ∗/
char         vsn[32];      /∗ Volume serial name of the media volume ∗/
```

The preceding fields have the following meaning:

position The position of the file recorded on the media.

creation_time
This is the time that the archive was made. If **creation_time** is zero, it is set to the value of **time()**.

offset The location of the copy in the archive file in units of 512 bytes.

n_vsns The number of volumes that this copy spans.

vsn	The volume serial name of the cartridge where the file resides.
media	The two-character media type. For example, the media type for DLT tape is **lt**. See **mcf**(4).

RETURN VALUES

Upon succesful creation of a file, a value of 0 is returned. Otherwise, a negative value is returned and **errno** is set to indicate the error. The possible return values are:

-1	user is not root
-2	invalid copy number
-3	invalid VSN
-4	file does not exist
-5	file open failed
-6	uid and gid do not match those for the existing file
-7	invalid VSN for this copy
-8	multiple copies but vbufsize incorrect
-9	media type invalid
-10	copy restore failed for some other reason

FILES

/etc/opt/SUNWsamfs/mcf
The configuration file for Sun SAM-FS or SAM-QFS.

SEE ALSO

sam_restore_file(3), **sam_stat**(3).

mcf(4).

NAME

sam_restore_file − Creates an offline SAM-FS file.

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsamut [** *library* **...]**

#include "**/opt/SUNWsamfs/include/stat.h**"

int sam_restore_file(const char **path***, struct sam_stat** **buf***, size_t** *bufsize***);**

DESCRIPTION

sam_restore_file( ) creates an offline file in a Sun SAM-FS or SAM-QFS file system. **sam_restore_file( )** creates an offline file using information supplied by the user and obtained from a source such as the archiver.log. The file must not exist.

Note that the program calling this function is responsible for creating all directories in the path *before* calling the function.

path is the pathname to the file to be created. It may be an absolute or relative pathname but must be no longer than PATH_MAX (see the **/usr/include/limits.h file).**

buf is a sam_stat structure (see **sam_stat**(3)).

bufsize is the size of the sam_stat structure (see **sam_stat**(3)).

The following members in the sam_stat structure must exist. All other fields are ignored.

```
ulong_t       st_mode      /* File mode (see mknod(2)) */
ulong_t       st_uid       /* User ID of the file's owner */
ulong_t       st_gid       /* Group ID of the file's owner */
u_longlong_t st_size       /* File size in bytes */
ulong_t       st_atime     /* Time of last access      */
ulong_t       st_ctime     /* Time of last file status change */
ulong_t       st_mtime     /* Time of last data modification  */
```

The following members in the sam_copy_s structure must exist for all copies, if any. All other fields are ignored.

```
char        media[4];    /* Two character media type. */
long_long    position;    /* Position of the file on the media. */
time_t      creation_time; /* Time the archive copy was created */
char        vsn[32];     /* Volume serial name of the media */
```

position The position of the file recorded on the media.

creation_time
This is the time that the archive was made. If **creation_time** is zero, it will be set to the value of **time()**.

vsn The volume serial name of the cartridge where the file resides.

media The two character media type. See mcf(4). For example, the media type for DLT tape is ***lt***.

RETURN VALUES

Upon succesful creation of a file a value of 0 is returned. Otherwise, a negative value is returned and ***errno*** is set to indicate the error. The possible return values are:

-1 user is not root
-2 invalid media type
-3 invalid VSN
-5 file does not exist
-6 restore failed for some other reason

FILES

sam_stat(3)

NAME

sam_segment − Set segment attributes on a file or directory

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsam [** *library* **...]**

#include "/opt/SUNWsamfs/include/lib.h"

int sam_segment(const char ∗*path***, const char** ∗*ops***);**

DESCRIPTION

sam_segment( ) sets segment attributes on a file or directory using a Sun SAM-FS or SAM-QFS system call. If a file is segmented, it is archived and staged in segment size chunks. Segment access is enabled by the data base license key. *path* is the file on which to set the attributes. *ops* is the character string of options, for example: "dl104857600". Individual options are described below.

OPTIONS

d Return the segment file attributes on the file to the default, i.e. reset to the file access instead of segment access. It not possible to reset a file that has already been segmented. When this option is specified, the attributes are reset to the default. If it is used, it should be the first character in the string.

l n Specifies the segment size in units of bytes. The segment_size must be greater than or equal to one megabyte. This segment size is the size at which the file will be segmented for purposes of archiving and staging. An error is returned if the file is greater than the segment size.

s n Specifies the number of segments to stage ahead when staging a segmented file. This means when an offline segment is read, in addition to staging the current segment, the next **n** segments are also staged. The default **n** is zero, which means there is no stage read ahead. The maximum **n** is 255.

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

sam_segment( ) fails if one or more of the following are true:

EINVAL	An invalid option was specified, or the file is neither a regular file nor a directory. The file exceeds the specified segment size.
EPERM	Not the owner or super-user.
EFAULT	*path* or *ops* points to an illegal address.
EINTR	A signal was caught during the **sam_segment( )** function.
ELOOP	Too many symbolic links were encountered in translating *path*.
EMULTIHOP	Components of *path* require hopping to multiple remote machines and the file system does not allow it.
ENAMETOOLONG	The length of the *path* argument exceeds {***PATH_MAX***}, or the length of a *path* component exceeds {***NAME_MAX***} while {***_POSIX_NO_TRUNC***} is in effect.
ENOENT	The named file does not exist or is the null pathname.
ENOLINK	*path* points to a remote machine and the link to that machine is no longer active.
ENOTDIR	A component of the path prefix is not a directory.
ENOTSUP	License does not support segment.

SEE ALSO

segment(1)

NAME

sam_setfa − Set attributes on a file or directory

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsam [** *library* **...]**

#include "**/opt/SUNWsamfs/include/lib.h**"

int sam_setfa(const char **path***, const char** **ops***);**

DESCRIPTION

sam_setfa( ) sets attributes on a file or directory using a Sun SAM-FS or SAM-QFS system call. *path* is the file on which to set the attributes. *ops* is the character string of options, for example: "ds1". Individual options are described below.

OPTIONS

d Return the file attributes on the file to the default, i.e. the stripe is reset to the mount default. When this option is specified, the attributes are reset to the default. If it is used, it should be the first character in the string.

D Specifies the direct I/O attribute be permanently set for this file. This means data is transferred directly between the user's buffer and disk. This attribute should only be set for large block aligned sequential I/O. The default I/O mode is buffered (uses the page cache). Direct I/O will not be used if the file is currently memory mapped. See directio(3C) for Solaris 2.6 and above for more details, however the Sun SAM-FS or SAM-QFS directio attribute is permanent.

g n Specifies the number of the striped group where the file is to be preallocated. *n* is a number 0 .. 127. *n* must be a striped_group defined in the filesystem.

l n Specifies the number of bytes to be preallocated to the file. This can only be applied to a file with no disk blocks assigned. This option is ignored for a directory. If an I/O attempts to extend a preallocated file, the caller will get an ENXIO error. If an attempt is made to preallocate a file with disk blocks assigned, or a segmented file, the caller will get an EINVAL error.

s n Specifies the number of allocation units to be allocated before changing to the next unit. If *n* is 1, this means the file will stripe across all units with 1 disk allocation unit (DAU) allocated per unit. If *n* is 0, this means the file will be allocated on one unit until that unit has no space. The default stripe is specified at mount. (see **mount_samfs (1M)**). Note, EINVAL is returned if the user sets stripe > 0 and mismatched stripe groups exist. Mismatched stripe groups means all striped groups do not have the same number of partitions. Striping across mismatched stripe groups is not allowed.

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

sam_setfa( ) fails if one or more of the following are true:

EINVAL An invalid option was specified, or the file is neither a regular file nor a directory.

EPERM Not the owner or super-user.

EFAULT *path* or *ops* points to an illegal address.

EINTR A signal was caught during the **sam_setfa( )** function.

ELOOP Too many symbolic links were encountered in translating *path*.

EMULTIHOP Components of *path* require hopping to multiple remote machines and the file system does not allow it.

ENAMETOOLONG The length of the *path* argument exceeds {***PATH_MAX***}, or the length of a

	path component exceeds {***NAME_MAX***} while {***_POSIX_NO_TRUNC***} is in effect.
ENOENT	The named file does not exist or is the null pathname.
ENOLINK	*path* points to a remote machine and the link to that machine is no longer active.
ENOTDIR	A component of the path prefix is not a directory.

SEE ALSO

setfa(1), **ssum**(1), **sam_ssum**(3) **sam_advise**(3) **directio**(3C), **mount_samfs**(1M)

NAME

sam_ssum − Set checksum attributes on a file

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsam [** *library* **...]**

#include "/opt/SUNWsamfs/include/lib.h"

int sam_ssum(const char **path***, const char** **ops***);**

DESCRIPTION

sam_ssum( ) sets the checksum attributes on a file using a Sun SAM-FS or SAM-QFS system call. *path* is the file on which to set the attributes. *ops* is the character string of options, for example: "gu". Individual options are described below.

If the *generate* (*g*) attribute is set (−*g*), a 128-bit value is generated when the file is archived. When the file is subsequently staged, the checksum is again generated and is compared against the value generated at archive time if the *use* (−*u*) attribute is set. By default, no checksum value is generated or used when archiving or staging a file.

The *generate* attribute must be set on a file before any archive copy has been made. Likewise, the selected algorithm cannot be changed after an archive copy has been made.

Direct access and partial release are not allowed on a file that has either of the checksum *generate* or *use* attributes set. Also, it is not valid to specify that a file never be archived as well as specify that a checksum be generated and/or used. Therefore, when a direct access, partial release, or archive never attribute is set on a file, attempting to set the checksum *generate* or *use* attribute on the file will result in an error and the attributes will be unchanged. Similarly, when either the checksum *generate* or *use* attribute is set on a file, attempting to set a direct access, partial release, or archive never attribute on the file will result in an error and the attributes will be unchanged.

A file that has the checksum *use* attribute set cannot be memory mapped. The file also must be completely staged to the disk before access is allowed to the file’s data; this means that accessing the first byte of offline data in an archived file that has this attribute set will be slower than accessing the same offline file when it does not have this attribute set.

OPTIONS

d Return the file’s checksum attributes to the default.

g Generate a checksum value for the file when archiving.

u Use the checksum value for the file when staging. The *generate* attribute must have been previously set, or must be set simultaneously.

n *n* is an integer specifying the algorithm to use to generate the 128-bit checksum value. The simple checksum algorithm provided by Sun Microsystems, Inc. is the default if no algorithm is specified but the *generate* attribute is set. *n* may be one of the following:

0 Use no algorithm.

1 Use a simple checksum algorithm that also factors in file length.

128 or higher
Site-specified algorithms.

For example, a valid options string is "gu1", setting the *generate* and *use* attributes, and specifying that the Sun-provided simple checksum algorithm be used to generate the value.

ERRORS

sam_ssum( ) fails if one or more of the following are true:

EINVAL An invalid option was specified, or the file is neither a regular file nor a directory.

EPERM Not the owner or super-user.

EFAULT	*path* or *ops* points to an illegal address.
EINTR	A signal was caught during the **sam_ssum( )** function.
ELOOP	Too many symbolic links were encountered in translating *path*.
ENAMETOOLONG	The length of the *path* argument exceeds {***PATH_MAX***}, or the length of a *path* component exceeds {***NAME_MAX***} while {***_POSIX_NO_TRUNC***} is in effect.
ENOENT	The named file does not exist or is the null pathname.
ENOLINK	*path* points to a remote machine and the link to that machine is no longer active.
ENOTDIR	A component of the path prefix is not a directory.

SEE ALSO

ssum(1), **stage**(1), **release**(1), **archive**(1), **sam_archive**(3), **sam_release**(3), **sam_stage**(3), **sls**(1)

NAME

sam_stage − Set stage attributes on a file

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsam [** *library* **...]**

#include "/opt/SUNWsamfs/include/lib.h"

int sam_stage(const char **path***, const char** **ops***);**

DESCRIPTION

sam_stage( ) sets stage attributes on a file or directory using a Sun SAM-FS or SAM-QFS system call. *path* is the file on which to set the attributes. *ops* is the character string of options, for example: "dn". Individual options are described below.

OPTIONS

a Sets the associative staging attribute on the file or directory. Associative staging is activated when a regular file that has the associative staging attribute set is staged. All files in the same directory that have the associative staging attribute set are staged. If a symbolic link has the associative staging attribute set, the file pointed to by the symbolic link is staged. Not valid with stage never attribute -n.

d Return the stage attributes on the file to the default, i.e. stage automatically as needed. When this option is specified attributes are reset to the default. If it is used, it should be the first character in the string.

i Specifies that the file be staged immediately.

n Specifies that the file never be automatically staged. The file will be read directly from the archive media. The mmap function is not supported if the sam_stage *n* attribute is set. The sam_stage *n* attribute is not valid with the associative staging attribute *a*. The sam_stage *n* attribute is not valid with either of the checksum *g* (*generate*) or *u* (*use*) attributes. (See **ssum**(1) or **sam_ssum**(3)). The sam_stage *n* attribute is mutually exclusive with the sam_release *p* attribute unless enabled by the data base license key.

p Stage the partial blocks back on-line.

s Disable associative staging for the current stage. This is only useful with the **i** option. This causes only the named file to be staged, not other files in the same directory with the associative attribute set.

w Wait for the file to be staged back on-line before completing. Not valid with **d** or **n**.

1 **2 3 4** Stage in the archive copy specified by the option.

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

sam_stage( ) fails if one or more of the following are true:

EINVAL	An invalid option was specified
EPERM	Not the owner or super-user.
ENXIO	No archive copy exists, or the specified archive copy does not exist.
EFAULT	*path* or *ops* points to an illegal address.
EINTR	A signal was caught during the **sam_stage( )** function.
ELOOP	Too many symbolic links were encountered in translating *path*.
EMULTIHOP	Components of *path* require hopping to multiple remote machines and the file system does not allow it.

ENAMETOOLONG	The length of the *path* argument exceeds {***PATH_MAX***}, or the length of a *path* component exceeds {***NAME_MAX***} while {***_POSIX_NO_TRUNC***} is in effect.
ENOENT	The named file does not exist or is the null pathname.
ENOLINK	*path* points to a remote machine and the link to that machine is no longer active.
ENOTDIR	A component of the path prefix is not a directory.
ENOTSUP	License does not support option.

NOTE

If the application writes (see **write**(2)) to a file or the application mmaps (see **mmap(2))** a file with prot set to PROT_WRITE, the file is staged in and the application waits until the stage has completed. The **stage −n** attribute is ignored and the file is completely staged back online.

SEE ALSO

sam-stagealld(1M), **mount_samfs**(1M), **stage**(1), **ssum**(1), **sam_ssum**(3), **mmap**(2), **write**(2)

NAME

sam_stat, sam_lstat − Get file status

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -R/opt/SUNWsamfs/lib -lsam [** *library* **...]**

#include "**/opt/SUNWsamfs/include/stat.h**"

int sam_stat(const char **path*, **struct sam_stat** **buf*, **size_t** *bufsize***);**

int sam_lstat(const char **path*, **struct sam_stat** **buf*, **size_t** *bufsize***);**

DESCRIPTION

sam_stat( ) obtains information about the file pointed to by *path*. Read, write, or execute permission of the named file is not required, but all directories listed in the path name leading to the file must be searchable.

sam_lstat( ) obtains file attributes similar to **sam_stat( )**, except when the named file is a symbolic link; in that case **sam_lstat( )** returns information about the link, while **sam_stat( )** returns information about the file the link references.

buf is a pointer to a **sam_stat( )** structure into which information is placed concerning the file.

bufsize is the length of the user's buffer to which *buf* points. This should be equal to or greater than sizeof(struct sam_stat).

The contents of the structure pointed to by *buf* include the following members:

```
/* POSIX stat structure. */
ulong_t    st_mode;       /* File mode (see mknod(2)) */
ulong_t    st_ino;        /* Inode number */
ulong_t    st_dev;        /* ID of device containing the file*/
ulong_t    st_nlink;      /* Number of links */
ulong_t    st_uid;        /* User ID of the file's owner */
ulong_t    st_gid;        /* Group ID of the file's owner */
u_longlong_t st_size;     /* File size in bytes */
time_t     st_atime;      /* Time of last access */
time_t     st_mtime;      /* Time of last data modification */
time_t     st_ctime;      /* Time of last file status change */

/* Sun SAM-FS information. */
uint_t              attr;   /* File attributes */
time_t              attribute_time;/* Time attributes last changed */
time_t              creation_time;/* Time inode created */
time_t              residence_time;/* Time file changed residence */
struct sam_copy_s copy[MAX_ARCHIVE];
uchar_t cs_algo;      /* Checksum algorithm indicator */
uchar_t flags;              /* Flags:  staging, stage err, etc. */
ulong_t gen;                /* Inode generation number */
```

st_mode The mode of the file as described in **mknod**(2). In addition to the modes described in **mknod**(2), the mode of a file may also be S_IFLNK if the file is a symbolic link. (Note that S_IFLNK may only be returned by **sam_lstat( )**.)

st_ino This field uniquely identifies the file in a given file system. The pair **st_ino** and **st_dev** uniquely identifies regular files.

st_dev This field uniquely identifies the file system that contains the file.

st_nlink This field should be used only by administrative commands.

st_uid	The user ID of the file's owner.
st_gid	The group ID of the file's owner.
st_size	For regular files, this is the address of the end of the file.
st_atime	Time when file data was last accessed. Changed by the following functions: **creat**, **mknod**, **pipe**, **utime**, and **read**.
st_mtime	Time when data was last modified. Changed by the following functions: **creat**, **mknod**, **pipe**, **utime**, and **write**.
st_ctime	Time when file status was last changed. Changed by the following functions: **chmod**, **chown**, **creat**, **link**, **mknod**, **pipe**, **unlink**, **utime**, and **write**.
attr	Attributes assigned to the file by samfs functions and operations.
attribute_time	Time when the samfs attributes last changed. Changed by the following functions: **sam_archive**, **sam_release**, **sam_stage**, and the samfs automatic archive, release and stage operations.
creation_time	Time when the inode was created for the file.
residence_time	Time when the file changed residency. Changed by the release and stage operations.
cs_algo	Indicates which algorithm is used when calculating the data verification value (checksum) for the file (see **ssum**(1)).
flags	Flags containing miscellaneous additional information about the file. The current implementation includes a bit indicating that a stage is pending or is in progress on the file, and a bit indicating that the last attempt to stage the file failed.
gen	The inode generation number.

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

sam_stat() and **sam_lstat()** fail if one or more of the following are true:

EACCES	Search permission is denied for a component of the path prefix.
EFAULT	*buf* or *path* points to an illegal address.
EINTR	A signal was caught during the **sam_stat()** or **sam_lstat()** function.
ELOOP	Too many symbolic links were encountered in translating *path*.
EMULTIHOP	Components of *path* require hopping to multiple remote machines and the file system does not allow it.
ENAMETOOLONG	The length of the *path* argument exceeds {***PATH_MAX***}, or the length of a *path* component exceeds {***NAME_MAX***} while {***_POSIX_NO_TRUNC***} is in effect.
ENOENT	The named file does not exist or is the null pathname.
ENOLINK	*path* points to a remote machine and the link to that machine is no longer active.
ENOTDIR	A component of the path prefix is not a directory.
EOVERFLOW	A component is too large to store in the structure pointed to by *buf*.

SEE ALSO

mknod(2), **stat**(2), **ssum**(1)

NAME

sam_vsn_stat − Get VSN status for an archive copy

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsam [** *library* **...]**

#include </opt/SUNWsamfs/include/stat.h>

int sam_vsn_stat(const char ∗*path***, int** *copy***, struct sam_section** ∗*buf***, size_t** *bufsize***);**

DESCRIPTION

The **sam_vsn_stat( )** function obtains information about the VSNs for the archive copy indicated by *copy* of the file pointed to by *path*. **sam_vsn_stat( )** obtains information about the VSNs for the indicated archive copy when the indicated archive copy uses multiple VSNs.

sam_vsn_stat( ) fails if called to obtain VSN stat information for an archive copy that only uses one VSN. Use the **sam_stat( )** subroutine to determine the number of VSNs used by a given archive copy and to get VSN information for archive copies that only use one VSN.

sam_vsn_stat( ) places VSN information for all of the sections that comprise the overflowed archive copy into *buf*.

Read, write, or execute permission of the named file is not required, but all directories listed in the path name leading to the file must be searchable.

copy is the archive copy number (0, 1, 2 or 3).

buf is a pointer to a **sam_section** structure into which VSN information is placed concerning the file's archive copy.

bufsize is the length of the user's buffer to which *buf* points. **sam_vsn_stat** places VSN information for each overflowed section that comprises the archive copy into *buf*. Hence, bufsize should be at least **sizeof(struct sam_vsn_stat)** ∗ **n_vsns** bytes, where **n_vsns** is the number of VSNs used by the archived copy.

The contents of the structure pointed to by *buf* include the following **struct sam_section** members:

```
char          vsn[32];
u_longlong_t length;
u_longlong_t position;
u_longlong_t offset;
```

vsn	The VSN of the section. This is a null-terminated string with a maximum of 31 characters.
length	The length of the section on the volume.
position	The position of the start of the archive file that contains this section.
offset	The offset of this file on the archive file.

RETURN VALUES

Upon successful completion, a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

sam_vsn_stat( ) fails if one or more of the following are true:

EACCES	Search permission is denied for a component of the path prefix.
EFAULT	*buf* or *path* points to an illegal address.
EINTR	A signal was caught during the **sam_vsn_stat( )** function.
ELOOP	Too many symbolic links were encountered in translating *path*.

EMULTIHOP	Components of *path* require hopping to multiple remote machines and the file system does not allow it.
ENAMETOOLONG	The length of the *path* argument exceeds {***PATH_MAX***}, or the length of a *path* component exceeds {***NAME_MAX***} while {***_POSIX_NO_TRUNC***} is in effect.
ENOENT	The named file does not exist or is the null pathname.
ENOLINK	*path* points to a remote machine and the link to that machine is no longer active.
ENOTDIR	A component of the path prefix is not a directory.
EOVERFLOW	A component is too large to store in the structure pointed to by *buf*.

USAGE

Call **sam_stat** to get the number of VSNs used for the archive copy. The call to **sam_stat** will write the sam stat information in your **struct sam_stat** buffer in the member **copy[copy].n_vsns .** If the archive copy uses only one VSN (the number of VSNs is 1), then your program or script must retrieve the VSN information for the archive copy from the copy member of the **sam_stat** structure that was filled in when your program or script called **sam_stat.** The copy member of the **sam_stat** structure is of type **struct sam_copy_s.**

A **struct sam_copy_s** structure has the following members:

```
u_longlong_t position;
time_t       creation_time;
uint_t       offset;
ushort_t     flags;
short        n_vsns;
char         media[4];
char         vsn[32];
```

position	Location of the archive file
creation_time	Time that the archive copy was created
offset	Location of the copy in the archive file
flags	Sun SAM-FS or Sun SAM-QFS archive copy status flags. These indicate whether the archive copy has been made, is stale, is damaged, etc. See **/opt/SUNWsamfs/include/stat.h** for bit masks which can be applied to these flags to resolve the state and status of the archive copy.
n_vsns	Number of VSNs used by the archived copy. Will be 1 in case of no overflow, will be greater than one if the archive copy overflows volumes.
media	Media type. This is a null-terminated string with a maximum of 3 characters.
vsn	The VSN of the copy. This is a null-terminated string with a maximum of 31 characters.

If the archive copy uses more than one VSN (the number of VSNs is greater than 1), then your program or script must call **sam_vsn_stat** to retrieve the VSN information for all of the sections that comprise the archive copy.

Do not call **sam_vsn_stat** if the archive copy uses only one VSN (does not overflow).

SEE ALSO

sam_stat(3)

NOTES

The Sun SAM-FS and the Sun SAM-QFS file systems permit a maximum of **MAX_VOLUMES** sections per archive copy. Hence, instead of dynamically allocating a buffer of structures, a more efficient method is to to declare a static array with **MAX_VOLUMES** number of elements.

The constant **MAX_VOLUMES** is declared in the following include file: **/opt/SUNWsamfs/include/rminfo.h .**

NAME

sam_archive − Set archive attributes on a file or directory

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsamrpc -l nsl [** *library* **...]**

#include "/opt/SUNWsamfs/include/samrpc.h"

int sam_archive(const char *∗path***, const char** *∗ops***);**

DESCRIPTION

This is the RPC-based version of **sam_archive**(3), allowing archive attributes on a file or directory to be set from a remote machine.

sam_archive(3X) sets archive attributes on a file or directory by sending its request to the Sun SAM-FS or SAM-QFS RPC server, **sam-rpcd**.

A call to **sam_initrpc**(3X) must be issued before calling this routine.

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and **errno** is set to indicate the error.

ERRORS

EDESTADDRREQ	**sam_initrpc** was not successfully called, as required, before making this call.

SEE ALSO

archive(1), **sam_archive**(3), **sam_initrpc**(3X), **sam_closerpc**(3X)

NAME

sam_closerpc − Perform RPC shutdown for Sun SAM-FS or SAM-QFS RPC API library

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsamrpc -l nsl [** *library* **...]**

#include "**/opt/SUNWsamfs/include/samrpc.h**"

int sam_closerpc( )*;*

DESCRIPTION

sam_closerpc() is the shutdown routine for the **libsamrpc** library. It destroys the RPC client handle and deallocates private data structures that were allocated with **sam_initrpc( )**.

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned.

SEE ALSO

sam_initrpc(3X), **sam_archive**(3X), **sam_release**(3X), **sam_stage**(3X), **sam_stat**(3X)

NAME

sam_initrpc − Perform RPC initialization for Sun SAM-FS or SAM-QFS RPC API library

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsamrpc -l nsl [** *library* **...]**

#include "/opt/SUNWsamfs/include/samrpc.h"

int sam_initrpc(char **rpchost***);**

DESCRIPTION

sam_initrpc() is the initialization routine for the **libsamrpc** library. It finds the RPC entry for the Sun SAM-FS or SAM-QFS server and creates an RPC client handle. In essence, this routine sets up the connection to the Sun SAM-FS or SAM-QFS host machine, required for other API calls in the **libsamrpc** library.

rpchost is the hostname of the Sun SAM-FS or SAM-QFS host. If *NULL*, **sam_initrpc( )** will check for an environment variable named *SAMHOST*. If such an environment variable exists, its setting will be taken for the hostname of the Sun SAM-FS or SAM-QFS host, otherwise the built-in default, **samhost**, will be used.

sam_initrpc( ) gets the RPC entry (program number) using the program name **samfs**. This information (the RPC program name and number), and the hostname, is used to set up communication with the Sun SAM-FS or SAM-QFS RPC API server process, **sam-rpcd**, which runs on the Sun SAM-FS or SAM-QFS host machine.

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

sam_initrpc( ) fails if one or more of the following are true:

EADDRNOTAVAIL	No RPC entry for the program name **samfs** could be found.

SEE ALSO

sam_closerpc(3X), **sam_archive**(3X), **sam_release**(3X), **sam_stage**(3X), **sam_stat**(3X)

NAME

sam_release − Set release attributes on a file or directory

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsamrpc -l nsl [** *library* **...]**

#include "**/opt/SUNWsamfs/include/samrpc.h**"

int sam_release(const char **path***, const char** **ops***);**

DESCRIPTION

This is the RPC-based version of **sam_release**(3), allowing release attributes to be set from a remote machine.

sam_release(3X) sets release attributes on a file or directory by sending its request to the Sun SAM-FS or SAM-QFS RPC server, **sam-rpcd**.

A call to **sam_initrpc**(3X) must be issued before calling this routine.

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

EDESTADDRREQ	**sam_initrpc** was not successfully called, as required, before making this call.

SEE ALSO

release(1), **sam_release**(3), **sam_initrpc**(3X), **sam_closerpc**(3X)

NAME

sam_segment − Set segment attributes on a file or directory

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsamrpc -l nsl [** *library* **...]**

#include "**/opt/SUNWsamfs/include/samrpc.h**"

int sam_segment(const char ∗*path***, const char** ∗*ops***);**

DESCRIPTION

This is the RPC-based version of **sam_segment**(3), allowing file attributes to be set from a remote machine.

sam_segment(3X) sets segment attributes on a file or directory by sending its request to the Sun SAM-FS or SAM-QFS RPC server, **rpc.sam**.

A call to **sam_initrpc**(3X) must be issued before calling this routine.

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

EDESTADDRREQ	**sam_initrpc** was not successfully called, as required, before making this call.

SEE ALSO

segment(1), **sam_segment**(3), **sam_initrpc**(3X), **sam_closerpc**(3X)

NAME

sam_setfa − Set attributes on a file or directory

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsamrpc -l nsl [** *library* **...]**

#include "**/opt/SUNWsamfs/include/samrpc.h**"

int sam_setfa(const char **path***, const char** **ops***);**

DESCRIPTION

This is the RPC-based version of **sam_setfa**(3), allowing file attributes to be set from a remote machine.

sam_setfa(3X) sets attributes on a file or directory by sending its request to the Sun SAM-FS or SAM-QFS RPC server, **sam-rpcd**.

A call to **sam_initrpc**(3X) must be issued before calling this routine.

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

EDESTADDRREQ	**sam_initrpc** was not successfully called, as required, before making this call.
EINVAL	A valid filename was not provided.
EPERM	The calling process is not superuser or the owner of the file specified.
EROFS	The file system is a read-only file system.

SEE ALSO

setfa(1), **sam_setfa**(3), **sam_initrpc**(3X), **sam_closerpc**(3X)

NAME

sam_stage − Set stage attributes on a file

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsamrpc -l nsl [** *library* **...]**

#include "**/opt/SUNWsamfs/include/samrpc.h**"

int sam_stage(const char ∗*path***, const char** ∗*ops***);**

DESCRIPTION

This is the RPC-based version of **sam_stage**(3), allowing stage attributes on a file to be set from a remote machine.

sam_stage(3x) sets stage attributes on a file or directory by sending its request to the Sun SAM-FS or SAM-QFS RPC server, **sam-rpcd**.

A call to **sam_initrpc**(3X) must be issued before calling this routine.

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and ***errno*** is set to indicate the error.

ERRORS

EDESTADDRREQ	**sam_initrpc** was not successfully called, as required, before making this call.

SEE ALSO

sam-stagealld(1M), **stage**(1), **sam_stage**(3), **sam_initrpc**(3X), **sam_closerpc**(3X)

NAME

sam_stat, sam_lstat − Get file status over a network connection

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/SUNWsamfs/lib -lsamrpc -l nsl [** *library* **...]**

#include "**/opt/SUNWsamfs/include/stat.h**"
#include "**/opt/SUNWsamfs/include/samrpc.h**"

int sam_stat(const char *∗path*, **struct sam_stat** *∗buf***);**

int sam_lstat(const char *∗path*, **struct sam_stat** *∗buf***);**

DESCRIPTION

These are the RPC-based versions of **sam_stat**(3) and **sam_lstat**(3).

sam_stat(3X) and **sam_lstat**(3X) get file status by sending a request to the Sun SAM-FS or SAM-QFS RPC server, **sam-rpcd**.

If the server machine is different from the local machine, *path* must be an absolute path. If the server machine is the local machine, *path* may be an absolute path or relative to the user's current working directory.

A call to **sam_initrpc**(3X) must be issued before these calls.

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and *errno* is set to indicate the error.

ERRORS

EDESTADDRREQ	**sam_initrpc** was not successfully called, as required, before making this call.
EINVAL	*path* is not an absolute pathname and the server (SAMHOST) machine is not the same as the local machine.

SEE ALSO

sam_stat(3), **sam_lstat**(3), **sam_initrpc**(3X), **sam_closerpc**(3X)

NAME

SamGUI.rsc − Sun SAM-FS and SAM-QFS GUI resource configuration file

SYNOPSIS

/etc/opt/SUNWsamfs/SamGUI.rsc

DESCRIPTION

The GUI resource configuration file contains settings and options for the Sun SAM-FS and SAM-QFS Java GUI tools. The file is read when **libmgr**(1M) is started. If the settings are changed, **libmgr**(1M) must be restarted. A number of default mappings are defined in this file. Lines starting with '#' are comment lines.

The **SamGUI.rsc** file has the following possible setting types: device, media, catalog, mount-requests, and screen.

device: eqId image filename
Set the image for a specific equipment identifier.

device: eqId name
Set the identification text for a given equipment identifier.

device: type image filename
Set the image for an equipment type of device. See **mcf**(4) for a list of equipment types.

media: type image filename
Set the image for a media type. See **mcf**(4) for a list of media types.

catalog: eqId columns fieldlist
Define the column fields to be displayed for the catalog of a given equipment identifier. If the equipment identifier is '∗', then the column definitions will apply to all the catalogs.

Valid column fields are:

Slot, Media, ImportExport, VSN, Barcode, "Access Count", Capacity, Space, "% Full", "Label Date", "Modification Date", "Mount Date"

mount-requests: ∗ columns fieldlist
Define the column fields to be displayed for the mount request table.

Valid column fields are:

Slot, Media, "Process ID", User, VSN, "Wait Time", "Request Count", "Request Time", "Assigned Robot", Priority

screen: mode sizes screenHeight smallFont normalFont
Screen modes are 0, 1 and 2. screenHeight is the height of the screen mode in number of pixels. The screen mode is selected based on screen height. For example, if mode 0 is specified as having a screen height of 480, then mode 0 sizings will be used when when the actual display resolution height is 480 or smaller. Font sizes are font point sizes (like 9, 10, 12, etc.). Icon sizes are the icon height in number of pixels.

SEE ALSO

libmgr(1M), **mcf**(4).

NOTES

Whenever a new version of the Sun SAM-FS and SAM-QFS Java GUI tools are installed, the existing **SamGUI.rsc** file is copied to **SamGUI.rsc.MMDDYY** for reference and backup purposes.

WARNINGS

To ensure clear image displays, try to use transparent GIF images sized to roughly 75x75 pixels.

NAME

archiver.cmd − Sun SAM-FS and SAM-QFS archiver commands file

SYNOPSIS

/etc/opt/SUNWsamfs/archiver.cmd

AVAILABILITY

SUNWsamfs

DESCRIPTION

Commands for controlling the archiver are read from **/etc/opt/SUNWsamfs/archiver.cmd**, which is the archiver commands file. The **archiver.cmd** file must be free from errors, or the archiver does not execute.

Archive Sets and associated media are defined in the archiver command file. Archive Sets are the mechanism that the archiver uses to direct files in a **samfs** file system to media during archiving.

All files in the file system are members of one and only one Archive Set. Characteristics of a file are used to determine Archive Set membership. All files in an Archive Set are copied to the media associated with the Archive Set. The Archive Set name is simply a synonym for a collection of media volumes.

Files are written to the media in an Archive File, which is written in **tar** format. The combination of the Archive Set and the **tar** format results in an operation that is just like using the command **find** to select files for the **tar** command.

In addition, the meta data (directories, the indices of segmented files, and the removable media information), are assigned to an Archive Set to be copied to media. The Archive Set name is the name of the file system. (See **mcf**(4)).

For segmented files, the archivable unit is the segment, not the entire file, so the properties and priorities apply to the segments themselves rather than to the entire file. The index of a segmented file contains no user data and so is assigned to the meta data archive set.

Symbolic links are considered data files for archival purposes.

Each Archive Set may have up to four archive copies defined. The copies provide duplication of files on different media. Copies are selected by the Archive Age of a file.

The archiver command file consists of command lines that are separated into several sections. The commands for the following sections are outlined in this manpage:

General Commands section
Archive Set Assignments section
Archive Copy Definitions section
Archive Set Copy Parameters section
VSN Pool Definitions section
VSN Associations section

Each of these lines consists of one or more fields separated by white space. Leading white space is ignored. Everything after a '#' character is ignored. Lines may be continued by using '\' as the last character on the line.

All parameter settings and Archive Set definitions apply to all file systems (global) until a file system command is encountered. Thereafter, the settings and definitions apply only to the named file system (local). The commands **archmax**, **bufsize**, **drives**, **notify**, and **ovflmin** can only be global and hence are not allowed after the first **fs**= command.

GENERAL COMMANDS SECTION

General commands definitions are identified by the '=' character in the second field or no additional fields.

archmax = *media target_size*

Set the Archive File maximum size for media *media* to *target_size*. Files to be archived will be placed on the media in a single Archive File of length less than or equal to *target_size*. If a single file is greater than *target_size*, then this restriction does not apply.

Sizes appropriate to the media are used by default.

bufsize = *media buffer_size* **[** *lock* **]**

Set the archive buffer size for media *media* to *buffer_size* ∗ *dev*_**blksize** , and (optionally) lock the buffer.

For *media*, specify a valid media type from the list on the **mcf**(4) man page.

For *buffer_size*, specify a number from 2 through 32. The default is 4. This value is multiplied by the *dev*_**blksize** value for the media type, and the resulting buffer size is used. The *dev_blksize* can be specified in the **defaults.conf** file.

The **lock** argument indicates whether or not the archiver should use locked buffers when making archive copies. If **lock** is specified, the archiver sets file locks on the archive buffer in memory for the duration of the **sam-arcopy**(1M) operation. This avoids paging the buffer, and it can provide a performance improvement. The **lock** argument should be specified only on large systems with large amounts of memory. If insufficient memory is present, it can cause an out of memory condition. The **lock** argument is effective only if direct I/O is enabled for the file being archived. By default, **lock** is not specified and the file system sets the locks on all direct I/O buffers, including those for archiving.

This directive can also be specified on an archive set basis by placing the **-bufsize=***buffer_size* and **-lock** directives between **params** and **endparams** directives. For more information on this, see the **-bufsize=***buffer_size* and **-lock** directives mentioned later on this man page.

For more information on *dev*_**blksize**, see the **defaults.conf** man page. For more information on enabling direct I/O, see the **setfa**(1) man page, the **sam_setfa**(3) library routine man page, or the **-O forcedirectio** option on the **mount_samfs**(1M) man page.

drives = *library count*

Set the number of drives to use for archiving on *library* (the library family set name as defined in the mcf) to *count*. The archiver will use only *count* number of drives in *library* to create archive copies. This command prevents the archiver from using all drives in a library and possibly interfering with staging.

The default value is the actual number of drives in the library.

Example:

```
drives = gr50 3
```

fs = *file_system*

Start local definitions for file system *file_system*. All parameter settings and Archive Set definitions will apply only to this file system. This command may be followed by copy definitions to define multiple copies for the meta data.

The defaults are no local definitions and one archive copy for the file system data.

interval = *time*

Set the interval between archive operations to *time*.

The default time is 10 minutes.

logfile = *filename*
Set the name of the archiver log file to *filename*, specified as an absolute pathname. The archiver log file contains a line for each file archived. The line contains information about the file that includes the date, time, media, volume, Archive Set, and the name of the file. Note that it is possible to have a separate log file for each file system (by placing a "logfile =" definition after a "fs =" definition).

The default is no log file.

notify = *filename*
Set the name of the archiver event notification script file to *filename*. This file is executed by the archiver to allow the system administrator to process various events in a site specific fashion. The script is called with a keyword for the first argument. The keywords are: **emerg**, **alert**, **crit**, **err**, **warning**, **notice**, **info**, and **debug**. Additional arguments are described in the default script.

The name of the default script is: */opt/SUNWsamfs/sbin/archiver.sh*.

ovflmin = *media minimum_size*
Set the minimum size of a file which will require more than one volume for media *media* to *minimum_size*. Files to be archived that are smaller than this size will be placed on only a single volume of the media. Files that are larger than this size will be allowed to be written to multiple volumes.

If not specified, volume overflow will not take place.

wait The archiver will not begin archiving until it receives a start command from *archiver*, *samu*, or *samcmd*. This is a mechanism to allow other activities to be performed before archiving begins. The wait may be applied globally or to one or more filesystems.

The default is no waiting.

ARCHIVE SET ASSIGNMENTS SECTION

Archive Set assignments are made by describing the characteristics of the files that should belong to the set. The statements that do this are patterned after the **find**(1) command. The Archive Set name is the first field, followed by the path relative to the Sun SAM-FS or SAM-QFS file system mount point. The path may be enclosed in quotation mark characters, for instance, "**project/gifs**". Within the quoted string, the usual character escapes are allowed, including octal character value.

The remaining fields are either the file characteristics for membership in the set, or controls for the set.

It is possible that the choice of file characteristics for several Archive Sets will result in ambiguous set membership. These situations are resolved in the following manner:

1. The Archive Set with the earliest definition in the command file is chosen.
2. Local definitions for the file system are chosen before the global definitions.

These rules imply that more restrictive Archive Set definitions should be closer to the beginning of the command file.

It is also possible to use the same Archive Set name for several different file characteristics. An example would assign files that are owned by several users into a single Archive Set.

Assigning files to a special archive set called **no_archive** prevents files from being archived. This can be useful for temporary files. The **no_archive** archive set assignment definition must be a local definition to be effective.

The Archive Set assignments may be followed by Archive Copy definitions.

You can specify one or more of the following file characteristics:

-user *uname*
: Include files belonging to user *uname*.

-group *gname*
: Include files belonging to group *gname*.

-minsize *size*
: Include files greater than or equal to *size*. *size* may be specified with the suffices ’b’, ’k’, ’M’, ’G’, and ’T’, for bytes, kilobytes, megabytes, gigabytes, and terabytes.

-maxsize *size*
: Include files less than *size*.

-name *regular_expression*
: Include files with full paths that match *regular_expression*. The Archive Set will restrict the search for matching paths.

-release *attributes*
: Set the release attributes (see **release**(1)) for all files matching the file characteristics on this Archive Set definition. *attributes* may be any of ’a’ always, ’d’ reset to default, ’n’ never, or ’p’ partial.

-stage *attributes*
: Set the stage attributes (see **stage**(1)) for all files matching the file characteristics on this Archive Set definition. *attributes* may be any of ’a’ associative, ’d’ reset to default, or ’n’ never.

Example:

When controlling archiving for a specific file system (using the **fs** = *fsname* command), commands local to the file system level are evaluated before the global commands. Thus, files may be assigned to a local archive set (including the **no_archive** archive set) instead of being assigned to a global archive set. This has implications when setting global archive set assignments such as **no_archive**.

Assume, for example, the following **archiver.cmd** segment:

```
no_archive . -name .*\.o$
fs = samfs1
allfiles   .
    1   10s
fs = samfs2
allfiles   .
    1   10s
```

At first look it appears that the administrator intended not to archive any of the **.o** files in both file systems. However, since the local archive set assignment **allfiles** is evaluated prior to the global archive set assignment **no_archive**, the **.o** files in in both file systems are archived.

To ensure that no **.o** files are archived, the following segment would be used:

```
fs = samfs1
no_archive . -name .*\.o$
allfiles   .
    1   10s
fs = samfs2
no_archive . -name .*\.o$
```

```
        allfiles   .
            1   10s
```

ARCHIVE COPY DEFINITIONS SECTION

The Archive Copy definitions determine when the archive copies are made for the files matching file characteristics. These definitions consist of lines beginning with a digit. This digit is the copy number.

The first fields after the copy number are the option flags as described below:

-release This causes the cache disk space for the files to be released immediately after the copy is made.

-norelease
This flag may be used to prevent automatic release of cache disk space until all copies marked with this flag are made. Using this flag on just one copy will have no effect on automatic release.

Note: **-release** and **-norelease** are mutually exclusive.

The next field is the Archive Age of the file when the archive copy is made. The age may be specified with the suffixes 's', 'm', 'h', 'd', 'w' and 'y', for seconds, minutes, hours, days, weeks and years. The default Archive Age is 4 minutes.

The next field is the Archive Age of the file when the copy is unarchived. The default is to never unarchive the copy.

ARCHIVE SET COPY PARAMETERS SECTION

Archive Set parameters may be set after all Archive Sets are defined. The beginning of this section is noted by the command **params**. The section is ended by the end of the archiver command file or the command **endparams**.

Setting an archive set parameter requires at least three fields: the Archive Set Copy, the parameter name and the parameter value.

The Archive Set Copy is the Archive Set name and copy number separated by '.'.

Parameters may be set for all archive sets by using the pseudo Archive Set Copy **allsets** for the command. All **allsets** commands must occur before those for any actual Archive Set Copies. Parameters set for individual Archive Set Copies override the parameters set by **allsets** commands.

Note: All parameter default values are 0 or **none** unless otherwise specified.

The parameter names and the descriptions are as follows:

-archmax *target_size*
Set the Archive File maximum size for this Archive Set to *target_size*. Files to be archived will be placed on the media in a single Archive File of length less than or equal to *target_size*. If a single file is greater than *target_size*, then this restriction does not apply.

If not specified, the **archmax** value for the media is used.

-bufsize = *buffer_size*
Set the archive buffer size to *buffer_size* ∗ *dev*_**blksize.** The default *buffer_size* is 4. Valid values are 2 through 32.

If not specified, the default buffer size value for the media is used. This directive can also be specified as a global directive. For more information on specifying an archive buffer size, see the **bufsize**=*media buffer_size* [**lock**] directive described on this man page in the GENERAL COMMANDS section.

-disk_archive *diskvol*

Defines a disk archive set. The archiver uses this parameter to maintain the file system hierarchy of the data as it is written to the archive disk's mount point. For more information on disk archiving, see the *Sun SAM-FS and Sun SAM-QFS Storage and Archive Management Guide*.

Only the priority and recycling archive set parameters are allowed when using **-disk_archive**.

-drivemin *min_size*
Set the multiple drives minimum size for this Archive Set to *min_size*. When the **-drives** parameter is selected, multiple drives will be used only if more than *min_size* data is to be archived at once. The number of drives to be used in parallel will be the lesser of *total_size / min_size* and the number of drives specified by **-drives**.

The default value is **archmax**.

-drives *number*
Set the maximum *number* of drives to use when writing the archive images for this Archive Set Copy to removable media.

Segments are striped across the specified *number* of drives. The segments are separated into *number* archive files.

Example:

```
      set_name.3 -drives 3
```

Allows the archiver to use up to 3 drives for archiving files in the archive set named *set_name.3*.

If not specified, one drive will be used.

-fillvsns The default action of the archiver is to utilize all volumes associated with an Archive Set for archiving. When a group of files is to be archived at the same time, a volume with enough space for all the files will be selected for use. This action may cause volumes to not be filled to capacity.

Selecting this parameter causes the archiver to attempt to fill volumes by separating the group of files into smaller groups.

-join *method*
Organize the files in this Archive Set Copy so that each Archive File contains only files that match according to *method*.

For selecting the desired organization, *method* may be:

none Organize the Archive File so that the size of the Archive File is less than **archmax**. If the size of a single file is greater than **archmax**, it will be the only file in the Archive File.

path Organize the Archive File so that the files included all have the same directory paths.

Note: The use of **-join path** writes all data files to the same archive file. Therefore, many directories with a few small files creates many archive files. These small archive files can performance of high speed tape drives. Also, **-join path** places all files from the same directory on a single volume. It is possible that a group of files might not fit on any available volume, including an empty volume. In such a case, files are never archived. For most applications, using the **-sort path** directive is more efficient than using the **-join path** directive, if the more restrictive operation of **-join path** is not a requirement.

-lock Lock the archive copy buffer for the duration of the **sam-arcopy**(1M) operation. The **-lock** directive is effective only if direct I/O is enabled for the file being archived. If not specified, the file system controls the locks on the archive copy buffer. By default, this directive is disabled.

This directive can also be specified as a global directive. For more information on controlling the archive buffer locks, see the **bufsize=***media buffer_size* [**lock**] directive described on this man page in the GENERAL COMMANDS section.

-offline_copy *method*
This parameter specifies the method to be used for archiving files that are offline at the time archival is to be made.

For selecting the desired offline file archiving method, *method* may be:

none Files are staged as needed for each archive file before copying to the archive volume.

direct Direct copy. Copy files directly from the offline volume to the archive volume without using the cache. Source volume and destination volume are different and two drives are available. For best performance in this mode, you should increase the filesystem mount parameter "stage_n_window" from its default of 256k.

stageahead
Stage the next archive file while each archive file is written to the destination. Two drives are available and room is available on cache for all files in one archive file.

stageall Stage all files before archiving. Use only one drive, and room is available on cache for all files.

-ovflmin *minimum_size*
Set the minimum size of a file that will require more than one volume in this Archive Set to *minimum_size*. Files to be archived that are smaller than this size will be placed on only a single volume of the media. Files that are this size or larger will be allowed to overflow one volume to at least one additional volume.

If not specified, the **ovflmin** value for the media will be used.

The following priority *value*s are floating-point numbers.

-priority age *value*
Set the "Archive Age" property multiplier for files in this Archive Set to *value*.

-priority archive_immediate *value*
Set the "Archive immediate" property multiplier for files in this Archive Set to *value*.

-priority archive_overflow *value*
Set the "Multiple archive volumes" property multiplier for files in this Archive Set to *value*.

-priority archive_loaded *value*
Set the "Archive volume loaded" property multiplier for files in this Archive Set to *value*.

-priority copy1 *value*
Set the "Copy 1" property multiplier for files in this Archive Set to *value*.

-priority copy2 *value*
Set the "Copy 2" property multiplier for files in this Archive Set to *value*.

-priority copy3 *value*
Set the "Copy 3" property multiplier for files in this Archive Set to *value*.

-priority copy4 *value*
Set the "Copy 4" property multiplier for files in this Archive Set to *value*.

-priority copies *value*
Set the "Copies made" property multiplier for files in this Archive Set to *value*.

-priority offline *value*
Set the "File off line" property multiplier for files in this Archive Set to *value*.

-priority queuewait *value*

Set the "Queue wait" property multiplier for files in this Archive Set to *value*.

-priority rearchive *value*
Set the "Rearchive" property multiplier for files in this Archive Set to *value*.

-priority reqrelease *value*
Set the "Required for release" property multiplier for files in this Archive Set to *value*.

-priority size *value*
Set the "File size" property multiplier for files in this Archive Set to *value*.

-priority stage_loaded *value*
Set the "Stage volume loaded" property multiplier for files in this Archive Set to *value*.

-priority stage_overflow *value*
Set the "Multiple stage volumes" property multiplier for files in this Archive Set to *value*.

-reserve [set | dir | user | group | fs]
This parameter specifies that the volumes used for archiving files in this Archive Set are "reserved". If this option is not used, Archive Sets are mixed on the media specified. This option specifies that each archive set has unique volumes. A so-called "ReserveName" is assigned to volumes as they are selected for use by the Archive Set. The ReserveName has three components: Archive Set, Owner, and filesystem. The keyword *set* activates the Archive Set. The keyword *fs* activates the filesystem component.

The keywords **dir**, **user**, and **group** activate the Owner component. These three are mutually exclusive. The Owner component is defined by the file being archived.

The *dir* keyword uses the directory path component imediately following the path specification of the Archive Set description.

The *user* keyword selects the user name associated with the file.

The *group* keyword selects the group name associated with the file.

-sort *method*
Files in the Archive Set may be sorted according to *method* before being archived. The effect of the sort is keep files together according to the property associated with the method.

For selecting the sort, *method* may be one of the following:

age	Sort each Archive File by ascending Archive Age.
none	No sorting of the Archive File is performed. Files will be archived in the order encountered on the file system.
path	Sort each Archive File by the full pathname of the file. This method will keep files in the same directories together on the archive media.
priority	Sort each Archive File by descending archive priority. This method will cause higher priority files to be archived first.
size	Sort each Archive File by ascending file size.

If not specified, no sorting will be performed.

-tapenonstop
When files are archived to tape, the default writing mechanism closes the removable media tape file in between each Archive File. This action causes the tape subsystem to write a TapeMark followed by an EOF1 label and two TapeMarks. Before another Archive File can be written, the tape must be positioned backwards over the EOF1 label.

Using the **tapenonstop** parameter causes the archiver to not close the removable media tape file between each Archive File, and write a Tape Mark to separate the Archive Files. This speeds writing Archive Files to tape. The tape cannot be unloaded in between Archive Files.

The following archive set parameters control recycling by archive set. If none of the following parameters are set for an archive set and the name of the archive set is not specified on the recycler's command line, the archive set will not be recycled. Volumes which comprise that archive set (unless also assigned to other archive sets) could be recycled as part of recycling the library which contains them.

-recycle_dataquantity *size*
: This option sets a limit of *size* bytes on the amount of data the recycler will schedule for rearchiving so as to clear volumes of useful data. Note that the actual number of volumes selected for recycling may also be dependant on the **-recycle_vsncount** parameter. The default is 1 gigabyte (1G).

-recycle_hwm *percent*
: This option sets the high water mark (hwm) for the archive set. The hwm is expressed as a percentage of the total capacity of the volumes associated with the archive set. When the utilization of those volumes exceeds *percent*, the recycler will begin to recycle the archive set. The default is 95%.

-recycle_ignore
: This option inhibits the recycler from recycling this archive set. All recycling processing occurs as usual, except any media selected to recycle are not marked "recycle". This allows the recycler's choice of media to recycle to be observed, without actually recycling any media.

-recycle_mailaddr *mail-address*
: This option specifies an email address to which informational messages should be sent when this archive set is recycled. The default is not to send any mail.

-recycle_mingain *percent*
: This option limits selection of volumes for recycling to those which would increase their free space by *percent* or more. Volumes not meeting the mingain parameter are not recycled. The default is 50%.

-recycle_vsncount *count*
: This option sets a limit of *count* on the number of volumes the recycler will schedule for rearchiving so as to clear volumes of useful data. Note that the actual number of volumes selected for recycling may also be dependant on the -recycle_dataquantity parameter. The default is 1.

VSN POOL DEFINITIONS SECTION

Collections of volumes may be defined in this section. The beginning of the section is noted by the command **vsnpools**. The section is ended by the end of the archiver command file or the command **endvsnpools**.

A VSN pool definition requires at least three fields: the pool name, the media type, and at least one VSN.

The media type is the two character mnemonic as described in the **mcf**(4) manpage.

VSNs are regular expressions as defined in **regcmp**(3G).

VSN ASSOCIATIONS SECTION

VSN associations are defined after all archive sets are defined. The beginning of the section is noted by the command **vsns**. The section is ended by the end of the archiver command file or the command **endvsns**.

A VSN association requires at least three fields: the Archive Set Copy, the media type, and at least one VSN.

The Archive Set Copy is the Archive Set name and copy number separated by '.'.

The media type is the two character mnemonic as described in the **mcf**(4) manpage.

VSNs are regular expressions as defined in **regcmp**(3G). or VSN pool denoted by the option name **-pool** *vsn_pool_name*

Each VSN on a vsns line is used without leading or trailing spaces as input to **regcmp**(3G). The compiled form is saved with the Archive Set Copy definition. When a volume is needed for an Archive Set Copy, each VSN of each library or manual drive that has sufficient space and is allowed to be used for archives, is used as the "subject" argument to **regex**(3G). The archive set copy vsn expressions are used as the "re" argument to **regex**(3G). If **regex**(3G) returns with a successful match, the volume is used for the archive set copy.

Example:

```
set_name.3 mo optic.*
```

Assigns all files in *set_name.3* to the *mo* media with VSNs beginning with *optic*.

SEE ALSO

archiver(1M), **archiver.sh**(4), **diskvols.conf**(4), **mcf**(4), **regcmp**(3G), **release**(1), **stage**(1), **sam-archiverd**(1M), **sam-arcopy**(1M), **sam-arfind**(1M), **sam-recycler**(1M)

NAME

archiver.sh − Sun SAM-FS or SAM-QFS archiver exception notification script

SYNOPSIS

/opt/SUNWsamfs/sbin/archiver.sh

AVAILABILITY

SUNWsamfs

DESCRIPTION

/opt/SUNWsamfs/sbin/archiver.sh is a script which is executed by the archiver when it encounters abnormal or exceptional events. It is intended as a means to allow site specific control of these events.

The archiver executes **/opt/SUNWsamfs/sbin/archiver.sh** with three arguments.

The first argument is a keyword identifying the severity and **syslog** level of the event. The keywords are: **emerg**, **alert**, **crit**, **err**, **warning**, **notice**, **info ,** and **debug**.

The second argument is the message number as found in the message catalog.

The third argument is the text of the translated message string.

DEFAULT FILE

As released, **/opt/SUNWsamfs/sbin/archiver.sh** is a script which will log the event to **syslog** using the **/usr/bin/logger** command.

The **emerg**, **alert**, **crit**,**and err** keywords generate mail to the root account, echoing the message string. Additionally, two **warning** messages have special behaviour.

Two directories are created in **/var/opt/SUNWsamfs/archiver** to handle the exceptions of no space or no VSNs for an archive set, named NoSpace and NoVsns. These directories will be populated with zero-sized files with the same name as their respective archive sets. It is the administrator's responsibility to remove these files when the specific condition no longer exists.

EXAMPLE

See **/opt/SUNWsamfs/examples/archiver.sh**.

SEE ALSO

archiver(1M), **archiver.cmd**(4)

NOTES

An existing copy of **/opt/SUNWsamfs/sbin/archiver.sh** will be preserved when the Sun SAM-FS or SAM-QFS software is upgraded.

NAME

defaults.conf − Set default values for Sun SAM-FS and SAM-QFS software

SYNOPSIS

/etc/opt/SUNWsamfs/defaults.conf

AVAILABILITY

SUNWqfs
SUNWsamfs

DESCRIPTION

The defaults configuration file allows the site to set certain default values within the Sun SAM-FS and SAM-QFS environments. The **defaults.conf** file is read when **sam−fsd** is started. It may be changed at any time while **sam−fsd** is running. The changes will take place when **sam−fsd** is restarted, or sent the signal SIGHUP. Temporary changes to the environment values can be made using the **samset**(1M) command.

The **defaults.conf** file consists of directive lines that are separated into two sections, the environment variable section and the trace file control section.

Environment variables.

The commands for the environment section of the file consists of a list of **keyword =** *value* pairs that set site-definable defaults. All **keyword** and *value* entries are case-sensitive and must be entered as shown. Values can be either unquoted strings (if string values are expected) or integers in decimal (123), octal, (0123) or hex (0x123) format.

The keywords and their expected arguments are as follows:

attended = yes | no

If **attended = yes**, it is assumed that an operator is available to mount media that is not flagged as unavailable by the historian; the default is **yes**. If **attended = no**, any request for media known to the historian is rejected unless it is already mounted.

debug = *options*

Sets the default for the debug flags used by the Sun SAM-FS and SAM-QFS daemons for logging messages. For *options*, specify a space-separated list of debug options from the list of possible options described on the **samset**(1M) man page. The default is **logging**.

devlog = *eq_ord* [*event ...*]

Manipulates the device log event flags for the device specified by **Equipment Ordinal** *eq_ord*. The *eq_ord* must be either the keyword **all** (to specify all devices) or must match an **Equipment Ordinal** from the **mcf** file.

The device log event flags control the events that get written to the device log files. For the list of possible *event* arguments, see the **samset**(1M) man page. To specify more than one *event*, separate the *event*s in the list with space characters. The default is **err retry syserr date**.

*dev*_**blksize =** *size*

Specifies the default block size for tapes of type *dev*. For *size*, specify **16**, **32**, **64**, **128**, **256**, **512**, **1024**, or **2048**. The *size* value is multiplied by 1024 to arrive at the actual block size.

For information on supported *dev* arguments and for information on the default released block sizes for various media, see the **mcf**(4) man page.

The default is used when no *size* is specified or during automatic labeling when **labels = barcodes** has been specified. For information on how the default can be overridden when manually labeling a tape, see the **tplabel**(1M) man page.

*dev*_**delay** = *seconds*
Specifies the dismount time, in seconds, for device type *dev*. After a cartridge is loaded onto this device type, this time must elapse before the cartridge unloaded and another cartridge is loaded. By default, *dev*_**delay = 30**. For information on supported *dev* arguments, see the **mcf**(4) man page.

*dev*_**position_timeout** = *seconds*
Specifies the timeout value, in seconds, to be used during tape positioning for device type *dev*. During most tape positioning command processing (such as locate and space) this is the maximum amount of time to wait for the command to complete. For information on the default values, see the example file (**/opt/SUNWsamfs/examples/defaults.conf**) supplied with your software. Any device not in the example file defaults to **1800** seconds. For information on supported *dev* arguments, see the **mcf**(4) man page.

*dev*_**unload** = *seconds*
Specifies the unload wait time, in seconds, for device type *dev*. This is the amount of time that the library daemons wait after the device driver returns from a SCSI unload command. This interval gives the library time to eject the media, open the door, and perform other actions before the daemon commands the library to remove the media. The *seconds* specified should the longest time needed for the worst-case library configured. For information on the default values, see the example file (**/opt/SUNWsamfs/examples/defaults.conf**) supplied with your software. Any device not in the example file defaults to **0** seconds. For information on supported *dev* arguments, see the **mcf**(4) man page.

dio_min_file_size = *size*
Stages files larger *size* megabytes with direct I/O. For *size*, enter an integer number such that **0** ≤ *size* ≤ **2147483647** megabytes. If *size* = **0**, paged I/O is used to stage all files. The default *size* is **100** megabytes.

exported_media = *value*
Declares exported media to be available or unavailable to the historian, as follows:

- If **exported_media = available**, media exported from a library is considered to be available in the historian. The default is **available**.
- If **exported_media = unavailable**, media exported from a library is considered to be unavailable in the historian. Cartridges with this characteristic are not used by the archiver, stager, or other Sun SAM-FS or SAM-QFS tools. They are considered to reside outside of the Sun SAM-FS or SAM-QFS environment. This might be used, for example, for cartridges to be transported to offsite storage.

For more information, see the **historian**(7) man page.

idle_unload = *seconds*
Specifies the time, in *seconds*, that a library-controlled device can be idle before the media in that device is unloaded. Specifying **idle_unload = 0** disables this feature. By default, **idle_unload = 600**, which is 10 minutes.

shared_unload = *seconds*
Specifies the time, in *seconds*, that a shared library-controlled device can be idle before the media in that device is unloaded. A device is shared if it is used by more than one Sun SAM-FS or SAM-QFS server. For more information on shared devices see the **sony**(7), the **ibm3494**(7), or the **stk**(7) man page. Specifying **shared_unload = 0** disables this feature. By default, **shared_unload = 60**, which is 60 seconds.

inodes This keyword is still accepted for backward compatibility, but it has no effect. For more information, see the **samfs.cmd**(4) man page.

labels = *mode*

For tape libraries with bar code label readers, this keyword sets the tape label equal to the first or the last characters of the bar code label (uppercased). For *mode*, specify either **barcodes**, **barcodes_low**, or **read**, as follows:

- If **labels = barcodes**, the first part of the bar code is used as the label. Default.
- If **labels = barcodes_low**, the last part of bar code is used as the label.
- If **labels = read**, the label is read from the tape. If you wish to have the labels different from the barcodes on a library with a bar code label reader, you must set **labels = read**.

When **labels** is set to **barcodes** or **barcodes_low**, a label is written to the tape before the write is enabled for any tape mounted for a write operation that is write enabled, unlabeled and has a readable bar code label.

log = *facility*

Sets the facility code used for issuing log messages. For information on the accepted *facility* types, see the **syslog**(3) man page. The default is **LOG_LOCAL7**.

oper_privileges = *privilege*

Adds privileges to the operator group. By default, members of the operator group do not have the privileges to perform the following tasks: media labeling, performing storage element movement actions, submitting full audit requests, changing a device state (except to **ON** a device), clearing mount requests, changing display options, and turning off automatic display refresh. To grant the privileges needed to perform those actions, specify one or more of the following *privilege* arguments.

privilege	**Result**
all	Grants all privileges in this list.
clear	Grants the ability to clear cartridge load requests.
fullaudit	Grants the ability to perform a full library audit.
label	Allows cartridge labeling.
options	Grants the ability to change display options, such as the cartridge displayed by **previewtool**(1M).
refresh	Grants the ability to turn off automatic display refresh.
slot	Allows mounting, unloading, and moving cartridges within a library.
state	Grants the ability to change the device state. Operator group members can **ON** devices regardless of this setting.

Use a space character between *privilege* arguments if specifying more than one.

operator = *group*

Specifies the name of the group that to be granted operational privileges within the GUI tools (**previewtool**, **devicetool**, and **robottool**) and command queues. Only one *group* name can be specified. Users must have their effective group IDs set to *group* in order to gain operational privileges.

For more information on the GUI tools, see the **previewtool**(1M), **devicetool**(1M), and **robottool**(1M) man pages.

optical = *media_type*

Sets the default media type to *media_type* when a generic optical disk (**od**) is requested. A string value is expected. For information on the accepted media types, see the **mcf**(4) man page. The default is **mo**.

previews = *requests*
: Sets the number of outstanding mount requests. Care should be taken when changing this value. Each entry takes about 500 bytes of shared memory. By default, **previews = 100**.

samrpc = on | off
: Invokes the RPC API server process. If **samrpc = on**, the RPC API server process, **sam-rpcd**, is automatically started when Sun SAM-FS or SAM-QFS is started. By default, **samrpc = off**, so **sam-rpcd** is not started automatically.

stale_time = *minutes*
: Sends an error to any request for removable media that has waited for *minutes* number of minutes. Setting **stale_time = 0**, disables this function. By default, **stale_time = 30**.

tape = *media_type*
: Sets the default media type to *media_type* when a generic tape (**tp**) is requested. A string value is expected. For information on the accepted media types, see the **mcf**(4) man page. The default is **lt**.

timeout = *seconds*
: Sets the timeout interval, in seconds, for direct access removable media. If a process fails to issue an I/O request to the device within this time, the device is removed from job assignment and the process receives an **ETIME** when the next I/O to the device commences. Specifying **timeout = 0** disables this timeout. By default, **timeout = 600**.

tp_mode = *mode*
: Specifies the mode set for tape drive device nodes when not under control of the Sun SAM-FS or SAM-QFS software. For information, see the **chmod**(2) man page. When the Sun SAM-FS or SAM-QFS software is controlling the drive, the mode bits are **0660**.

Trace file controls.

The daemon trace files are controlled by directives in the trace file section. This section begins with the **trace** directive, and ends with the **endtrace** directive. The trace file control directives are of the form:

daemon_name.*variable_name* = *value*
daemon_name = **on**
daemon_name = **off**

daemon_name may be one of: **sam-archiverd**, **sam-catserverd**, **sam-fsd**, **sam-ftpd**, **sam-recycler**, **sam-sharefsd**, **sam-stagerd**, and **all**.

If *daemon_name* is **all**, then the *variable_name* is set to *value* for all daemons.

For the form: *daemon_name* **= on** the trace file controls will be set to the pre-defined values for *daemon_name*.

In particular, using only the directive
all = on
will enable tracing for all daemons. The trace files will be written to files named for the daemons (e.g. 'sam-ftpd') in the subdirectory '/var/opt/SUNWsamfs/trace'.

For the form: *daemon_name* **= off** tracing will be turned off for *daemon_name*.

The *variable_name* is one of: **file**, **options**, **age**, or **size**.

daemon_name.**file** = *file_name*
: set the name of the trace file to *file_name*. The default is no trace file.

: If the *daemon_name* is **all**, then *file_name* is the name of the trace subdirectory that will contain the daemon tracefiles. *file_name* must be absolute in this case. The default subdirectory is **/var/opt/SUNWsamfs/trace**.

If *file_name* is relative (no leading '/'), the file name will be made relative to the trace base directory. If the file does not exist, **sam-fsd** will create it.

daemon_name.**options** = *option_list*

Set the trace file options to *option_list*. *option_list* is a space separated list of trace options. A trace option is an event to trace, or an element to include in the trace line. To exclude an *option*, prefix the *option* with a '-'.

For selecting events, *option* may be one or more of:

none	Clear all event types.
all	Set event types for tracing the most interesting events. These are: **cust err fatal ipc misc proc ftp**.
alloc	Memory allocations.
cust	Customer notification syslog or notify file messages.
err	Non-fatal program errors.
fatal	Fatal syslog messages.
files	File actions.
ftp	File transfer events.
ipc	Inter process communication.
misc	Miscellaneous.
oprmsg	Operator messages.
proc	Process initiation and completion.
queue	Archiver queue contents when changed.

For selecting message elements, *option* may be one or more of:

date	Include the date in message (the time is always included).
module	Include source file name and line number in message.
type	Include event type in message.

The pre-defined events are: **cust**, **err**, **fatal**, **misc**, **proc**, **ftp**. The message elements program[pid] and time are always included and can't be deselected.

daemon_name.**age** = *age*

Set the time between trace file rotations to *age*. *age may be specified with the* seconds, minutes, hours, days, weeks and years. **sam-fsd** can perform trace file "rotations" using the script **/opt/SUNWsamfs/sbin/trace_rotate.sh**. Trace file rotations are useful to control the size of trace files.

daemon_name.**size** = *size*

Set the trace file *size* at which trace file rotations will be performed. *size* may be specified with the suffices 'b', 'k', 'M', 'G', and 'T', for bytes, kilobytes, megabytes, gigabytes, and terabytes.

EXAMPLES

Here is a sample **defaults.conf** configuration file.

```
optical = mo
debug = logging debug timing
tape = lt
log = LOG_LOCAL7
timeout = 30
idle_unload = 600
```

```
tp_mode = 0666
rc_delay = 10
cy_delay = 10
ml_delay = 10
hp_delay = 10
ds_delay = 10
lt_unload = 7
st_unload = 15
lt_blksize = 16
operator = sam
oper_privileges = label slot
trace
all = on          # Turn on tracing for all daemons
sam-archiverd.size = 10M # Rotate archiver trace file after 10 megabytes
sam-ftpd.file = /tmp/sam-ftpd.trace  # change file name for sam-ftp daemon
sam-recycler = off  # Turn off tracing for sam-recycler daemon
endtrace
```

FILES

/opt/SUNWsamfs/examples/defaults.conf
Contains an example of a **defaults.conf** file.

SEE ALSO

request(1).

devicetool(1M), **previewtool**(1M), **robottool**(1M), **samset**(1M), **sam-fsd**(1M), **tplabel**(1M).

chmod(2).

syslog(3).

mcf(4), **samfs.cmd**(4), **trace_rotate.sh**(4).

historian(7)

NAME

dev_down.sh − Sun SAM-FS or SAM-QFS device down notification script

SYNOPSIS

/opt/SUNWsamfs/sbin/dev_down.sh

AVAILABILITY

SUNWsamfs

DESCRIPTION

/opt/SUNWsamfs/sbin/dev_down.sh is a script which is executed by sam-robotsd(1M) when a device is marked "down" or "off".

DEFAULT FILE

As released, **/opt/SUNWsamfs/examples/dev_down.sh** contains a script which will mail root with the relevant information.

EXAMPLE

The following is an example **/opt/SUNWsamfs/sbin/dev_down.sh** file:

```
#!/bin/sh
#

#   /opt/SUNWsamfs/sbin/dev_down.sh - Take action in the event
#       a device is marked down by the Sun SAM-FS or SAM-QFS software.
#
#   arguments:  $5: device identifier
#
# Change the email address on the following line to send
# email to the appropriate recipient.
/usr/ucb/mail -s "Sun SAM-FS Device downed" root <<EOF
`date`
Sun SAM-FS has marked the device $5,
as down or off.
EOF
```

The example sends email to root to report that a device has been marked "down" or "off".

SEE ALSO

sam-robotsd(1M)

NAME

diskvols.conf − Defines disk archive volumes for Sun SAM-FS or Sun SAM-QFS environments

SYNOPSIS

/etc/opt/SUNWsamfs/diskvols.conf

AVAILABILITY

SUNWsamfs

DESCRIPTION

A Sun SAM-FS or Sun SAM-QFS file can have one or more of its archive copies written to a disk archive resource. A *disk volume* that represents the resource is stored in the inode of the archived file.

The disk volume configuration file, **diskvols.conf**, defines the mapping between a disk volume and the corresponding resource. The **sam-fsd** daemon reads the **diskvols.conf** file when the **sam−fsd** daemon is started. The **diskvols.conf** file can be changed at any time while the **sam−fsd** daemon is running. The changes take effect when the **sam−fsd** daemon is restarted or sent the signal SIGHUP.

The mappings are specified one per line. Each line consists of two fields separated by white space. Leading white space is ignored. Everything after a pound character (#) is ignored. Lines can be continued by using a backslash character (\) as the last character on the line. The syntax for this line is as follows:

disk_volume resource

where:

disk_volume
An alphanumeric string. The string can contain up to 31 characters.

resource A resource specification in one of the following formats:

pathname	This format contains the path name of the disk archive directory on the local host.
[*host*:]*pathname*	This format specifies the *host* as the name of the disk archive server and *pathname* as the path name of the disk archive directory on that host.

CLIENT DEFINITIONS SECTION

The **clients** and **endclients** directives delimit this section of the **diskvols.conf** file.

The client definitions section defines the trusted client systems. After the disk archiving server accepts a client connection, it verifies that the socket address belongs to a host in the trusted client definitions section. If not, the connection is refused.

EXAMPLES

This example shows two **diskvols.conf** files.

File 1 is a **diskvols.conf** file on client system earth that defines the following:

- There is one volume serial name (VSN) for a local disk archive.
- There are two remote VSNs. Remote VSN **remote1** resides in **/quidditch** on the remote server **gryffindor**, and remote VSN **remote2** resides in **/quidditch** on remote server **ravenclaw**.

```
#
# This is file /etc/opt/SUNWsamfs/diskvols.conf on local system earth
#
local_archive       /DiskArchive
remote1  gryffindor:/quidditch
remote2   ravenclaw:/quidditch
```

File 2 is the **diskvols.conf** file that resides on the server system **gryffindor** and **ravenclaw**. Only the **diskvols.conf** file for server **gryffindor** is shown.

```
#
# This is file /etc/opt/SUNWsamfs/diskvols.conf on server system gryffindor
#
clients
earth
endclients
```

SEE ALSO

archiver(1M), **sam-fsd**(1M).

archiver.cmd(4).

NAME

ftp.cmd − Sun SAM-FS or SAM-QFS file transfer server directives file

SYNOPSIS

/etc/opt/SUNWsamfs/ftp.cmd

AVAILABILITY

SUNWsamfs

DESCRIPTION

Directives for controlling the Sun SAM-FS or SAM-QFS file transfer server are read from **/etc/opt/SUNWsamfs/ftp.cmd**. In the **ftp.cmd** file, each directive must appear on its own line. Each directive has the following format:

keyword = *value*

Comment lines can appear in the **ftp.cmd** file. A pound sign (#) in column 1 indicates a comment line.

The **ftp.cmd** file accepts the following directives:

logfile = *filename*
: Sets the name of the ftp log file to *filename*, specified as an absolute pathname. By default, no log file is written.

  The ftp log file contains a line for each file transferred. The line contains the date, time, and the name of the file.

EXAMPLES

The following is an example **/etc/opt/SUNWsamfs/ftp.cmd** file:

```
logfile = /var/opt/SUNWsamfs/log/ftp
```

The results of the ftp daemon's operations are found in **/var/opt/SUNWsamfs/log/ftp**.

FILES

The following files are used by the file transfer server:

/etc/opt/SUNWsamfs/ftp.cmd	File transfer server command file.

SEE ALSO

sam-ftpd(1M).

NAME

hosts.fs − Host information for Sun QFS shared file systems

SYNOPSIS

/etc/opt/SUNWsamfs/hosts.*fs*

AVAILABILITY

SUNWqfs

SUNWsamfs

DESCRIPTION

The **/etc/opt/SUNWsamfs/hosts.***fs* file specifies the hosts and network interfaces used by a Sun QFS shared file system. The *fs* suffix must be the family set name of the Sun QFS shared file system as specified in the **mcf**(4) file.

The file **/etc/opt/SUNWsamfs/hosts.***fs* is required by **sammkfs**(1M) at the time a Sun QFS shared file system is created. The **sammkfs**(1M) command reads **/etc/opt/SUNWsamfs/hosts.***fs* and integrates the information into the file system when initializing the file system. The file system's shared hosts information can be subsequently modified using the **samsharefs**(1M) command.

Another file, **hosts.***fs***.local**(4), can also reside on each host system included in the shared file system. Daemons local to each host system use the shared hosts file and the local hosts file, if any, to initialize network connections for the shared file system.

Each file system's shared hosts file determines the host configuration for that file system. This includes the following:

- The identity of the file system's metadata server.
- The host systems (and host IP interfaces) that are allowed to connect to the Sun QFS shared file system's metadata server.
- The identities of the potential metadata server hosts. These are systems that can act as the file system's metadata server if the preferred metadata server is unavailable.

The **hosts.***fs* file is comprised of lines containing five fields of information. Each line corresponds to one host that is permitted to access the file system. The fields are as follows:

Field Number	**Content**
1	The name of the host. The host name field contains the name of a host that is to be permitted access to the shared file system. The value of this field must match the output of the **hostname**(1) command on that host.
2	The host IP addresses. The host IP address field contains a list of one or more host IP interface addresses or names that the metadata server must be able to resolve to IP addresses. If there are multiple IP interfaces that a host can use to connect to a server, they must be separated by commas.
3	The server priority of the host. The server priority field is a numeric field. If the field is zero, the host cannot act as the metadata server for the file system. If the field is nonzero, the host can act as the metadata server for the file system.
4	A number that indicates the stager priority. This numeric field is not used by the shared file system software. It is recommended that this field be set to zero.
5	A server field. This optional field must be set for one of the hosts in the **hosts.***fs* file. That host must have a nonzero server priority field. If present, this field must contain the string **server**.

In this file, a pound character (#) indicates a comment. Comments continue from the pound character to the end of the line. All characters to the right of the pound character are ignored.

After the file system is initialized using the **sammkfs**(1M) command, only the metadata server host is permitted to run the **samfsck**(1M) to repair the file system. The server on which **sammkfs**(1M) is run is typically declared to be the metadata server.

When a client is attempting to connect to the metadata server, the client obtains the list of names and addresses from the second field, which is the host IP address field, of the server's row in the **hosts.***fs* file. It attempts to connect to these names, in the order in which they appear, until it connects successfully. If the client has a local **hosts.***fs***.local**(4) file, only the names or addresses that are present in both files are used. The **hosts.***fs***.local**(4) file determines the order in which host connections are attempted.

When a metadata server receives a connect attempt, it performs address lookups on the values from the second column of the **hosts.***fs* file until it finds one that matches the IP address of the incoming connection. If it fails to find one, it refuses the connection.

EXAMPLES

Example 1. The following is a sample **hosts.***fs* configuration file called **/etc/opt/SUNWsamfs/hosts.shsam1**.

```
#
# shsam1 config, titan/tethys servers, mimas/dione clients
#
# This file goes in titan:/etc/opt/SUNWsamfs/hosts.shsam1,
# and is used by 'sammkfs -S shsam1' to initialize the FS
# meta data.  Subsequent changes to the configuration are
# made using samsharefs(1M).
#
#
titan   titan.xyzco.com 1 0 server
tethys  tethys.xyzco.com 2 0
mimas   mimas.xyzco.com 0 0
dione   dione.xyzco.com 0 0
```

Example 2. This hosts configuration file is more complicated that the one in example 1. It supports a configuration where two potential servers also have a private interconnect between them.

```
#
# shsam1 config, titan/tethys servers, mimas/dione clients
#
# This file goes in titan:/etc/opt/SUNWsamfs/hosts.shsam1, and
# is used by mkfs -S to initialize the FS meta data.  Subsequent
# changes to the configuration are made using samsharefs(1M).
#
#
titan   titan-ge,titan.xyzco.com 1 0 server
tethys  tethys-ge,tethys.xyzco.com 2 0
mimas   mimas.xyzco.com 0 0
dione   dione.xyzco.com 0 0
```

To ensure that **titan** and **tethys** always connect to each other through their private interfaces, **titan-ge** and **tethys-ge**, each must have a hosts.shsam1.local file (see **hosts.fs.local**(4)). To avoid the inefficiencies of attempting to connect to the unreachable **titan-ge** and **tethys-ge** interfaces, **mimas** and **dione** should also have their own hosts.shsam1.local files.

FILES

/opt/SUNWsamfs/examples/hosts.shsam1
Contains an example of a **hosts.***fs* file.

/opt/SUNWsamfs/examples/hosts.shsam1.local.server

/opt/SUNWsamfs/examples/hosts.shsam1.local.client
Contain examples of **hosts.*fs*.local**(4) files.

SEE ALSO

hostname(1).

samfsck(1M), **samfsconfig**(1M), **sammkfs**(1M), **samsharefs**(1M), **sam-sharefsd**(1M).

hosts.fs.local(4), **mcf**(4).

NAME

hosts.fs.local − Local host information for Sun QFS shared file systems

SYNOPSIS

/etc/opt/SUNWsamfs/hosts.*fs*.local

AVAILABILITY

SUNWqfs

SUNWsamfs

DESCRIPTION

A **/etc/opt/SUNWsamfs/hosts.*fs*.local** file can reside on each host system included in the Sun QFS shared file system. This file is used in conjuntion with the shared hosts file, which resides in the shared file system and is initialized by **sammkfs**(1M) from **hosts.fs**(4), to initialize network connections between the hosts of a shared file system. For more information, see the **hosts.fs**(4) and **samsharefs**(1M) man pages.

The Sun QFS shared file system daemon uses the **/etc/opt/SUNWsamfs/hosts.*fs*.local** file and the shared hosts file present in the file system during initialization and reconfiguration to determine the server interfaces to which it should attempt to connect. Its function is to restrict the server interfaces to which each client connects. The *fs* portion of the name must be the family set name of the Sun QFS shared file system as specified in the **mcf** file. For more information on the **mcf** file, see the **mcf**(4) man page.

Each line in the **hosts.*fs*.local** file corresponds to a possible metadata server. Each line contains the following fields:

Field Number	**Content**
1	The name of the host. This field contains the name of a potential metadata server host to which the local host can connect. This field must match the first field of the host in the shared hosts file. You can use the **samsharefs**(1M) command to verify the content of the fields of the shared hosts file.
2	A comma-separated list of host IP names or addresses. This should be a subset of the second field from the same hosts entry in the shared hosts file.

The **hosts.*fs*.local** file is typically generated by copying the shared file system's shared hosts file to **/etc/opt/SUNWsamfs/hosts.*fs*.local** on each host. Each line referring to a non-server host is then deleted, and the third through fifth fields in the remaining lines are deleted. The network topology of the hosts is then examined in conjunction with the file, and the server interfaces that the local host should not attempt to connect to are removed from the second field. When all of these have been removed, the file is written out. The **samd**(1M) command is then used to cause any configuration changes to take effect.

During startup and file system reconfiguration, the **sam-sharefsd**(1M) daemon attempts to connect to the server host. To do this, it searches the shared hosts file for the server's identity, and it extracts the list of IP names and addresses from the server's shared hosts file entry. The daemon then looks up the server's name in the file system's local hosts file, if any. If a local hosts file does not exist, the daemon uses the list from the shared hosts file. If the local hosts file does exist, then the corresponding list of host addresses is found in the local hosts file, the two lists of host addresses are searched (lexically) for common entries, and a common list is generated. The ordering of the list is determined by the local hosts file (left-most first). The names or addresses in the common list are looked up and used to attempt to connect to the server. If an attempt fails, the daemon attempts using any remaining addresses in order until all the addresses have been tried.

EXAMPLES

The following shared hosts configuration file supports a configuration in which two potential servers share a private interconnection and communicate to the other hosts sharing the file system using a separate network. The examples in this section show the **hosts.shsam1.local** files that can be found on the various hosts.

```
#
# shsam1 config, titan/tethys servers, mimas/dione clients
#
# This file goes in titan:/etc/opt/SUNWsamfs/hosts.shsam1, and
# is used by 'mkfs -S shsam1' to initialize the FS meta data.
# Subsequent changes to the configuration are made using
# samsharefs(1M).
#
titan   titan-ge,titan.xyzco.com 1 0 server
tethys  tethys-ge,tethys.xyzco.com 2 0
mimas   mimas.xyzco.com 0 0
dione   dione.xyzco.com 0 0
```

To ensure that **titan** and **tethys** always connect to each other through their private interfaces, **titan-ge** and **tethys-ge**, each requires a **hosts.*fs*.local**(4) file. To achieve this, files **titan:/etc/opt/SUNWsamfs/hosts.shsam1.local** and **tethys:/etc/opt/SUNWsamfs/hosts.shsam1.local** would contain the following lines:

```
#
# shsam1 server local config, titan/tethys servers, mimas/dione clients
#
titan   titan-ge
tethys  tethys-ge
```

To avoid the delays and inefficiencies of having **mimas** and **dione** attempt to connect to **titan** and **tethys** through the inaccessible, private **titan-ge** and **tethys-ge** interfaces, **mimas** and **dione** should also have their own **hosts.*fs*.local**(4) files. Files **mimas:/etc/opt/SUNWsamfs/hosts.shsam1.local** and **dione:/etc/opt/SUNWsamfs/hosts.shsam1.local** contain the following lines:

```
#
# shsam1 client local config, titan/tethys servers, mimas/dione clients
#
titan   titan.xyzco.com
tethys  tethys.xyzco.com
```

FILES

/opt/SUNWsamfs/examples/hosts.shsam1
Contains an example of a **hosts.*fs*** file.

/opt/SUNWsamfs/examples/hosts.shsam1.local.server

/opt/SUNWsamfs/examples/hosts.shsam1.local.client
Contain examples of **hosts.*fs*.local** files.

SEE ALSO

samfsck(1M), **samfsconfig**(1M), **sammkfs**(1M), **samsharefs**(1M), **sam-sharefsd**(1M).

hosts.fs(4), **mcf**(4).

NAME

inquiry.conf − SCSI inquiry strings for Sun SAM-FS or SAM-QFS device types

SYNOPSIS

/etc/opt/SUNWsamfs/inquiry.conf

AVAILABILITY

SUNWsamfs

DESCRIPTION

The inquiry configuration file, **inquiry.conf**, maps a SCSI device to a Sun SAM-FS or SAM-QFS device type. The **inquiry.conf** file contains the vendor identification and product identification reported by a SCSI device in response to an inquiry command.

Entries in the file are made up of three quoted fields separated by a comma and/or white space and optionally followed by a comment. These entries have the following format:

"*vendor_id*"**,** "*product_id*"**,** "*SAM-FS_name*" *#comment*

The *vendor_id* and *product_id* are the vendor identification (8 characters) and product identification (16 characters) as reported in the inquiry data. The *SAM-FS_name* is the Sun SAM-FS or SAM-QFS device name as described subsequently on this man page.

Trailing spaces do not need to be supplied in the *vendor_id* or the *product_id* fields. Any occurrence of a quotation mark ("), a comma (**,**), or a back slash (\) in any *_id* field should be prefaced with the escape character, which is a back slash (\). Blank lines and lines beginning with a pound character (#) are ignored.

The following device names are supported within the Sun SAM-FS and SAM-QFS environments:

Device Name	**Device Type**
acl2640	ACL 2640 tape library
acl452	ACL 4/52 tape library
adic448	ADIC 448 tape library
adic100	ADIC Scalar 100 tape library
adic1000	ADIC Scalar 1000 and Scalar 10K tape library
ampexd2	Ampex D2 tape drive
ampex410	Ampex 410 Media Changer
ampex810	Ampex 810 tape library
ampex914	Ampex 914 tape library
archdat	Archive Python 4mm DAT drive
atl1500	Sun StorEdge L25 and L100 and ATL M1500 and M2500 libraries.
atlp3000	ATL P3000, P4000 and P7000 tape library
cyg1803	Cygnet Jukebox 1803 library
dlt2000	Digital Linear Tape (2000, 4000, 7000, 8000 and SuperDLT series) drive
dlt2700	Digital Linear Tape Media Changer/Stacker (2000, 4000, 7000, 8000 and SuperDLT series)
docstor	DISC automated library
exb210	Exabyte 210 tape library
exbx80	Exabyte X80 tape library
exb8505	Exabyte 8505 8mm cartidge tape drive

exbm2	Exabyte Mammoth-2 8mm cartidge tape drive
fujitsu_128	Fujitsu M8100 128 track tape drive
grauaci	GRAU media library
hpc7200	HP L9/L20/L60 series libraries
hpc1716	HP erasable optical disk drive
hpoplib	HP optical library
ibmatl	IBM ATL library
ibm0632	IBM multifunction optical disk drive
ibm3570	IBM 3570 tape drive
ibm3570c	IBM 3570 media changer
ibm3580	IBM 3580, Seagate Viper 200 and HP Ultrium (LTO) tape drives
ibm3584	IBM 3584 media changer
ibm3590	IBM 3590 tape drive
lms4100	Laser Magnetic Laserdrive 4100
lms4500	Laser Magnetic Laserdrive 4500
metd28	Metrum D-28 tape library
metd360	Metrum D-360 tape library
qual82xx	Qualstar 42xx, 62xx and 82xx series tape library
rap4500	Laser Magnetic RapidChanger 4500
rsp2150	Metrum RSP-2150 VHS video tape drive
sonyait	Sony AIT tape drive
sonydms	Sony Digital Mass Storage tape library
sonydtf	Sony DTF tape drive
speclog	Spectra Logic Libraries
stk4280	StorageTek 4280 Tape drive
stk9490	StorageTek 9490 Tape drive
stk9840	StorageTek 9840 Tape drive
stkapi	StorageTek API library
stkd3	StorageTek D3 Tape drive
stk97*xx*	StorageTek 97*xx* Media Libraries
stkl*xx*	StorageTek L20, L40, and L80 Tape Libraries and Sun StorEdge L7 and L8 autoloaders.

EXAMPLES

The following is an example configuration file:

```
"ARCHIVE", "Python",          "archdat"      # Archive python dat tape
"ATL",     "ACL4/52",         "acl452"       # ACL 4/52 tape library
"ATL",     "ACL2640",         "acl2640"      # ACL 2640 tape library
"ATL",     "P3000",           "atlp3000"     # ATL P3000 tape library
"EXABYTE", "EXB-8505",        "exb8505"      # Exabyte 8505 8mm tape
"CYGNET",  "CYGNET-1803",     "cyg1803"      # Cygnet Jukebox 1803
"IBM",     "0632",            "ibm0632"      # IBM multifunction optical
```

```
"DISC",    "D75-1",             "docstor"      # DISC automated library
"DEC",     "TZ Media Changer", "dlt2700"      # digital linear tape changer
"DEC",     "DLT2000",           "dlt2000"      # digital linear tape
"DEC",     "DLT2700",           "dlt2000"      # digital linear tape
"Quantum", "TZ Media Changer", "dlt2700"      # digital linear tape changer
"Quantum", "DLT2000",           "dlt2000"      # digital linear tape
"Quantum", "DLT4500",           "dlt2000"      # digital linear tape
"HP",      "C1716T",            "hpc1716"      # HP erasable optical disk
"HP",      "C1710T",            "hpoplib"      # HP optical library
"HP",      "C1107A",            "hpoplib"      # HP optical library
"HP",      "C1160A",            "hpoplib"      # HP optical library
```

The existence of a device in the previous example file does not imply that the device is supported by Sun SAM-FS or SAM-QFS.

SEE ALSO

mcf(4).

NOTES

Whenever a new version of Sun SAM-FS or SAM-QFS is installed, the existing **inquiry.conf** file is copied to **inquiry.conf.***MMDDYY* for reference and back-up purposes.

During device identification, the *vendor_id* and *product_id* values are only compared through the length of the string supplied in the **inquiry.conf** file. To insure an exact match, the entries should be ordered with longer names first.

WARNINGS

This interface is supplied to circumvent problems that occur when hardware vendors change the *vendor_id* and *product_id* values returned. For example, some hardware vendors return a different value for a *product_id* if the hardware is supplied by an OEM.

Sun Microsystems, Inc. does not support mapping untested hardware to a Sun SAM-FS or SAM-QFS name.

NAME

mcf − Master configuration file for Sun QFS, SAM-FS, and SAM-QFS

SYNOPSIS

/etc/opt/SUNWsamfs/mcf

AVAILABILITY

SUNWsamfs

SUNWqfs

DESCRIPTION

The **mcf** file defines the devices and family sets used by Sun QFS, SAM-FS, and SAM-QFS. The **mcf** file is read when **sam−fsd** is started. It may be changed at any time while **sam−fsd** is running. The changes will take place when **sam−fsd** is restarted, or sent the signal SIGHUP.

The following examples show an **mcf** file for a Sun SAM-FS environment and an **mcf** file for a Sun QFS file system.

Example 1. The following is an example of a Sun SAM-FS **mcf** file:

```
#
# Sun SAM-FS file system configuration example
#
# Equipment       Eq Eq Family Dev Additional
# Identifier      Or Tp Set    St  Parameters
# --------------- -- -- ------ --- ----------
samfs1            60 ms samfs1
/dev/dsk/c1t1d0s6 61 md samfs1 -
/dev/dsk/c2t1d0s6 62 md samfs1 -
/dev/dsk/c3t1d0s6 63 md samfs1 -
/dev/dsk/c4t1d0s6 64 md samfs1 -
/dev/dsk/c5t1d0s6 65 md samfs1 -
#
samfs2             2 ms samfs2
/dev/dsk/c1t1d0s0 15 md samfs2 on
/dev/dsk/c1t0d0s1 16 md samfs2 on
#
/dev/samst/c0t2d0 20 od -      on
/dev/samst/c1t2u0 30 rb hp30   on   /usr/tmp/hp30_cat
/dev/samst/c1t5u0 31 od hp30   on
/dev/samst/c1t6u0 32 od hp30   on
/dev/rmt/0cbn     40 od -      on
/dev/samst/c1t3u1 50 rb ml50   on   /usr/tmp/ml50_cat
/dev/rmt/2cbn     51 tp ml50   on
```

Example 2. The following is an example of a QFS **mcf** file:

```
#
# QFS file system configuration example
#
# Equipment       Eq Eq Family Dev Additional
# Identifier      Or Tp Set    St  Parameters
# --------------- -- -- ------ --- ----------
#
qfs1               1 ms qfs1
/dev/dsk/c1t0d0s0 11 md qfs1   on
/dev/dsk/c1t1d0s5 12 md qfs1   on
```

As the preceding examples show, each line in the **mcf** file is divided into six fields. The format of the fields in the **mcf** file is as follows:

```
Equipment  Equipment  Equipment  Family  Device  Additional
Identifier Ordinal    Type       Set     State   Parameters
```

The **Equipment Identifier**, **Equipment Ordinal**, and **Equipment Type** fields are required for each entry. The **mcf** file can contain comments. Each comment line must begin with a pound character (#). Blank lines are ignored. The fields in the file must be separated by white space. A dash character (−) can be used to indicate a field with no entry.

This man page describes the content of a Sun QFS, SAM-FS, or SAM-QFS **mcf** file. For more configuration information, see the *Sun SAM-FS or SAM-QFS File System Administrator's Guide*, publication SG-0006. After your Sun SAM-FS or SAM-QFS software is installed, you can see more examples of **mcf** files in the following directory:

/opt/SUNWsamfs/examples

MCF File Fields

The fields in the **mcf** file are defined as follows:

- The **Equipment Identifier** defines a file system and the devices associated with the file system. This field can be no longer than 127 characters in length.
- The **Equipment ordinal** can range from 1 to 65535. The **Equipment ordinal** should be kept low in order to keep the internal software tables small.
- The **Equipment type** field defines a disk cache family set, defines the disks in the family, and defines other types of devices.
- The **Family Set** field defines and associates related groups of devices. The **Family Set** name is an arbitrary name selected by the user when the **mcf** is created.
- Valid values for the **Device State** field are as follows: **on** (default), **off**, **unavail**, or **down**. This field is used for disk devices, libraries, drives, and other devices.
- The **Additional Parameters** field can contain the path to a library catalog file, an interface file, or other configuration information. The **Additional Parameters** field can be no longer than 127 characters. For example, this field can be used to specify a nondefault location for the library catalog file.

File System Disks

When defining a disk cache family set, the following entries differentiate a Sun SAM-FS file system from a Sun QFS or SAM-QFS file system:

ms	A Sun SAM-FS disk cache family set. There are no meta devices. Metadata resides on the data device(s).
ma	A Sun QFS or SAM-QFS disk cache family set with one or more meta devices. Metadata resides on these meta devices. File data resides on the data device(s).

A maximum of 252 separate magnetic disk devices can be defined for each **ms** or **ma** disk cache family set.

The **Family Set** field is required for file system disks. It is used to define the magnetic disks that make up the family set. For a magnetic disk device, the **Family Set** field entry must match a **Family Set** defined on an **ms** or **ma** entry.

The keyword **shared** must be specified in the **Additional Parameters** field if the file system is a shared file system. A **shared** file system is built by using the **−S** option to the **sammkfs**(1M) command. For more information on this option, see the **sammkfs**(1M) man page.

For each disk device, the **Equipment Identifier** field is the path to a special file, such as **/dev/dsk/c***n***t***n***d***n***s***n*. If the meta devices are not present on the clients in a shared file system, the keyword **nodev** must be specified in the **Equipment Identifier** field for the mm devices.

The following equipment types are used to define the disk devices that reside within an **ms** or **ma** file system:

mm	A magnetic disk that is part of an **ma** disk cache family set. Metadata is allocated on this device. At least one **mm** device is required in an **ma** file system.
md	A magnetic disk that is part of an **ms** or an **ma** disk cache family set. This device stores data allocated in small Disk Allocation Units (DAUs) of 4 kilobytes and large DAUs of 16, 32, or 64 kilobytes. In an **ms** family set, this device stores metadata and file data. In an **ma** family set, this device stores only file data.
mr	A magnetic disk that is part of an **ma** disk cache family set. This device allocates only large DAUs for file data.
g*XXX*	A magnetic disk that is part of an **ma** disk cache family set. The *XXX* identifies a striped group of devices. This device stores data. The allocation unit is the large DAU size multiplied by the number of members in the striped group. The *XXX* must be a decimal number in the range **0** ≤ *XXX* ≤ **127**. These devices must be the same physical size. It is not possible to use the **samgrowfs**(1M) command to increase the size of a striped group. However, it is possible to add additional striped groups.

The **Equipment Identifier** is used during the **mount**(1M) process as the **Device To Mount**. The **Device To Mount** is the first field in **/etc/vfstab** file for the mount point. For more information on this, see the **mount**(1M), **mount_samfs**(1M), or **vfstab**(1M) man pages.

SCSI-attached Libraries

Several identifiers can be used to define SCSI-attached libraries in the **mcf** file. For each SCSI-attached library, the **Equipment Identifier** field must contain the path (such as **/dev/samst/c***n***t***n***u***n*) to the special file for the device created by the **samst** device driver. For more information on the device driver, see the **samst**(7) man page.

The **Family Set** field is required. It is used to associate the library controller with the drives in the library. All devices associated with the library must have the same **Family Set** name.

The **Additional Parameters** field is optional. This field can be used to specify a nondefault location for the library catalog file. By default, catalogs are written to **/var/opt/SUNWsamfs/catalog/***family_set_name*. This file is used to store information about each piece of media in the library.

The following **Equipment Type** field entries can be used to define manually mounted or automated libraries that are attached through a SCSI interface:

Equipment Type Field Content	**Definition**
rb	Generic SCSI library that is automatically configured by Sun SAM-FS or SAM-QFS software. **NOTE:** An **rb** definition is preferred for all SCSI-attached libraries. The remainder of the library definitions in this list are supported but are not recommended for use in an **mcf** file. If a library in this list is defined in the **mcf** file as **rb**, Sun SAM-FS and SAM-QFS set the appropriate type based on the SCSI vendor code.
ad	ADIC Scalar 448 libraries.
ae	ADIC Scalar 100 libraries.

al	Sun StorEdge L25 and L100 and ATL M1500 and M2500 libraries.
as	ADIC Scalar 1000 and Scalar 10K libraries.
a1	Ampex ACL libraries.
a2	Ampex DST810 libraries.
a3	Ampex DST914 libraries.
q8	Qualstar 42xx, 62xx and 82xx series libraries
ac	ATL Products 4/52, 2640, 7100, and P-series tape libraries, and Sun 1800, 3500, L1000 and L11000 tape libraries.
cy	Cygnet optical disk libraries.
ds	DocuStore and Plasmon optical disk libraries.
eb	Exabyte 210, Sun L280, and ATL Products L-series tape libraries.
e8	Exabyte X80 libraries.
hc	HP L9/L20/L60 series
hp	Hewlett Packard optical disk libraries.
ic	IBM 3570 media changer.
me	Metrum and Mountain Gate libraries.
pd	Plasmon D-Series DVD-RAM libraries.
ml	Quantum DLTx700 tape libraries.
dm	Sony DMF and DMS libraries.
sl	Spectra Logic and Qualstar tape libraries.
s9	StorageTek 97xx series libraries.
sn	StorageTek L20, L40, and L80 tape libraries and Sun StorEdge L7 and L8 autoloaders.
il	IBM 3584 tape libraries.

Network-attached Libraries

This subsection describes how to define a network-attached library in your **mcf** file.

For each Network-attached library, the **Equipment Identifier** field must contain the path to the "parameters file" for the device.

The **Family Set** field is required. It is used to associate devices with the library. All devices associated with the library must have the same **Family Set** name.

The **Additional Parameters** field is optional. This field can be used to specify a nondefault location for the library catalog file. By default, catalogs are written to **/var/opt/SUNWsamfs/catalog/***family_set_name*. This file is used to store information about each piece of media in the library.

The network-attached library definitions are as follows:

Equipment Type Field Content	**Definition**
gr	ADIC/GRAU Network-attached library. The **Equipment Identifier** field must contain the path to the parameters file for the **grauaci** interface. For more information, see the **grauaci**(7) man page.
fj	Fujitsu LMF library. The **Equipment Identifier** field must contain the path to the parameters file for the **fujitsulmf** interface. For more information, see the

	fujitsulmf(7) man page.
im	IBM 3494 interface. The **Equipment Identifier** field must contain the path to the parameters file for the **ibm3494** interface. For more information, see the **ibm3494**(7) man page.
pe	Sony network-attached interface. The **Equipment Identifier** field must contain the path to the parameters file for the **sony** interface. For more information, see the **sony**(7) man page.
sk	StorageTek ACSLS interface. The **Equipment Identifier** field must contain the path to the parameters file for the ACSLS interface. For more information, see the **stk**(7) man page.

The Historian

The **hy** identifier in the **Equipment Type** field identifies the Sun SAM-FS or SAM-QFS historian.

The **Equipment Identifier** field must contain the string **historian**.

The **Family Set** must contain a dash character (−).

The **Additional Parameters** field is optional. This field can be used to specify a nondefault location for the historian. By default, the historian is written to **/var/opt/SUNWsamfs/catalog/historian**. This file is used to store information about the media handled by the **historian**. For more information, see the **historian**(7) man page.

Optical Disk Drives

This subsection describes the optical disk drive devices supported by Sun SAM-FS and SAM-QFS.

In the **mcf** file, a line describing an optical device must contain the following:

- The **Equipment Identifier** field must be the path to the special file, such as **/dev/samst/c***n***t***n***u***n*, for the **samst** device driver. For more information, see the **samst**(7) man page.
- The **Family Set** field is used to associate the drive with the library that has the same **Family Set**. If the family set is defined as a dash (−), the drive is assumed to be manually loaded.
- The **Equipment Type** field contains the optical drive identifier, as follows:

Equipment Type Field Content	**Definition**
od	Generic optical disk. A disk that is automatically configured by Sun SAM-FS or SAM-QFS. If you specify **od**, Sun SAM-FS or SAM-QFS sets the appropriate type based on the SCSI vendor code. **NOTE** that an **od** definition is preferred for all optical drives. If you specify **od** in the **Equipment Type** field, the Sun SAM-FS or SAM-QFS software sets the appropriate type based on the SCSI vendor code. The remainder of the definitions in this list are supported but are not recommended for use in an **mcf** file.
o2	12 inch WORM drive.
wo	5 ¼ inch optical WORM drive.
mo	5 ¼ inch erasable optical drive. The Sun SAM-FS and SAM-QFS environments support disks with 512-, 1024-, and 2048-byte sectors.
mf	IBM Multi Function optical drive.

Note that for all magneto-optical media, the default **archmax** value is 5 megabytes.

Tape Drives

This subsection describes the set of tape drives supported by Sun SAM-FS or SAM-QFS software for use in manually mounted and automated libraries.

A line in the **mcf** file for a tape drive must contain information in the following other fields:

- The **Equipment Identifier** must be the path to the raw device, typically, **/dev/rmt/***n*bn. However, it can be any symbolic link that also points to the proper special file in the /devices tree. You must specify the BSD no-rewind path.

  If the **ST_AUTODEN_OVERRIDE** drive option bit is set in an **st.conf** entry, you cannot specify a compression preference by changing the dev entry. Any attempt to specify compression is ignored. This is determined by the Sun Microsystems SCSI tape driver. The compression state of the drive is determined by its power-on default.

  For more information, see the **mtio**(7) man page. If the device supports compression, then that path should be specified for better tape usage.

- The **Family Set** field must be used to associate the device with the library that has the same **Family Set** name. If the family set is a dash character (**-**), then the device is assumed to be a manually loaded device.

- The **Additional Parameters** is required for a tape drive if the **Equipment Identifier** field does not contain information in a **/dev/rmt/*** format (the standard **st** device driver). If specified, the **Additional Parameters** field must contain the path to the special file, such as **/dev/samst/c***n***t***n***u***n*, for the **samst** device driver. For more information, see the **samst**(7) man page.

If Sun SAM-FS or SAM-QFS has access to a tape device, no other user should be allowed access the device during that period. Sun SAM-FS and SAM-QFS change the mode on the path supplied in the **mcf** file to **0660** at startup, or when the device state moves from **down** to **on**. When the state moves from **on** to **down**, the mode is set to the value of **tp_mode** in the **defaults.conf** file. For more information, see tbe **defaults.conf**(4) man page.

The following list shows the tape drives for each type of tape media supported. The tape drives supported by Sun SAM-FS and SAM-QFS are as follows:

Equipment Type Field Content	**Definition**
tp	Generic tape drive. These tapes are automatically configured by Sun SAM-FS or SAM-QFS. **NOTE** that a **tp** definition is preferred for all tape drives except for the Ampex DST. If you specify **tp** in the **Equipment Type** field, the Sun SAM-FS or SAM-QFS software sets the appropriate type based on the SCSI vendor code. The remainder of the definitions in this list are supported but are not recommended for use in an **mcf** file. This NOTE does not apply to the Ampex DST. For more information on that, see the **dst**(7) man page.
d2	Ampex DST tape drive. In the **defaults.conf** file, the default block size keyword for this media is **d2_blksize = 1024**.
dt	DAT 4mm tape drive. In the **defaults.conf** file, the default block size keyword for this media is **dt_blksize = 16**.
lt	Digital linear tape (DLT) drive. In the **defaults.conf** file, the default block size keyword for this type of media is **lt_blksize = 128**.
xt	Exabyte (850x) 8mm tape drive. In the **defaults.conf** file, the default block size keyword for this media is **xt_blksize = 16**.
xm	Exabyte Mammoth-2 8mm tape drive. In the **defaults.conf** file, the default block size keyword for this media is **xm_blksize = 128**.
fd	Fujitsu M8100 128-track tape drive. In the **defaults.conf** file, the default block size

	keyword for this media is **fd_blksize = 256**.
i7	IBM 3570 tape drive. In the **defaults.conf** file, the default block size keyword for this media is **i7_blksize = 128**.
li	IBM 3580, Seagate Viper 200 and HP Ultrium (LTO) In the **defaults.conf** file, the default block size keyword for this media is **li_blksize = 256**.
ib	IBM 3590 tape drive. In the **defaults.conf** file, the default block size keyword for this media is **ib_blksize = 256**.
vt	Metrum VHS (RSP-2150) tape drive. In the **defaults.conf** file, the default block size keyword for this media is **vt_blksize = 128**.
at	Sony AIT tape drive. In the **defaults.conf** file, the default block size keyword for this media is **at_blksize = 128**.
so	Sony DTF tape drive. In the **defaults.conf** file, the default block size keyword for this media is **so_blksize = 1024**.
st	StorageTek 3480 tape drive. In the **defaults.conf** file, the default block size keyword for this media is **st_blksize = 128**.
se	StorageTek 9490 tape drive. In the **defaults.conf** file, the default block size keyword for this media is **se_blksize = 128**.
sg	StorageTek 9840 tape drive. In the **defaults.conf** file, the default block size keyword for this media is **sg_blksize = 256**.
d3	StorageTek D3 tape drive. In the **defaults.conf** file, the default block size keyword for this media is **d3_blksize = 256**.
sf	StorageTek T9940 tape drive. In the **defaults.conf** file, the default block size keyword for this media is **sf_blksize = 256**.

For all tapes, the Sun SAM-FS or SAM-QFS system sets the block size to a media-specific default. For information on how to change the default block size, see the **defaults.conf**(4) man page.

For all tapes, the default **archmax** value is 512 megabytes.

Disk Archiving

The archiver can be configured to archive directly to online disk cache. To enable disk archiving, you must perform the following steps:

1. Create directories in online disk cache to serve as destinations for the archive copies.
2. Create the **/etc/opt/SUNWsamfs/diskvols.conf** file.
3. Edit the **archiver.cmd** file and add the **-disk_archive** directive.

The media type for a disk volume is **dk**. The block size for a disk volume is **dk_blksize=1024**. This value cannot be changed.

These steps are explained in more detail in the *Sun QFS, SAM-FS, and SAM-QFS Installation and Configuration Guide* and in the *Sun SAM-FS and SAM-QFS Storage and Archive Management Guide*.

SAM-Remote Device Definitions

Several identifiers define devices when using the Sun SAM-Remote client or SAM-Remote server software. For more information on configuring the Sun SAM-Remote client or the SAM-Remote server, see the **sam_remote**(7) man page or see the *Sun SAM-Remote Administrator's Guide*, publication SG-0003.

The identifiers used when configuring the Sun SAM-Remote client or SAM-Remote server are as follows:

Equipment Type Field Content	**Definition**
ss	Sun SAM-Remote server. The **Equipment Identifier** field must contain the path name to the server configuration file. The **Family Set** field must identify the server. That is, it must be the same as the **Family Set** name of the server. It must match the name used in the client side definition. It is used by the clients to associate the device with the server of the same **Family Set** name.
sc	Sun SAM-Remote client. The **Equipment Identifier** field must contain the path name to the client configuration file. The **Family Set** field must contain an identifier that is the same as the family set name of the server. It is used by the clients to associate the device with the server of the same **Family Set** name. The **Additional Parameters** field must contain the full path name of the client's library catalog file.
rd	Sun SAM-Remote pseudo-device. The **Equipment Identifier** field must be the path to the pseudo-device, such as **/dev/samrd/rd2**. The **Family Set** field must be the name of the server. It is used by the clients to associate the device with the server of the same **Family Set** name.

FILES

/opt/SUNWsamfs/examples Contains example **mcf** files.

SEE ALSO

Sun SAM-Remote Administrator's Guide, publication SG-0003.

Sun SAM-FS or SAM-QFS File System Administrator's Guide, publication SG-0006.

chmod(1).

build_cat(1M), **dump_cat**(1M), **mount**(1M), **mount_samfs**(1M), **sammkfs**(1M). **sam-fsd**(1M),

defaults.conf(4), **inquiry.conf**(4), **vfstab**(4).

dst(7), **fujitsulmf**(7), **grauaci**(7), **historian**(7), **ibm3494**(7), **mtio**(7), **sam_remote**(7), **samst**(7), **sony**(7), **st**(7), **stk**(7).

NAME

preview.cmd − Sun SAM-FS or SAM-QFS preview directives file

SYNOPSIS

/etc/opt/SUNWsamfs/preview.cmd

AVAILABILITY

SUNWsamfs

DESCRIPTION

An archive or stage request for a volume that is not currently loaded goes to the preview area for future consideration. A user can control the scheduling of preview requests, thus overriding the default behavior, by entering directives in the **preview.cmd** file.

The **preview.cmd** file contains directives for modifying preview request priorities. The directives allow users to increase the priority for specific VSNs and change archive request priorities based on the file system states regarding High Water Mark (HWM) and Low Water Mark (LWM). These directives are read by **sam-initd** at start-up time, and all values specified are stored in shared memory. The priority specifications cannot be changed while the **sam-initd** daemon is running.

The **preview.cmd** file can contain comments. A comment begins with a pound character (#) and extends through the end of the line.

DIRECTIVES

The directives in the **preview.cmd** file are specified one per line. With regard to their placement within the **preview.cmd** file, there are two types of directives:

- Global directives. These directives apply to all file systems. Directives are assumed to be global if they appear in the **preview.cmd** file prior to any **fs =** directives.
- Directives specific to a particular file system. File system specific directives must appear after the global directives in the **preview.cmd** file. A directive line with the following form names a specific file system and indicates that all subsequent directives apply only to that file system:

  `fs =` *file_system_family_set_name*

  A subsequent **fs =** directive in the **preview.cmd** file declares a set of directives that apply to another file system. File system specific directives override general directives.

Some directives can be used as both global and file system specific directives. This can be useful, for example, if you want to specify the **hwm_priority** directive globally to apply to most Sun SAM-FS or SAM-QFS file systems but you also want to use it as a file system specific directive to specify a different value for one particular file system.

The following sections describe the directives that can appear in a **preview.cmd** file. You can specify either an integer or a floating point value as an argument to the **_priority** directives, but the system stores the value as a floating point value internally.

GLOBAL DIRECTIVES

Global directives must appear in the **preview.cmd** file before any **fs =** directives. They cannot appear after an **fs =** directive. The global directives are as follows:

vsn_priority = *value*

This directive specifies the *value* by which the priority is to increase for VSNs marked as high-priority VSNs. For more information, see the **chmed**(1M) man page. The **vsn_priority = 1000.0** by default.

age_priority = *factor*

This global directive specifies a *factor* to to be applied to the time (in seconds) that a request is allowed to wait in the preview area to be satisfied. The *factor* is as follows:

- A *factor* > 1.0, increases the weight of the time when calculating the total priority.

- A *factor* < 1.0, decreases the weight of the time when calculating the total priority.
- A *factor* = 1.0 has no effect on the default behavior. The **age_priority = 1.0** by default.

For more information, see the PRIORITY CALCULATION section of this man page.

FILE SYSTEM SPECIFIC DIRECTIVE

The **fs =** directive specifies a particular file system and applies only to that specified file system. This directive's syntax is as follows:

fs = *file_system_family_set_name*
This directive indicates that the subsequent directives apply only to the indicated *file_system_family_set_name*.

GLOBAL OR FILE SYSTEM SPECIFIC DIRECTIVES

Several directives can be used either globally or as file system specific directives. These directives are as follows:

hwm_priority = *value*
This directive indicates the *value* by which the priority is to increase for archiving requests versus staging after the file system crosses the HWM level. This means that the releaser is running. The **hwm_priority = 0.0** by default.

hlwm_priority = *value*
This directive indicates the *value* by which the priority is to increase for archiving requests versus staging. This directive is effective when the file system is emptying, and the amount of data is between the HWM and the LWM. Because the file system is emptying, you may want to give priority to loads for stage requests. The **hlwm_priority = 0.0** by default.

lhwm_priority = *value*
This directive indicates the *value* by which the priority is to increase for archiving requests versus staging. This directive is effective when the file system is filling up, and the amount of data is between the HWM and the LWM. Because the file system is filling up, you may want to give priority to loads for archive requests. The **lhwm_priority = 0.0** by default.

lwm_priority = *value*
This directive specifies the *value* by which the priority is to increase for archiving requests versus staging when the file system is below the LWM level. The **lwm_priority = 0.0** by default.

PRIORITY CALCULATION

The total preview request priority is the sum of all priorities and is calculated as follows:

Total priority = vsn_priority + wm_priority + age_priority ∗ time_in_sec

The **wm_priority** in the previous equation refers to whichever condition is in effect at the time, either **hwm_priority**, **hlwm_priority**, **lhwm_priority**, or **lwm_priority**. All priorities are stored as floating point numbers.

EXAMPLES

Example 1. This example **preview.cmd** file sets both the **vsn_priority** and **hwm_priority** for the **samfs1** file system. Other Sun SAM-FS or SAM-QFS file systems not specified here use the default priority for the HWM. All file systems use the default priorities for the LWM and the state between LWM and HWM.

```
vsn_priority = 1000.0
fs = samfs1
hwm_priority = 100.0
```

Example 2. The next example **preview.cmd** file sets priority factors for all Sun SAM-FS or SAM-QFS file systems, but it sets an explicit and different HWM priority factor for the **samfs3** file system.

```
hwm_priority = 1000.0
hlwm_priority = -200.0
lhwm_priority = 500.0
fs = samfs3
hwm_priority = 200.0
```

SEE ALSO

chmed(1M), **sam-initd**(1M).

NAME

recycler.cmd − Sun SAM-FS or SAM-QFS sam-recycler commands file

SYNOPSIS

/etc/opt/SUNWsamfs/recycler.cmd

AVAILABILITY

SUNWsamfs

DESCRIPTION

Commands for controlling **sam-recycler**(1M) are read from **/etc/opt/SUNWsamfs/recycler.cmd**. These commands are given one per line.

logfile = *filename*

Set the name of the log file to *filename*. This file shows the overall media utilization and a sorted list of VSNs in the order in which they will be recycled. The default is no log file. See **sam-recycler**(1M) for more information.

no_recycle *media-type VSN-regexp [VSN-regexp...]*

Disallow **sam-recycler**(1M) from recycling the VSNs which match the *media-type* and the regular expression(s), *VSN-regexp*.

robot-family-set parameters

This command sets recycling parameters for a particular library identified by *robot-family-set* (this is the name given as the fourth field in the **/etc/opt/SUNWsamfs/mcf** file line defining the library for which you wish to set the parameters).

parameter

can be one more of the following:

-dataquantity *size*

This parameter sets a limit of *size* bytes on the amount of data the recycler will schedule for rearchiving in order to clear volumes of useful data. Note that the actual number of volumes selected for recycling may also be dependent on the **-vsncount** parameter. The default is 1 gigabyte (1G).

-hwm *percent*

establishes the high-water mark for the media utilization in the indicated library, specified as an integer percentage of total capacity. When the utilization of those volumes exceeds *percent*, **sam-recycler**(1M) will begin to recycle the library. The default is 95.

-ignore will keep **sam-recycler**(1M) from selecting any candidates from the specified library. The intent of this parameter is to allow a convenient way of testing other parameters.

-mail *mailaddress*

will cause **sam-recycler**(1M) to mail a message to the indicated *mailaddress* when a library’s media utilization exceeds the high-water mark. Omission of *mailaddress* prevents any mail from being sent.

-mingain *percent*

This parameter limits selection of volumes for recycling to those which would increase their free space by *percent* or more. Volumes not meeting the **-mingain** parameter are not recycled. The default is 50.

-vsncount *count*

This parameter sets a limit of *count* on the number of volumes the recycler will schedule for rearchiving in order to clear volumes of useful data. Note that the actual number of volumes selected for recycling may also be dependent on the **-dataquantity** parameter. The default is 1.

To preserve compatability with pre-existing **/etc/opt/SUNWsamfs/recycler.cmd** files, an alternative, less powerful, syntax is allowed for the library recycling parameters command.

robot-family-set robot-high-water VSN-minimum-percent-gain options

This command sets recycling parameters for a particular library identified by *robot-family-set* (this is the name given as the fourth field in the **/etc/opt/SUNWsamfs/mcf** file line defining the library for which you wish to set the parameters). *robot-high-water* establishes the high-water mark for the media utilization in the indicated library, specified as an integer percentage of total capacity. When the utilization of those volumes exceeds *percent*, **sam-recycler**(1M) will begin to recycle the library. The *VSN-minimum-percent-gain* (aka min-gain) value specifies a threshold of space available to be reclaimed (as an integer percent of total capacity of the VSN) below which VSNs will not be selected for recycling. The *options* consist of zero or more of the following: **ignore** - which will keep **sam-recycler**(1M) from selecting any candidates from the specified library. **mail** *mailaddress* - which will cause **sam-recycler**(1M) to mail a message to the indicated *mailaddress* when a library's media utilization exceeds the high-water mark. Omission of *mailaddress* prevents any mail from being sent.

script = *filename*

Supply the name of the file executed when a volume is to be relabeled. The default is **/opt/SUNWsamfs/sbin/recycler.sh**

ARCHIVER'S COMMAND FILE

The archiver's command file, **/etc/opt/SUNWsamfs/archiver.cmd**, can also specify recycling parameters for archive sets. Each archive set which has recycling parameters applied in **/etc/opt/SUNWsamfs/archiver.cmd** will be considered as a pseudo library containing just the VSNs which the archiver assigns to the archive set. See **archiver.cmd**(4) for more information. Archive set names may not be specified in the **/etc/opt/SUNWsamfs/recycler.cmd** file.

DEFAULT FILE

If there is no **/etc/opt/SUNWsamfs/recycler.cmd** file, then, for each library, a line is constructed:

library **-dataquantity 1G -hwm 95 -ignore -mail root -mingain 50 -vsncount 1**

and logging is disabled.

EXAMPLE

The following is an example **/etc/opt/SUNWsamfs/recycler.cmd** file:

```
logfile = /var/adm/recycler.log
lt20 -hwm 75 -mingain 60 -ignore
hp30 -hwm 90 -mingain 60 -mail root
gr47 -hwm 95 -mingain 60 -ignore mail root
no_recycle lt DLT.*
```

The results of **sam-recycler**(1M) operation are found in **/var/adm/recycler.log**. Three libraries are defined with various high-water marks. The first library is not recycled, but the usage information for the VSNs it contains will appear in the log, and no mail will be generated. The second library is recycled (that is, VSNs are emptied of valid archive images and relabeled) and root is sent e-mail when the library exceeds the 90% high-water mark. The third library is not recycled, but root is notified if usage exceeds the high-water mark.

For hp30, only VSNs whose recycling would free up at least 60% of the capacity of the VSN are considered.

No medium which is of media type *lt* and whose VSN begins with DLT will be recycled.

SEE ALSO

archiver.cmd(4), **mcf**(4), **sam-recycler**(1M)

NAME

recycler.sh − Sun SAM-FS or SAM-QFS sam-recycler post-process shell escape

SYNOPSIS

/opt/SUNWsamfs/sbin/recycler.sh

AVAILABILITY

SUNWsamfs

DESCRIPTION

/opt/SUNWsamfs/sbin/recycler.sh is a script which is executed by sam-recycler when it has finished draining a VSN of all known active archive images.

DEFAULT FILE

As released, **/opt/SUNWsamfs/sbin/recycler.sh** contains a script which will mail root with the relevant information.

EXAMPLE

The following is an example **/opt/SUNWsamfs/sbin/recycle.sh** file:

```
#!/bin/csh

#   /opt/SUNWsamfs/sbin/recycler.sh - post-process a VSN after
#   sam-recycler has drained it of all known active archive
#   copies.

#   Arguments are:
#      $1 - generic media type "od" or "tp" - used to
#                   construct the name of the appropriate label
#                   command - tplabel or odlabel
#
#      $2 - VSN being post-processed
#
#      $3 - slot in the library where the VSN is
#           located
#
#      $4 - equipment number of the library where the
#           VSN is located
#
#      $5 - actual media type ("mo", "lt", etc.) - used to
#           chmed the medium if required
#
#      $6 - family set name of the library where the VSN is
#           located
#

if ( $6 != hy ) then
    /opt/SUNWsamfs/sbin/chmed -R $5 $2
    /opt/SUNWsamfs/sbin/chmed -W $5 $2
    /opt/SUNWsamfs/sbin/${1}label -vsn $2 -old $2 \
         $4:$3
else
    mail root <</eof
Please bring VSN $2 of type $5 back on site.   It has
finished recycling and can be reused.
/eof
```

```
 endif
```

The example first checks to see if the VSN is in a physical robot. If it is, the example first clears the "read-only" and "write-protect" catalog bits, then issues a tplabel or odlabel command to relabel the medium with its existing label. Relabeling has the effect of clearing all the expired archive images from the medium, thus enabling it to be re-used by the archiver. Labeling also clears the recycle bit in the VSN's catalog entry.

If the VSN is in the historian catalog, then an e-mail message is sent to root. Note that a medium in a manually-mounted drive is shown in the historian catalog as well, so you may wish to check to see if the VSN is currently in a drive and relabel it if so.

SEE ALSO

sam-recycler(1M), **tplabel**(1M), **odlabel**(1M)

NAME
releaser.cmd − Sun SAM-FS and SAM-QFS releaser command file

SYNOPSIS
/etc/opt/SUNWsamfs/releaser.cmd

AVAILABILITY
SUNWsamfs

DESCRIPTION
Directives for controlling the releaser can be read from the **/etc/opt/SUNWsamfs/releaser.cmd** file. The directives must appear one per line.

Comment lines are permitted. Comment lines must begin with a pound character (#), and the comment can extend through the rest of the line.

Directives that appear prior to any **fs=** directive are applied to all file systems. Directives that appear after a **fs=** directive are applied to the specified file system only. Directives that are specific to a file system override general directives.

The directives on this man page are divided into groups. The weight directives for size and age determine the release priority of a file. The miscellaneous directives control whether a log file is written, whether there is a minimum age required for files, and other aspects of releasing.

WEIGHT DIRECTIVES
The following weights are used to calculate the release priority of each file in the file system. Each file's priority is composed of two parts: size priority and age priority. The size priority plus the age priority equals the file's total release priority.

Size Priority
The size priority is determined by the value of the **weight_size** directive. This directive has the following format:

weight_size=*weight_size_value*

Sets the weight factor for the size of the file to *weight_size_value*. Specify a floating-point number in the following range:
0.0 ≤ *weight_size_value* ≤ 1.0. The default is 1.0.

The *weight_size_value* is multiplied by the size of the file in 4-kilobyte blocks to arrive at the size component of the file's release priority.

Age Priority
The age priority can be calculated in one of the following ways:

- The first method multiplies the value of the **weight_age=** directive by the most recent of the following ages: access age, modify age, and residence change age. The access age is defined as the current time minus the file's last access time. The **weight_age** directive has the following format:

  weight_age=*weight_age_value*

  Sets the weight factor for the overall age of the file to *weight_age_value*. The *weight_age_value* is multiplied by the most recent of the file's access, modify or residence change age to arrive at the age component of the file's release priority. Specify a floating-point number in the following range:
  0.0 ≤ *weight_age_value* ≤ 1.0. The default is 1.0.

  If you specify a **weight_age=** directive for a given file system, you cannot specify **weight_age_access=**, **weight_age_modify=**, or **weight_age_residence=** directives for the same file system.

- The second method allows you to specify separate weights for the access, modify, and residence ages. The ages are calculated in units of 60-second minutes.

If you want to specify separate weights for the access, modify, and residence ages, use the following directives in the **releaser.cmd** file:

weight_age_access=*weight_age_access_value*
Sets the weight factor for the access age of the file to *weight_age_access_value*. Specify a floating-point number in the following range:
0.0 ≤ *weight_age_access* ≤ 1.0. The default is 1.0.

The *weight_age_access_value* is multiplied by the file's access age (expressed in minutes). This product, added to the sum of the products of the modify and residence-change ages multiplied by their respective weights, becomes the age component of the file's release priority.

If you specify a **weight_age=** directive for a given file system, you cannot specify a **weight_age_access=** directive for the same file system.

weight_age_modify=*weight_age_modify_value*
Sets the weight factor for the modify age of the file to *weight_age_modify_value*. Specify a floating-point number in the following range:
0.0 ≤ *weight_age_modify* ≤ 1.0. The default is 1.0.

The *weight_age_modify_value* is multiplied by the file's modify age (expressed in minutes). This product, added to the sum of the products of the modify and residence-change ages multiplied by their respective weights, becomes the age component of the file's release priority.

If you specify a **weight_age=** directive for a given file system, you cannot specify a **weight_age_modify=** directive for the same file system.

weight_age_residence=*weight_age_residence_value*
Sets the weight factor for the residence-change age of the file to *weight_age_residence_value*. Specify a floating-point number in the following range:
0.0 ≤ *weight_age_residence* ≤ 1.0. The default is 1.0.

The *weight_age_residence_value* is multiplied by the file's residence-change age (expressed in minutes). This product, added to the sum of the products of the modify and residence-change ages multiplied by their respective weights, becomes the age component of the file's release priority.

If you specify a **weight_age=** directive for a given file system, you cannot specify a **weight_age_residence=** directive for the same file system.

MISCELLANEOUS DIRECTIVES

The following miscellaneous directives can be specified in the **releaser.cmd** file:

fs = *file_system_family_set_name*
Specifies to the releaser that the subsequent directives apply to the indicated *file_system_family_set_name* only.

no_release
Prevents the releaser from releasing any files. This directive is useful when you are tuning the priority weights. Also see the **display_all_candidates** directive. By default, files are released.

rearch_no_release
Prevents the releaser from releasing files marked to be rearchived. By default, files marked for rearchive are released.

logfile = *filename*

Sets the name of the releaser's log file to *filename*. By default, no log file is written.

display_all_candidates
Writes the releaser priority for each file, as it is encountered, to the log file. This can be useful in tuning when used in conjunction with the **no_release** directive. This directive allows you to judge the effect of changing the priority weights. By default file priority is not displayed in any way.

min_residence_age = *time*
Sets the minimum residency age to *time* seconds. This is the minimum time a file must be online before it is considered to be a release candidate. The default is 600 seconds (10 minutes).

EXAMPLES

Example 1. This example file sets the **weight_age=** and **weight_size=** directives for the **samfs1** file system. No releaser log is produced.

```
fs = samfs1
weight_age = .45
weight_size = 0.3
```

Example 2. This example provides weights for all file systems. All file system releaser runs are logged to **/var/adm/releaser.log**.

```
weight_age = 1.0
weight_size = 0.03
logfile = /var/adm/releaser.log
```

Example 3. This example specifies weights and log files for each file system.

```
logfile = /var/adm/default.releaser.log

fs = samfs1

weight_age = 1.0
weight_size = 0.0
logfile = /var/adm/samfs1.releaser.log

fs = samfs2

weight_age_modify = 0.3
weight_age_access = 0.03
weight_age_residence = 1.0
weight_size = 0.0
logfile = /var/adm/samfs2.releaser.log
```

Example 4. This example is identical in function to example 3, but it uses specifies the **weight_size=** directive globally.

```
logfile = /var/adm/default.releaser.log
weight_size = 0.0

fs = samfs1

weight_age = 1.0
logfile = /var/adm/samfs1.releaser.log
```

```
                fs = samfs2

                weight_age_modify = 0.3
                weight_age_access = 0.03
                weight_age_residence = 1.0
                logfile = /var/adm/samfs2.releaser.log
```

SEE ALSO

release(1).

NAME

samfs.cmd − Sun SAM-FS or SAM-QFS mount commands file

SYNOPSIS

/etc/opt/SUNWsamfs/samfs.cmd

AVAILABILITY

SUNWsamfs

DESCRIPTION

Commands for controlling **samfs** mount parameters are read from **/etc/opt/SUNWsamfs/samfs.cmd**. These commands serve as defaults, and can be superseded by parameters on the mount command. See mount_samfs (1M). The **/etc/opt/SUNWsamfs/samfs.cmd** file is read when **sam-fsd** is started. It may be changed at any time while **sam-fsd** is running. The changes will take place when **sam-fsd** is restarted, or sent the signal SIGHUP.

These commands are given one per line. Comments begin with a # and extend through the end of the line. Commands given before any "fs =" line apply in general to all filesystems; "fs =" introduces commands which are specific to the mentioned filesystem only. Filesystem-specific commands override general commands.

COMMANDS

See **mount_samfs**(1M) under OPTIONS for the list of supported commands. The following additional commands are available as well.

inodes = *n*

This sets the maximum number of incore inodes assigned to all *samfs* file systems. Each incore inode allocates 512 bytes of memory. If inodes is zero or less than *ncsize*, a default value will be set to *ncsize*. *ncsize* is the size of the Solaris name cache. The default is zero. This parameter applies to all file systems.

fs = *fs_name*

This command specifies the following commands apply only to the indicated file system with family set name *fs_name*.

EXAMPLE

This example file sets *high* and *low* for 2 different filesystems, samfs1 and samfs2.

```
fs = samfs1
  high = 90
  low = 80
fs = samfs2
  high = 80
  low = 75
```

SEE ALSO

release(1), **setfa**(1), **mount_samfs**(1M), **sam_releaser**(1M), **sam_advise**(3), **sam_setfa**(3), **sam-fsd**(1M), **directio**(3C), **mcf**(4)

NAME

sefdata − System Error Facility (SEF) data for Sun SAM-FS and SAM-QFS

SYNOPSIS

/var/opt/SUNWsamfs/sef/sefdata

#include "**/opt/SUNWsamfs/include/sefvals.h**"

#include "**/opt/SUNWsamfs/include/sefstructs.h**"

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **sefdata** file contains the data gathered from the log sense pages of peripheral tape devices used by Sun SAM-FS and SAM-QFS. Each time the Sun SAM-FS or SAM-QFS software unloads a cartridge from a drive, pertinent log sense pages are obtained from the device, and a record is written to the **sefdata** file. Each record consists of a header followed by some number of log sense pages.

The record header has the format of a **sef_hdr** structure. This structure is defined in **/opt/SUNWsamfs/include/sefstructs.h**, and it has the following components:

```
struct sef_hdr {
    uint_t      sef_magic;          /∗ magic # for app to sync file posn ∗/
    uint_t      sef_version;        /∗ version number ∗/
    uint_t      sef_size;           /∗ size of this record, excl. header ∗/
    uint16_t    sef_eq;             /∗ equipment number of this device ∗/
    char        sef_devname[128];   /∗ pathname of device ∗/
    uchar_t     sef_vendor_id[9];   /∗ vendor id from inquiry ∗/
    uchar_t     sef_product_id[17]; /∗ product id from inquiry ∗/
    uchar_t     sef_revision[5];    /∗ revision level from inquiry ∗/
    uchar_t     sef_scsi_type;      /∗ device type from inquiry ∗/
    vsn_t       sef_vsn;            /∗ vsn of media that was mounted ∗/
    time_t      sef_timestamp;      /∗ timestamp of this record ∗/
}
```

The fields of the **sef_hdr** structure have the following meanings:

Field	**Content**
sef_magic	Has the value **SEFMAGIC**, as defined in **/opt/SUNWsamfs/include/sefvals.h**.
sef_version	Has the value **SEFVERSION**, as defined in **/opt/SUNWsamfs/include/sefvals.h**.
sef_size	The size of this record, excluding the header.
sef_eq	The equipment number of the device, as configured in the **mcf** file. For more information, see the **mcf**(4) man page.
sef_devname	A character string containing the path name of the device.
sef_vendor_id	The vendor identification of the device, as obtained from inquiry.
sef_product_id	The product identification of the device, as obtained from inquiry.
sef_revision	The revision level of the device, as obtained from inquiry.
sef_scsi_type	The device type, as obtained from inquiry.
sef_vsn	Volume Serial Name (VSN) of the volume mounted in the device when the data was generated.
sef_timestamp	Time that this record as written to the data file.

Following the header in each record is some number of log sense pages. Each log sense page consists of a SCSI-standard header followed by triplets of parameter codes, control values, and parameter values. For the exact format of the log sense pages returned by the devices in use at your site, consult the documentation provided with those devices.

FILES

File	Purpose
/var/opt/SUNWsamfs/sef/sefdata	Contains SEF information.
/opt/SUNWsamfs/include/sefvals.h	Contains values, such as those for **SEFMAGIC** and **SEFVERSION**.
/opt/SUNWsamfs/include/sefstructs.h	Contains include files for the SEF header, the SCSI-standard header, and other structures.

SEE ALSO

Sun SAM-FS and SAM-QFS Storage and Archive Management Guide, publication SG-0008.

sefreport(1M).

NAME

stager.cmd − Sun SAM-FS or SAM-QFS stager directives file

SYNOPSIS

/etc/opt/SUNWsamfs/stager.cmd

AVAILABILITY

SUNWsamfs

DESCRIPTION

Directives for controlling the Sun SAM-FS or SAM-QFS stager are read from **/etc/opt/SUNWsamfs/stager.cmd**. In the **stager.cmd** file, each directive must appear on its own line. Each directive has the following format:

keyword = *value*

Comment lines can appear in the **stager.cmd** file. A pound sign (#) in column 1 indicates a comment line.

The **stager.cmd** file accepts the following directives:

drives = *library count*

Sets the number of drives to use for staging on media library *library* to a number specified by *count*. The default value is the actual number of drives in *library*.

The *library* specified must be the family set name of a media library as defined in the **mcf** file. If this directive is specified, the stager uses only *count* number of drives in the media library to stage archive copies. This directive prevents the stager from using all drives in a media library and possibly interfering with archiving.

For example, the following directive specifies that 3 drives should be used for staging in an ADIC/Grau media library.

```
drives = gr50 3
```

bufsize = *media buffer_size* [**lock**]

Sets the stage buffer size for a specific media type.

For *media*, specify a media type from the **mcf**(4) man page.

For *buffer_size*, specify an integer value in the range **2** ≤ *buffer_size* ≤ **32**. The default is **4**. The *buffer_size* specified is multiplied by the default block size for *media*. For more information on default block sizes, see the *dev*_**blksize** description on the **defaults.conf**(4) man page.

If **lock** is specified, the stager locks the stage buffer in memory. If the stage buffer is locked, system CPU time can be reduced.

logfile = *filename*

Sets the name of the stager log file to *filename*, specified as an absolute pathname. By default, no log file is written.

The stager log file contains a line for each file staged. The line contains the date, time, media, VSN, inode generation number of the file, position and offset of where the file is stored, and the name of the file.

maxactive = *number*

Sets the maximum number of stage requests that can be active at one time in the stager to an integer *number*. The minimum *number* is **1**. The default *number* is **1000**.

EXAMPLES

The following is an example **/etc/opt/SUNWsamfs/stager.cmd** file:

```
logfile = /var/opt/SUNWsamfs/log/stager
drives= hp30 1
```

The results of the stager's operations are found in **/var/opt/SUNWsamfs/log/stager**. For the media library specified as **hp30**, the stager is allowed to use only 1 drive for staging files.

FILES

The following files are used by the stager:

/etc/opt/SUNWsamfs/stager.cmd	Stager command file.

SEE ALSO

sam-stagerd(1M).

defaults.conf(4), **mcf**(4).

NAME

media − List of media supported by SAM-FS

AVAILABILITY

SUNWsamfs

DESCRIPTION

This man page is obsolete. All information maintained on this man page prior to the Sun Microsystems SAM-FS and SAM-QFS 4.0 release has been moved to the **mcf**(4) man page.

This man page will be removed in a future major release.

NAME

acl2640 − The ACL2640 Automated Tape Library

AVAILABILITY

SUNWsamfs

DESCRIPTION

The ACL2640 tape library supports 264 DLT tape cartridges and 3 DLT tape drives. The library has a import/export unit that may be used to import or export media into the library. The import unit takes one cartridge at a time and the export unit will hold up to 12 cartridge.

CONFIGURATION

The ACL2640 should **NOT** be configured with auto-clean when running Sun SAM-FS and SAM-QFS software.

IMPORT/EXPORT MEDIA

To import media the door on the ACL2640 must first be opened. To open the door issue the **import**(1M) command or function the import button in **robottool**(1M) then follow the instructions in the ACL2640 operator's Guide.

To export media, use the **export**(1M) command or the export button in **robottool** to move media into the export unit then following the instructions in the ACL2640 Operator's Guide.

FILES

mcf	The configuration file for Sun SAM-FS and SAM-QFS software.

SEE ALSO

export(1M), **import**(1M), **mcf**(4), **sam-robotsd**(1M), **robottool**(1M)

NAME

acl452 − The ACL 4/52 Automated Tape Library

AVAILABILITY

SUNWsamfs

DESCRIPTION

The ACL 4/52 tape library supports 48 DLT tape cartridges and 4 DLT tape drives. The library has a 4 slot import/export unit that may be used to import or export media into the library. These import/export slots may also be used as storage slots thus extending the storage capacity to 52 slots.

CONFIGURATION

The ACL 4/52 should **NOT** be configured with auto-clean or auto-load when running Sun SAM-FS and SAM-QFS software. Auto-load may be used during initial loading of cartridges as long as the Sun SAM-FS and SAM-QFS software is not running.

IMPORT/EXPORT MEDIA

To import media, the door on the ACL 4/52 must first be opened. To open the door, issue the **export**(1M) command or function the export button in **robottool**(1M) then push the OPEN button on the ACL 4/52 front panel. The door should open and you may place media in any of the slots. You may then close the door by pressing the CLOSE button then manually closing the door when the ACL 4/52 display indicates that it is ready. The Sun SAM-FS and SAM-QFS software will not recognize the new media until the **import**(1M) command is issued or the import button is functioned in **robottool**. Anytime you close the door, you must issue the **import** function.

To export media, use the **move**(1M) command or the move button in **robottool** to move media into the import/export unit then issue the **export** command or function the export button in **robottool**. Push the OPEN button on the ACL 4/52 front panel to open the door.

If the door is already open, you must close the door and issue the **import** command before attempting to move media into the import/export unit.

The slot numbers for the import/export unit are 48, 49, 50 and 51.

Note: After opening or closing the door, the ACL 4/52 goes offline until it has re-initialized. This will cause delays since the library must become online before any commands may be issued.

FILES

mcf	The configuration file for the Sun SAM-FS and SAM-QFS software

SEE ALSO

export(1M), **import**(1M), **move**(1M), **mcf**(4), **sam-robotsd**(1M), **robottool**(1M)

NAME

dst − The AMPEX DST interface

AVAILABILITY

SUNWsamfs

DESCRIPTION

Using AMPEX DST tape drives with Sun SAM-FS and SAM-QFS software requires the installation of the AMPEX DST Tape Device Driver supplied by AMPEX *BEFORE* the SUNWsamfs package is installed. Since the AMPEX driver does not follow the same naming conventions as the standard Solaris tape driver (see **st**(7D)) the definitions of the *equipment identifier* and *additional parameter* fields of the mcf (see **mcf**(4)) are modified as follows:

equipment identifier

is the path name to the non-rewinding DST device special file. Using the configuration supplied by AMPEX, this would be /dev/rdst∗,1 or /dev/rdst∗.1. In addition, the device driver bit DST_ZERO_ON_EW must be set for this configuration (dst_dev_options = 0x00004001). This will require a change to the dst.conf in /usr/kernel/drv.

additional parameter

is the path name to the "no I/O on open" DST device special file. Using the configuration supplied by AMPEX, this would be /dev/rdst∗,7 or /dev/rdst∗.7 (dst_dev_options = 0x0000001f).

NOTES and CAUTIONS

The Sun SAM-FS and SAM-QFS software requires version 3.4 of the AMPEX DST Tape Device Driver; earlier versions will not work.

The Sun SAM-FS and SAM-QFS software uses only the first partition (partition 0) on a pre-formatted tape and the vsn must be the same as the volid converted to upper case. If **tplabel** is used with the **-erase** option, the tape will initialized with a single partition that occupies the entire tape with the volid the same as the vsn. This is the same as using the AMPEX dd2_format_tape utility specifying -format ":Initialize:VSN:0:1:::::".

The DST310 tape drive has an option for spacing the tape to physical end of tape before ejecting. This option should **NOT BE USED**. Under certain conditions the drive will not correctly re-position the tape when it is re-inserted and data loss can occur.

The DST configuration file (/usr/kernel/drv/dst.conf) **MUST BE CHANGED** to turn on the DST_ZERO_ON_EW flag for the second entry of the dst_dev_options table (this correlates to /dev/rdst∗,1 and /dev/rdst∗.1). If this change is not made, end-of-media will not be properly detected. Note: After changing dst.conf, the driver must be reloaded (either by unloading the DST SCSI module or rebooting).

SEE ALSO

tplabel(1M), **AMPXdst**(7), **mtio**(7), **st**(7), **defaults.conf**(4), **inquiry.conf**(4)

NAME

fujitsulmf − The Fujitsu LMF Automated Tape Library

AVAILABILITY

SUNWsamfs

DESCRIPTION

fujitsulmf is the Sun SAM-FS and SAM-QFS software interface to the Fujitsu LMF library. This interface utilizes the LMF interface supplied by Fujitsu. For more information on LMF, see the LMF MTL Server/Client User's Guide supplied by Fujitsu.

CONFIGURATION

It is assumed that the site has the LMF server configured and operating with the LMF library.

The "equipment identifier" field in the **mcf** file, (see **mcf**(4)), is the full path name to a parameters file used by *fujitsulmf*. This file consists of a list of keyword = value pairs or a keyword followed by a drivename = value pair. All keywords and values are case-sensitive and must be entered as shown.

lmfdrive

There is one *lmfdrive* line for every drive assigned to this client. Following the *lmfdrive* keyword is a drivename = path, where:

drivename

is the drivename as configured in LMF.

path is the pathname to the device. This name must match the "equipment identifier" of an entry in the **mcf** file.

EXAMPLE

Here are sample parameters files and mcf entries for an LMF library.

```
#
# This is file: /etc/opt/SUNWsamfs/lmf50
#
# the name "LIB001DRV000" is from the LMF configuration
#
lmfdrive LIB001DRV000 = /dev/rmt/0cbn   # a comment
#
# the name "LIB001DRV001" is from the LMF configuration
#
lmfdrive LIB001DRV001 = /dev/rmt/1cbn  # a comment

The mcf file entries.

#
# Sample mcf file entries for an LMF library
#
/etc/opt/SUNWsamfs/lmf50 50   fj   fj50  -  /var/opt/SUNWsamfs/catalog/fj50_cat
/dev/rmt/0cbn           51   fd   fj50  -  /dev/samst/c2t5u0
/dev/rmt/1cbn           52   fd   fj50  -  /dev/samst/c2t6u0
```

IMPORT/EXPORT

Since the physical adding and removing of media in the LMF library is done with LMF utilities, the import/export commands and libmgr import/export menus will only affect the library catalog. The **import** command has an optional parameter (see **import**(1M)) (−**v**) for supplying the volser to be added. *fujitsulmf* will verify that LMF knows about the volser before updating the catalog with the new entry. The **export** command (see **export**(1M)) will remove the entry from the catalog.

CATALOG

There are two utilities used to maintain the library catalog used by LMF. **build_cat** (see **build_cat**(1M)) is used to build the catalog. **dump_cat** (see **dump_cat**(1M)) and **build_cat** together are used to change the size of the catalog.

To initialize a catalog with 1000 slots run:

```
build_cat /tmp/catalog_file < /dev/null
```

then move /tmp/catalog_file to the path pointed to in the mcf file for this media changer. Use import to populate the catalog with the volumes allowed by DAS. Or, you can create a file with the list of volumes and supply it as input to **build_cat** (see **build_cat**(1M)) for the format of the input file.

If the size of the catalog needs to be increased, execute something like:

```
dump_cat file1 | build_cat -s 2000 /tmp/file2
```

This would create a new catalog file (/tmp/file2) with room for 2000 entries and initialize it with the entries from file1. This should only be done when the Sun SAM-FS and SAM-QFS software is not running and **sam-initd** has been shutdown (see **sam-initd**(1M)).

FILES

mcf	The configuration file for the Sun SAM-FS and SAM-QFS software.
/opt/SUNWsamfs/lib/liblmf2.so	The LMF library supplied by Fujitsu.

SEE ALSO

build_cat(1M), **dump_cat**(1M), **export**(1M), **import**(1M), **mcf**(4), **sam-robotsd**(1M), **libmgr**(1M)

NAME

grauaci − The ADIC/Grau Automated Tape Library through the ACI

AVAILABILITY

SUNWsamfs

DESCRIPTION

grauaci is the Sun SAM-FS and SAM-QFS software interface to the ADIC/Grau Network-attached library. This interface utilizes the DAS/ACI 3.10E interface supplied by ADIC. For more information on DAS/ACI, see the *DAS/ACI 3.10E Interfacing Guide* and the *DAS Administration Guide.* Both manuals are supplied by ADIC.

CONFIGURATION

Sun assumes that your site has the DAS server configured and operating with the ADIC/Grau library. In the DAS configuration file for this client, the **avc** (avoid volume contention) and the **dismount** parameters should both be set to **true**.

The Equipment Identifier field in the **mcf** file is the full path name to a parameters file used by **grauaci**. This file consists of a list of *keyword = value* pairs or a keyword followed by a *drivename = value* pair. For more information on the **mcf** file, see the **mcf**(4) man page.

All keywords and values, including the following, are case sensitive and must be entered as shown:

Keyword	**Value**
client	This is the name of this client as defined in the DAS configuration file. This is a required parameter.
server	This is the hostname of the server running the DAS server code. This is a required parameter.
acidrive	There is one **acidrive** line for every drive assigned to this client. Following the **acidrive** keyword is a *drivename = path,* string that is as follows:

drivename	The drive name as configured in the DAS configuration file.
path	The path name to the device. This name must match the Equipment Identifier of an entry in the **mcf** file.

If the library contains different media types, then there must be a separate media changer for each of the media types. Each media changer must have a unique client name in the DAS configuration, a unique library catalog and a unique parameters file.

EXAMPLE

The following example shows sample parameters files and **mcf** entries for a ADIC/Grau library supporting DLT tape and HP optical drives. The catalog files are placed in the default directory, which is **/var/opt/SUNWsamfs/catalog**.

```
#
# This is file: /etc/opt/SUNWsamfs/gr50
#
client = grau50
server = DAS-server
#
# the name "drive1" is from the DAS configuration file
#
acidrive drive1 = /dev/rmt/0cbn         # a comment
#
# the name "drive2" is from the DAS configuration file
#
acidrive drive2 = /dev/rmt/1cbn         # a comment
```

```
#
# This is file: /etc/opt/SUNWsamfs/gr60
#
client = grau60
server = DAS-server
#
# the name "DH03" is from the DAS configuration file
#
acidrive DH03 = /dev/samst/c1t1u0

The mcf file entries.

#
# Sample mcf file entries for an ADIC/Grau library - DLT
#
/etc/opt/SUNWsamfs/gr50  50      gr      gr50   -  gr50cat
/dev/rmt/0cbn           51      lt      gr50   -  /dev/samst/c2t5u0
/dev/rmt/1cbn           52      lt      gr50   -  /dev/samst/c2t6u0

#
# Sample mcf file entries for an ADIC/Grau library - HP optical
#
/etc/opt/SUNWsamfs/gr60  60      gr      gr60   -  gr60cat
/dev/samst/c1t1u0       61      od      gr60   -
```

IMPORT/EXPORT

The physical adding and removing of cartridges in an ADIC/Grau network-attached library is accomplished using the DAS utilities. The **import**(1M) and **export**(1M) commands and the **libmgr**(1M) and **robottool**(1M) import and export menus affect only the library catalog. Therefore, importing and exporting cartridges with the ADIC/Grau network-attached library consists of the following two-step process:

1) Physically import or export the cartridge using the DAS utilities.

2) Virtually update the automated library catalog using the Sun SAM-FS or SAM-QFS import and export utilities.

The **import**(1M) command has an optional **-v** parameter for supplying the VSN to be added. The **grauaci** interface verifies that DAS knows about the VSN before updating the catalog with the new entry. The **export**(1M) command removes the entry from the catalog. For more information on importing and exporting, see the **import** and **export**(1M) man pages.

CATALOG

There are several methods for building a catalog for an ADIC/Grau network-attached library. You should use the method that best suits your system configuration, and this is typically determined by the size of the catalog that is needed.

Method 1: Create a catalog with existing VSN entries. (Please note this method only works for tapes. It does not work for barcoded optical media.) You can build a catalog that contains entries for many tapes by using the **build_cat**(1M) command. As input to **build_cat**(1M), you need to create a file that contains the slot number, VSN, barcode, and media type. For example, file **input_vsns** follows:

```
0 TAPE01    TAPE01      lt
1 TAPE02    TAPE02      lt
2 TAPE03    TAPE03      lt
```

The **input_vsns** file can be used as input to the **build_cat**(1M) command, as follows:

```
    build_cat input_vsns /var/opt/SUNWsamfs/grau50cat
```

Method 2: Create a null catalog and import VSN entries. You can create an empty catalog and populate it. To create a catalog that will accommodate 1000 slots, use the **build_cat** command, as follows:

```
       build_cat -s 1000 /dev/null /var/opt/SUNWsamfs/catalog/grau50cat
```

Use the **import**(1M) command to add VSNs to this catalog, as follows:

```
       import -v TAPE01 50
```

For ADIC/Grau optical media, it is very important to import the A side of barcoded optical media. The Sun SAM-FS and SAM-QFS software queries the ADIC/Grau database to find the barcode for the B side and fills in the catalog entry for the B side appropriately. The A side of optical media in the ADIC/Grau automated library is the left side of a slot as you face the slots.

Method 3: Use the default catalog and import VSN entries. If a catalog path name is not specified in the **mcf** file, a default catalog is created in **/var/opt/SUNWsamfs/catalog/***family_set_name* when the Sun SAM-FS or SAM-QFS software is initialized. Following initialization, you must import VSN entries to this catalog. Use the **import**(1M) command, as follows:

```
       import -v TAPE01 50
```

In the preceding **import**(1M) command, **50** is the Equipment Identifier of the automated library as specified in the **mcf** file.

FILES

mcf	The configuration file for the Sun SAM-FS and SAM-QFS software.
/opt/SUNWsamfs/lib/libaci.so	The ACI library supplied by ADIC.

SEE ALSO

build_cat(1M), **dump_cat**(1M), **export**(1M), **import**(1M), **libmgr**(1M), **sam-robotsd**(1M).

mcf(4).

NAME

historian − The Sun SAM-FS and SAM-QFS historian

AVAILABILITY

SUNWsamfs

DESCRIPTION

historian is a catalog that keeps track of volumes that have been exported from an automated library or that have been unloaded from manually loaded devices.

CONFIGURATION

The **historian** catalog is similar to the catalog for an automated library but since there are no devices associated with it, has no family set name. If there is no historian catalog configured in the mcf file (see **mcf**(4)) one will be created as:

```
    historian    n+1    hy    -    -    /var/opt/SUNWsamfs/catalog/historian
```

Where *n+1* is the highest equipment number defined in the mcf file plus 1.

The historian catalog will be created with 32 entries when the catalog server initializes and can grow during execution. Each time the catalog fills, 32 entries of approximately 200 bytes each will be added. Make sure the historian's catalog resides on a file system large enough to hold the expected size. Since the catalog is needed before a sam file system can be mounted, DO NOT put the catalog on a Sun SAM-FS or SAM-QFS file system.

Two configuration parameters in the defaults.conf file (see **defaults.conf**(4)) affect the way the system will react to requests for media or requests to add media to the historian catalog. If *exported_media* is set to unavailable, then any media exported from a media changer will be set to unavailable in the historian. Any request for media flagged as unavailable will receive an ESRCH error. If *attended* is set to "no" (operator is NOT available), then any request for media in the historian catalog will be sent back to the file system with an error (ESRCH). Any request for media currently loaded in a manually loaded drive will be accepted no matter what the state of the attended or unavailable flags are.

EFFECTS OF HISTORIAN

Whenever the file system receives the error ESRCH for a stage request, it will automatically generate a stage request for the next archived copy (unless the last stage request was for the last copy). For a removable media request, the error ESRCH will be returned to the user.

IMPORT/EXPORT

import (see **import**(1M)) is used to insert entries to the historian catalog.

export (see **export**(1M)) is used to remove entries from the historian catalog. You may export by slot or vsn.

CATALOG

The catalog server will create a new, empty catalog in the default file location if none exists or no catalog is specified in the mcf file. Alternately, the **build_cat** command (see **build_cat**(1M)) may be used to build the initial catalog.

To initialize a catalog with 32 slots run:

```
build_cat − /tmp/catalog_file < /dev/null
```

then move /tmp/catalog_file to the path pointed to in the mcf file for the historian. Or, you can create a file with the list of volumes and supply it as input to **build_cat** (see **build_cat**(1M)) for the format of the input file.

FILES

mcf	The configuration file for the Sun SAM-FS and SAM-QFS software.
defaults.conf	Default information.
/var/opt/SUNWsamfs/catalog/historian	Default historian catalog file.

SEE ALSO

build_cat(1M), **dump_cat**(1M), **export**(1M), **defaults.conf(4),** **mcf**(4), **sam-robotsd**(1M), **robottool**(1M)

NAME

ibm3494 − The IBM3494 interface through lmcpd

AVAILABILITY

SUNWsamfs

DESCRIPTION

ibm3494 is the Sun SAM-FS and SAM-QFS interface to the IBM 3494 library. This interface utilizes the **lmcpd** interface supplied by IBM. For more information on configuration and interfacing the IBM libraries, see the documentation supplied with the IBM hardware and for **lmcpd**.

CONFIGURATION

It is assumed that the site has the **lmcpd** daemon configured and operating with the 3494 library.

The "equipment identifier" field in the **mcf** file, (see **mcf**(4)), is the full path name to a parameters file used by **ibm3494**. This file consists of *keyword* = *value* and *path_name* = *value* pairs. All keyword/path_name/values are case-sensitive.

The keywords are:

name This is the name assigned by the system administrator and configured in the **/etc/ibmatl.conf** file and the symbolic name of the library. This parameter must be supplied, there is no default.

category The category is a hex number between 0x0001 and 0xfeff. Media controlled by Sun SAM-FS or SAM-QFS will have its category set to this value. The default for category is 4.

access Access to the library may be **shared** or **private**. If **private**, then any media imported into the library (category = 0xff00) will be added to the catalog and its category will be changed to that specified by **category** above. If **shared**, then the **import** command (see **import**(1M)) will have to be used to add media to the catalog. The default for access is **private**.

device_path_name

There is one *device_path_name* entry for every drive in the library attached to this machine. The *device_path_name* is the path to the device. Following the *device_path_name* is the "device number" as described in the IBM documentation. The system adminsistrator can determine this number by running the IBM supplied utility **mtlib**. Following the *device number* is the *shared* parameter. This parameter is optional and is used to indicate the drive is shared with other Sun SAM-FS or SAM-QFS servers. See examples below.

EXAMPLE

The example uses the following file and information obtained from the IBM supplied utility mtlib. Both are documented in the materials supplied by IBM.

```
#
# This is file: /etc/ibmatl.conf
# Set this file up according the documentation supplied by IBM.
3493a   198.174.196.50  test1
```

After **lmcpd** is running, run **mtlib** to get the device numbers.

```
mtlib -l 3493a -D
  0, 00145340 003590B1A00
  1, 00145350 003590B1A01
```

Here is a sample parameters file and **mcf** entries for a IBM 3494 library.

```
#
# This is file: /etc/fs/samfs/ibm60
#
name = 3493a                    # From /etc/ibmatl.conf
/dev/rmt/1bn = 00145340         # From mtlib output
```

```
/dev/rmt/2bn = 00145350 shared   # From mtlib output
access=private
category = 5

# The mcf file entries.
#
# IBM 3494 library
#
/etc/opt/SUNWsamfs/ibm60 60      im      ibm3494e - ibmcat
/dev/rmt/1bn            61      tp      ibm3494e
/dev/rmt/2bn            62      tp      ibm3494e
```

IMPORT/EXPORT

Import media into the library by placing the new media into the I/O slots and closing the door. The library will lock the door and move the media into the storage area. Only 100 volumes can be imported at one time. If you are running with access=private, the library will inform the daemon as the media is moved and the media will be added to the catalog. If running with access=shared, then an **import** (see **import**(1M)) command will need to be executed to add the media to the catalog.

Exporting media (in all modes) is performed by the **export** (see **export**(1M)) command. This command will move the media to the I/O area and the output mode light on the operator panel will light. The operator can then remove the media from the I/O area.

CATALOG

If running with access=shared then a catalog will need to be built before starting samfs. There are two utilities used to maintain the library catalog. **build_cat** (see **build_cat**(1M)) is used to build the catalog. **dump_cat** (see **dump_cat**(1M)) and **build_cat** together are used to change the size of the catalog.

To initialize a catalog with 1000 slots run:

```
build_cat /tmp/catalog_file < /dev/null
```

then move **/tmp/catalog_file** to the path pointed to in the **mcf** file for this library. (In this case /var/opt/SUNWsamfs/catalog/ibmcat). Use **import** to populate the catalog with the volumes. Or, you can create a file with the list of volumes and supply it as input to **build_cat** (see **build_cat**(1M) for the format of the input file).

If the size of the catalog needs to be increased, execute something like:

```
dump_cat file1 | build_cat -s 2000 /tmp/file2
```

This would create a new catalog file (**/tmp/file2**) with room for 2000 entries and initialize it with the entries from file1. This can only be done when the Sun SAM-FS or SAM-QFS software is not running and **sam-initd** has been shutdown (see **sam-initd**(1M)).

FILES

mcf	The configuration file for the Sun SAM-FS and SAM-QFS software.
/etc/ibmatl.conf	Configuration file used by **lcmpd**.
/opt/SUNWsamfs/lib/libibmlmcp.so	A shared object version of the runtime library supplied by IBM

SEE ALSO

build_cat(1M), **dump_cat**(1M), **export**(1M), **import**(1M), **mcf**(4), **sam-robotsd**(1M), **robottool**(1M)

NAME

ibm3584 − Describes using the IBM 3584 UltraScalable Tape Library with Sun SAM-FS or SAM-QFS software

AVAILABILITY

SUNWsamfs

DESCRIPTION

The IBM 3584 UltraScalable tape library can be used by Sun SAM-FS and SAM-QFS software with few modifications to its typical operations. This man page describes the steps you need to take to use this library effectively. The following topics are described:

- Cleaning
- Using the IBM 3584 tape library’s partitioning feature

CLEANING

The IBM 3584 UltraScalable Tape Library can be used in a SAM-FS environment, but you need to disable automatic cleaning and enable hosted cleaning. No other modifications need to be made to this library’s default configuration.

Host cleaning enables the host to detect the need to clean an Ultrium Tape Drive and to control the cleaning process. Host cleaning with a cleaning cartridge is only supported when you disable automatic cleaning and only for the logical library in which each cleaning cartridge is stored. When you enable automatic cleaning, or when the cleaning cartridge is stored in a different logical library, the host application does not have access to the cleaning cartridge.

When automatic cleaning is disabled, the library continues to detect the need to clean a tape drive. When the need is detected, the library displays the physical location of the drive in the following message:

CLEAN [Fx,Rzz]

The preceding message is interpreted as follows:

- **F** represents the frame, and **x** represents its number
- **R** represents the row, and **xx** represents its number

The message clears after you clean the drive by using the supported cleaning method. The cleaning cycle takes less than 2 minutes.

When you enable or disable automatic cleaning, the selected setting is stored in nonvolatile memory and becomes the default during later power-on cycles.

To disable automatic cleaning, perform the following steps:

1. Ensure that a cleaning cartridge is loaded in the library.
2. From the library’s activity screen, press **MENU**. The main menu displays, as follows:

```
-----------------------------------------------------------------------
Main Menu          Panel 0002

Library Status
Manual Operations
Settings
Usage Statistics
Vital Product Data
Service

[BACK]  [UP]  [DOWN]  [ENTER]
-----------------------------------------------------------------------
```

3. Press **UP** and **DOWN** to highlight Settings. Press **ENTER**. The **Settings** menu displays, as follows:

```
--------------------------------------------------------------------------
Settings              Panel 0100

Configuration
Cleaning Mode
Display/Change SCSI IDs
Add/Remove Control Paths
Date/Time
Sounds

[BACK]  [UP]  [DOWN]  [ENTER]
--------------------------------------------------------------------------
```

4. Press **UP** or **DOWN** to highlight **Cleaning Mode**. Press **ENTER**. The **Cleaning Mode** screen displays and indicates whether automatic cleaning is currently enabled or disabled.

```
--------------------------------------------------------------------------
Cleaning Mode         Panel 0110

Automatic Cleaning is ENABLED
Disable Automatic Cleaning

[BACK]  [ENTER]
--------------------------------------------------------------------------
```

5. The **ENTER** key acts as a toggle switch for the two choices. Press **ENTER** until **Disable Automatic Cleaning** is highlighted. You should receive the following message:

   If you continue you will set the Automatic Cleaning Mode to DISABLED. If you disable automatic cleaning you should ensure that each logical library has at least one cleaning cartridge since host-initiated cleaning can not use a cleaning cartridge located in a different logical library. Do you want to continue?

6. Press **YES** to disable automatic cleaning. The **Cleaning Mode** screen redisplays with the new setting.

7. Press **BACK** until you return to the **Activity** screen from step 1.

PARTITIONING

If your IBM 3584 tape library contains 2 or more drives, it can be partitioned into 2 or more logical libraries. If you have partitioned this library, make sure that it is operating as you configured it prior to installing any SAM-FS software. For more information on partitioning this library, see your IBM documentation. This subsection describes aspects of using the partitioning feature with the Sun SAM-FS and SAM-QFS software.

When a cartridge is exported (as opposed to being placed in the drawer by a human), only the partition from which it was exported can access that drawer slot. If the cartridge is removed and re-inserted by a human, it is accessable to any/all partitions. The act of removal referred to in this subsection consists of the following steps:

1. Open door.
2. Remove cartridge(s).
3. Close door.
4. Wait for door to lock and then unlock.
5. Open door.

6. Replace cartridge(s).
7. Close door.

NOTES

Much of the text on this man page was derived from the *IBM 3584 UltraScalable Tape Library Planning and Operator Guide*, IBM publication GA32-0408-01, copyright IBM Corporation 2000.

SEE ALSO

IBM 3584 UltraScalable Tape Library Planning and Operator Guide, IBM publication GA32-0408-01.

http://www.ibm.com/storage/hardsoft/tape/lto/3584

NAME

sam-remote, sam-clientd, sam-serverd − Describes the Sun SAM-Remote interface and daemons

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-serverd *mshmid pshmid equip*

/opt/SUNWsamfs/sbin/sam-clientd *mshmid pshmid equip*

AVAILABILITY

SUNWsamfs

DESCRIPTION

The Sun SAM-Remote client and server software allows automated libraries to be shared among the Solaris systems in a Sun SAM-FS or SAM-QFS environment. Sun SAM-Remote allows you to configure multiple storage clients that archive and stage files from a centralized optical and/or tape library. This environment also allows you to make multiple archive copies on various media housed in multiple libraries.

DAEMONS

The Sun SAM-Remote daemons, **sam-serverd** and **sam-clientd**, control Sun SAM-Remote. The **sam-robotsd** daemon starts the **sam-serverd** and **sam-clientd** daemons. The identifiers associated with these daemons are as follows:

mshmid	The identifier of the master shared memory segment created by **sam-initd**.
pshmid	The identifier of the preview shared memory segment created by **sam-initd**.
equip	The equipment number of the device.

For more information on the **sam-robotsd** or **sam-initd** daemons, see the **sam-robotsd**(1M) or **sam-initd**(1M) man pages.

CONFIGURATION

Configuring the Sun SAM-Remote client and server software involves adding lines to the **mcf** file on both the system to be used as the Sun SAM-Remote client and on the system to be used as the Sun SAM-Remote server.

In addition, a client configuration file must be created on the Sun SAM-Remote client, and a server configuration file must be created on the Sun SAM-Remote server.

Up to ten clients can be configured per server. For a complete description of the Sun SAM-Remote configuration process, see the *Sun SAM-Remote Administrator's Guide*, publication SG-0003.

FILES

mcf	The master configuration file for Sun SAM-FS, SAM-QFS, the SAM-Remote client, and the SAM-Remote server.
/opt/SUNWsamfs/lib/librmtsam.so	The Sun SAM-Remote shared object library.

SEE ALSO

sam-initd(1M), **sam-robotsd**(1M).

mcf(4).

Sun SAM-Remote Administrator's Guide, publication SG-0003.

NAME

samst − Driver for SCSI media changers and optical drives

SYNOPSIS

samst@*target*,*lun*:*a*

AVAILABILITY

SUNWsamfs

DESCRIPTION

This driver handles embedded SCSI-2 and CCS-compatible SCSI media changers, optical drives, CD-ROM drives and non-motion I/O for tape drives

The type of device is determined using the SCSI inquiry command.

The only I/O supported for optical devices is ‘‘raw’’. **samst** supports 512-, 1024-, 2048-, and 4096-byte sector sizes for optical media. The names of the raw files are found in **/dev/samst**.

Special handling during open

If *O_NDELAY* or *O_NONBLOCK* is specified on the open, then the device does not have to be in the ready state for the open to succeed. This allows the opening of a device for initialization or to check the media type.

ERRORS

EACCES	Permission denied.
EBUSY	The device was opened exclusively by another thread.
EFAULT	The argument was a bad address.
EINVAL	Invalid argument.
EIO	An I/O error occurred.
ENOTTY	This indicates that the device does not support the requested ioctl function.
ENXIO	During opening, the device did not exist.

FILES

/kernel/drv/samst.conf	driver configuration file
/dev/samst/c*n***t***n***u***n*	raw files

where:

c*n*	controller *n*
t*n*	SCSI target id *n* (0-6)
u*n*	SCSI LUN *n* (0-7)

SEE ALSO

driver.conf(4), **samdev**(1M)

ANSI Small Computer System Interface-2 (SCSI-2)

DIAGNOSTICS

Error for command ’<command name>’ Error Level: Fatal
Requested Block <n>, Error Block: <m>
Sense Key: <sense key name>
Vendor ’<vendor name>’: ASC = 0x<a> (<ASC name>), ASCQ = 0x<b>, FRU = 0x<c>

The command indicated by <command name> failed. The Requested Block is the block where the transfer started and the Error Block is the block that caused the error. Sense Key, ASC, and ASCQ information is returned by the target in response to a request sense command.

Check Condition on REQUEST SENSE

A REQUEST SENSE command completed with a check condition. The original command will be retried a number of times.

Not enough sense information
The request sense data was less than expected.

Request Sense couldn’t get sense data
The REQUEST SENSE command did not transfer any data.

Reservation Conflict
The drive was reserved by another initiator.

SCSI transport failed: reason ’xxxx’ : {**retrying**|**giving up**}
The host adapter has failed to transport a command to the target for the reason stated. The driver will either retry the command or, ultimately, give up.

Unhandled Sense Key <**n**>
The REQUEST SENSE data included an invalid sense key.

Unit not Ready. Additional sense code 0x<**n**>
The drive is not ready.

device busy too long
The drive returned busy during a number of retries.

incomplete read/write - retrying/giving up
There was a residue after the command completed normally.

logical unit not ready
The unit is not ready.

NOTES

This driver can accept removable media devices that identify themselves as "direct access" by setting the variable **samst_direct** to non-zero. You may do this using the "set" command in the **/etc/system** file (see **system**(4)).

Whenever a new version of Sun SAM-FS or SAM-QFS is installed, the existing **samst.conf** file is copied to **samst.conf.MMDDYY** for reference and backup purposes.

NAME

sony − Attaches a Sony network-attached tape library through the DZC-8000S interface

AVAILABILITY

SUNWsamfs

DESCRIPTION

The **SUNWsamfs** package contains the Sun SAM-FS and SAM-QFS interface to a Sony network-attached library. This interface utilizes the DZC-8000S 3.01 interface supplied by Sony. For more information on DZC-8000S, see the *Sony PetaSite Application Interface DZC-8000S* manual. This manual is supplied by Sony.

CONFIGURATION

It is assumed that the site has the PetaSite Controller (PSC) configured and operating with the Sony library. In the Execute Mode of the PSC configuration, the following must be set to on:

- Thread With Load
- Unthread with Fast Unload
- Unthread with Eject
- Wait for Drive Use

The **Equipment Identifier** field in the Sun SAM-FS or SAM-QFS **mcf** file must be the full path name to a Sony parameters file. For more information on specifying a parameters file, see the **mcf**(4) man page.

The parameters file consists of a list of *keyword* **=** *value* pairs. All *keyword* and *value* specifications are case-sensitive and must be entered as shown on this man page. The *keyword* and *value* specifications are as follows:

userid = *userid*

Identifies the user during initialization of the Sony library functions. The *userid* values can be specified in hexadecimal or decimal. The valid range is from 0 to PSCUSERIDMAX(0xfff), which is 0 <= *userid* <= 65535 (decimal) or 0 <= *userid* <= 0xffff (hexadecimal). This is a required parameter.

server = *serverid*

Specifies the host name of the server running the PSC server code. This is a required parameter.

sonydrive *binnum* = *path* [**shared**]

Specifies characteristics of the tape drive. There must be one **sonydrive** line for every drive assigned to Sun SAM-FS or SAM-QFS in the **mcf** file.

The following arguments follow the **sonydrive** keyword:

binnum	Specifies the bin number assigned to the drive in the PSC configuration. The bin number can be identified using the PSC Monitoring and Maintenance terminal. This is a required argument.
path	Specifies the Solaris **/dev/rmt/** path name to the device. The *path* must match the **Equipment Identifier** of an entry in the **mcf** file. This is a required argument.
shared	Specifies that this drive is shared with other processes. For example, this drive can be shared between multiple Sun SAM-FS or SAM-QFS servers. This is an optional argument.

EXAMPLE

The following example shows the configuration files for a network-attached Sony library with Sony DTF tapes.

Here are the sample entries in the **mcf** file. The catalog file is placed in the default directory, which is **/var/opt/SUNWsamfs/catalog**.

The **mcf** file is as follows:

```
#
# This is the file: /etc/opt/SUNWsamfs/mcf
# This file shows sample mcf entries for a Sony network-attached
# robot with Sony DTF tapes.
#
/etc/opt/SUNWsamfs/sonyfile 50 pe sony50 on /var/opt/SUNWsamfs/sony50cat
/dev/rmt/0cbn              51 so sony50 on
/dev/rmt/1cbn              52 so sony50 on
```

The parameters file for a Sony library supporting Sony DTF tapes is as follows:

```
#
# This is file: /etc/opt/SUNWsamfs/sonyfile
#
# The userid identifies the user during initialization of
# the PetaSite library functions. Valid IDs are 0 to
# PSCUSERIDMAX(0xfff).
#
userid = 65533
#
# The server identifies the hostname for the server running
# the DZC-8000S server code.
#
server = europa
#
# The sonydrive bin number 1001 is from the PSC configuration file
#
sonydrive 1001 = /dev/rmt/0cbn shared  # a comment
#
# The sonydrive bin number 1002 is from the PSC configuration file
#
sonydrive 1002 = /dev/rmt/1cbn         # a comment
```

IMPORT/EXPORT

The physical adding and removing of cartridges in a Sony network-attached library is accomplished using the PSC utilities. The **import**(1M) and **export**(1M) commands and the **libmgr**(1M) and **robottool**(1M) import and export menus affect only the library catalog. Therefore, importing and exporting cartridges with the Sony network-attached library proceeds according to the following two-step process:

1. Physically import or export the cartridge using the PSC software.
2. Virtually update the library catalog using the Sun SAM-FS or SAM-QFS import/export utilities.

The **import**(1M) command has an optional **−v** option that allows you to specify the VSN to be added. The **samsony** package verifies that PSC knows about the VSN before updating the catalog with the new entry. The **export**(1M) command removes the entry from the catalog.

CATALOG

There are several methods for building a catalog for a Sony network-attached library. You should use the method that best suits your system configuration, typically depending on the size of the catalog that is needed.

Method 1: Create a catalog with existing VSN entries. You can build a catalog that contains entries for many tapes by using the **build_cat**(1M) command. As input to the **build_cat**(1M) command, you need to create a file that contains the slot number, VSN, bar code label, and media type. For example, the file **input_vsns** follows:

```
0  "SEG001"  "SEG001"  so
1  "SEG002"  "SEG002"  so
2  TEST1     TEST1     so
3  TEST2     TEST2     so
```

The **input_vsns** file can be used as input to the **build_cat**(1M) command as follows:

```
build_cat input_vsns /var/opt/SUNWsamfs/sony50cat
```

Method 2: Create a null catalog and import VSN entries. You can create an empty catalog and populate it. To create a catalog that will accommodate 1000 slots, use the **build_cat**(1M) command as follows:

```
build_cat -s 1000 /dev/null /var/opt/SUNWsamfs/catalog/sony50cat
```

Use the **import**(1M) command to add VSNs to this catalog, as follows:

```
import -v "SEG005" 50
```

Method 3: Use the default catalog and import VSN entries. If a catalog path name is not specified in the **mcf** file, a default catalog is created in **/var/opt/SUNWsamfs/catalog/***family_set_name* when Sun SAM-FS or SAM-QFS is initialized. Following initialization, you must import VSN entries to this catalog by using the **import** command as follows:

```
import -v "SEG005" 50
```

In the previous **import**(1M) command, **50** is the **Equipment Ordinal** of the library as specified in the **mcf** file.

FILES

mcf	The configuration file for the Sun SAM-FS and SAM-QFS software.
/opt/SUNWsamfs/lib/libpsc.so	The PSC library supplied by Sony.

SEE ALSO

build_cat(1M), **dump_cat**(1M), **export**(1M), **import**(1M), **libmgr**(1M), **sam-robotsd**(1M).

mcf(4).

NAME

ssi.sh − The configuration file for the StorageTek (STK) Client System Interface CSI.

AVAILABILITY

SUNWsamfs

DESCRIPTION

ssi.sh is a script that allows users to select values for several dynamic environment variables used by the CSI. The STK API code defines default values for these variables if they are not defined dynamically. To allow the most flexibility in setting these variables, a shell script **/etc/opt/SUNWsamfs/ssi.sh**, is used by the stk daemon (see **sam-stkd**(1M)), to start the ssi_so. In general, most sites do not need to change the variables within this script.

CONFIGURATION

An example script can be found in **/opt/SUNWsamfs/examples/ssi.sh**. This script, or a user created script, must be copied to **/etc/opt/SUNWsamfs/ssi.sh**

It is assumed that the site has the server daemons (CSI and ACSLM) configured and operating with the STK library.

The following environment variables are defined in the shell script supplied in **/opt/SUNWsamfs/examples/ssi.sh**:

CSI_TCP_RPCSERVICE
This variable is used to define whether the CSI operates as a TCP RPC Server.

CSI_UDP_RPCSERVICE
This variable is used to define whether the CSI operates as a UDP RPC server.

The CSI can operate as a TCP and a UDP server simultaneously.

CSI_CONNECT_AGETIME
This defines the value of the maximum age of pending requests in the CSI's request queue. This variable is accessed as a "C" character array datatype, expressed as an integer number of seconds. A value of 172800 indicates two days.

Messages older than this value are removed from the queue and the CSI sends an entry to the Event Log when this happens.

CSI_RETRY_TIMEOUT
This defines the minimum amount of time, in seconds, that the CSI will wait between attempts to establish a network connection.

CSI_RETRY_TRIES
This variable defines the number of attempts the CSI will make to transmit a message. Pending messages are discarded if a connection cannot be established within the defined number of tries.

The CSI_RETRY_TIMEOUT and CSI_RETRY_TRIES together determine the minimum total time the CSI will attempt to send a message.

SEE ALSO

sam-robotsd(1M), **stk**(7), **ssi_so**(7)

NAME

ssi_so − The StorageTek (STK) ACSAPI client daemon.

AVAILABILITY

SUNWsamfs

DESCRIPTION

ssi_so is a shared object version of the SSI daemon supplied by STK. This daemon is the interface that the *samfs* uses (see **stk**(7)) to communicate with the ACSLM.

The ssi needs a number of parameters set to communicate with the ACSLM. These parameters are set through shell environment variables. To allow the most flexibility in setting these variables, a shell script (**/opt/SUNWsamfs/sbin/ssi.sh**) is used by the stk daemon (see **sam-stkd**(1M)), to start a ssi_so daemon. In general, most sites should not need to change the variables within this script.

SEE ALSO

sam-robotsd(1M), **stk**(7)

NAME

stk − The StorageTek interface through ACSAPI

AVAILABILITY

SUNWsamfs

DESCRIPTION

stk is the Sun SAM-FS and SAM-QFS interface to the StorageTek libraries. This interface utilizes the ACSAPI interface supplied by StorageTek. The SUNWsamfs package installs the libraries and daemons for the client side of the API. For more information on ACSAPI and interfacing the StorageTek libraries, see the documentation supplied with the StorageTek hardware and server side daemons.

CONFIGURATION

It is assumed that the site has the server daemons (CSI and ACSLM) configured and operating with the StorageTek library.

The **equipment identifier** field in the **mcf** file, (see **mcf**(4)), is the full path name to a parameters file used by **stk**. This file consists of *keyword* = *value* and *path_name* = *value* pairs. All *keyword*, *path_name*, and *value* entries are case-sensitive.

The *keywords* are:

access This is the *user_id* used by this client for access control. If this parameter is not supplied, the access control string will be a null string (no *user_id*).

hostname
This is the *hostname* for the server that is running ACSLS. If the hostname is not supplied, the default will be localhost. All sites should set this value.

portnum
This is the *portnum* for SSI services on the server that is running ACSLS. If the port number is not supplied, the default is 50004. Please note that if you are running co-hosted ACSLS 5.3 or higher, the default value does not work (try a higher port number, like 50014). If you are running multiple connections to ACSLS servers, then the port number for each **stk** configuration file needs to be unique (for example, 50014 in one, 50015 in the next, etc.).

capacity This is used to set the capacity of the media supported by the StorageTek. The parameter to **capacity** is a comma separated list of *index* = *value* pairs enclosed in parentheses. *index* is the index into the media_type file (supplied by StorageTek and located on the ACS system) and *value* is the capacity of that media type in units of 1024 bytes. You should only need to supply this entry if the ACS is not returning the correct media type or new media types have been added. Sun SAM-FS and SAM-QFS have defaults for index values that were current at the time of release. Generally, it is necessary to supply an *index* only for new cartridge types. For the capacity of each cartridge type, see the *Sun QFS and Sun SAM-FS Storage and Archive Management Guide*.

device_path_name
There is one *device_path_name* entry for every drive attached to this client. The *device_path_name* is the path to the device on the client. Following the *device_path_name* is the description of this drive in terms of the StorageTek library. This description starts with an open parenthesis followed by 4 *keyword* = *value* pairs followed by a close parenthesis. The *keyword* = *value* pairs between the parentheses may be separated by a comma (,), a colon (:) or by white space. Following the close parenthesis is an optional keyword used by Sun SAM-FS and SAM-QFS software to designate when a drive is shared with other Sun SAM-FS and SAM-QFS servers. The keyword identifiers and their meanings are as follows:

acs is the ACS number for this drive as configured in the StorageTek library.

lsm is the LSM number for this drive as configured in the StorageTek library.

panel is the PANEL number for this drive as configured in the StorageTek library.

drive is the DRIVE number for this drive as configured in the StorageTek library.

shared The shared keyword follows the close parenthesis. This keyword is optional and is used to indicate the drive is shared with other Sun SAM-FS and SAM-QFS servers.

EXAMPLE

Here is a sample parameters file and **mcf** entries for a StorageTek library:

```
#
# This is file: /etc/opt/SUNWsamfs/stk50
#
hostname = acsls_server_name
portnum = 50004
access = some_user  # No white space allowed in the user_id field
/dev/rmt/0cbn = (acs=0, lsm=1, panel=0, drive=1)         #a comment
/dev/rmt/1cbn = (acs=0, lsm=1, panel=0, drive=2) shared  #a comment
capacity = (0=215040, 1=819200, 5=10485760)
```

The **mcf** file entries that reference this configuration file are:

```
#
# Sample mcf file entries for a StorageTek library
#
/etc/opt/SUNWsamfs/stk50      50  sk  sk50   - /var/opt/SUNWsamfs/catalog/sk50
/dev/rmt/0cbn                 51  st  sk50   -
/dev/rmt/1cbn                 52  st  sk50   -
```

IMPORT/EXPORT

Since the physical adding and removing of cartridges in the StorageTek library is done with ACSLM utilities, the **import/export** commands and GUI buttons will only affect the library catalog. The **import** command has optional parameters for supplying a single volume to be added or to add a number of volumes from a pool (see **import**(1M)). **export** (see **export**(1M)) will remove an entry from the catalog.

CATALOG

The Sun SAM-FS and Sun SAM-QFS systems automatically build a library catalog for a StorageTek automated library. However, you must populate the library catalog. For information on populating the library catalog, see the *Sun QFS and Sun SAM-FS Storage and Archive Management Guide*.

FILES

mcf	The configuration file for the Sun SAM-FS and SAM-QFS software.
/opt/SUNWsamfs/sbin/ssi.sh	A shell script used to start **ssi_so**.
/opt/SUNWsamfs/sbin/ssi_so	A shared object version of the SSI daemon supplied by StorageTek.
/opt/SUNWsamfs/lib/stk/*	The libraries needed by the API interface supplied by StorageTek.
/opt/SUNWsamfs/sbin/stk_helper	A program to issue commands for the StorageTek ACSAPI

SEE ALSO

build_cat(1M), **dump_cat**(1M), **export**(1M), **import**(1M), **sam-robotsd**(1M), **robottool**(1M).

mcf(4).

ssi_so(7).

Sun QFS and Sun SAM-FS Storage and Archive Management Guide