NAME

archive − Set archive attributes and archive files

SYNOPSIS

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

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

AVAILABILITY

LSCsamfs

DESCRIPTION

archive sets archive attributes on and archives one or more files. By default, a file is archived some time after its creation. (Default archive operation is configured by the system administrator.) If neither −**n** nor −**d** is specified the files are marked to be archived immediately.

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

OPTIONS

−**d** Return the archive attributes on the file to the default. When this option is specified attributes are first reset to the default, then other attribute-setting options are processed. The only action taken is that attributes are reset; no archiving is done.

−**f** Do not report errors.

−**n** Specifies that this file never be archived. Only the super-user can set this attribute on a file. When this option is specified, the only action taken is that the attribute is set. This attribute cannot be set on a file that has either the checksum generate or use (**ssum** −**g** or **ssum** −**u**) bit set.

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

SEE ALSO

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

NAME

release − Release disk space and set 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

LSCsamfs

DESCRIPTION

release sets release attributes of 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 filesystem's high water mark is reached.

If any of the −**a**, −**d**, −**n**, −**p**, or −**s** options is 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

−**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. When −**d** is specified attributes are first reset to the default, then other attribute-setting options are processed. If the *partial* attribute is reset, all blocks are released for an offline regular file.

−**f** Do not report errors.

−**n** Specify that the disk space for this file never be released. Only the super-user can set this attribute on a file. Not valid with −**a** or −**p**.

−**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. By default, the minimum size of the portion retained on disk is the mount_samfs parameter *partial*; this is adjustable using the −**s** option. 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.

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 command *stage -p* can be used to stage the partial blocks to the disk.

−**s** *partial_size*
Set the *partial* attribute on the file so that, when the file's disk space is released, the first *partial_size* kilobytes of that disk space will be retained. Not valid with release −**n** or stage −**n**. Also not valid with either of the checksum *generate* or *use* attributes (**ssum** −**g** or **ssum** −**u**). 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)**).

−**r** Recursively performs the operation (releasing disk space or setting release 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 a release will be attempted.

SEE ALSO

archive(1), **stage**(1), **ssum**(1), **mount_samfs**(1M)

NAME

request − Create a removable-media file

SYNOPSIS

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

AVAILABILITY

LSCsamfs

DESCRIPTION

request creates a removable-media file allowing access to either tape or optical disk. The media is specified by *media* (see **media**(5)). *vsn* is the Volume Serial Name of the media. For tape, each *vsn* is limited to 6 characters. For optical media the limit is 31 characters. Multiple *vsns* may be specified by separating them with "/". *file* is the file path name of the removable-media file to be created. The file must reside on a SAM-FS 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 SAM-FS 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"). Note that SAM-FS 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.

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.

−**v** *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 -p 0x286 lt YYY /sam1/xxx
```

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.

SEE ALSO

basename(1), **open**(2), **media**(5)

NAME

setfa − set file attributes

SYNOPSIS

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

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

AVAILABILITY

LSCsamfs

DESCRIPTION

setfa sets attribute 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 that the direct I/O attribute be 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).

−**g** *stripe_group*
Specifies the number of the striped group where the file is to be preallocated. *stripe_group* is a number 0 .. 127, and must be a striped_group defined in the filesystem.

−**l** *length*
Specifies the number of bytes to be preallocated to the file. This can only be applied to a file of zero size. This option is ignored for a directory. If an I/O attempts to extend a preallocated file, the caller will get an ENXIO 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)**).

−**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 where attributes are set.

SEE ALSO

stage(1), **archive**(1), **ssum**(1) **mount_samfs**(1M)

NAME

sfind − Search for files in a directory hierarchy

SYNOPSIS

sfind [path...] [expression]

AVAILABILITY

LSCsamfs

DESCRIPTION

This manual page documents the LSC version of the GNU version of **find**. The GNU version of **find** is modified to support the features of the LSC samfs file system. The following added tests reference characteristics of files resident on a samfs file system:

archived, archive_d, archive_n, copies, copy, copy_d, damaged, mt, mt1, mt2, mt3, mt4, offline, online, release_d, release_a, release_n, release_p, rmin, rtime, ssum_g, ssum_u, ssum_v, 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.

-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.

-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.

−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_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 *vsn*
File has an archive copy on the indicated *vsn* for any copy.

−vsn1 *vsn*

−vsn2 *vsn*

−vsn3 *vsn*

−vsn4 *vsn*
File has an archive copy on the indicated *vsn* 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
```

NAME

sls − List contents of directories

SYNOPSIS

sls [−abcdfgiklmnpqrstuxABCDFGLNQRSUX12] [−w cols] [−T cols] [−I pattern] [−−all] [−−escape] [−−directory] [−−inode] [−−kilobytes] [−−numeric-uid-gid] [--no-group] [−−hide-control-chars] [−−reverse] [−−size] [−−width=cols] [−−tabsize=cols] [−−almost-all] [−−ignore-backups] [−−classify] [−−file-type] [−−full-time] [−−ignore=pattern] [−−dereference] [−−literal] [−−quote-name] [−−recursive] [−−sort={none,time,size,extension}] [−−format={long,verbose,commas,across,vertical,single-column,detailed}] [−−time={atime,access,use,ctime,status}] [−−help] [−−version] [path...]

DESCRIPTION

This manual page documents the LSC version of the GNU version of **ls**. The GNU version of **ls** is modified to support the features of the LSC SAM-FS file server. The following added options display characteristics of files resident on a SAM file system:

−D, −−format=detailed , −2

This program lists each given file or directory path. Directory contents are sorted alphabetically. For **ls**, files are by default listed in columns, sorted vertically, if the standard output is a terminal; otherwise they are listed one per line. For **dir**, files are by default listed in columns, sorted vertically. For **vdir**, files are by default listed in long format.

OPTIONS

−a, −−all
List all files in directories, including all files that start with ‘.’.

−b, −−escape
Quote nongraphic characters in file names using alphabetic and octal backslash sequences like those used in C.

−c, −−time=ctime, −−time=status
Sort directory contents according to the files’ status change time instead of the modification time. If the long listing format is being used, print the status change time instead of the modification time.

−d, −−directory
List directories like other files, rather than listing their contents.

−f Do not sort directory contents; list them in whatever order they are stored on the disk. The same as enabling *−a* and *−U* and disabling *−l*, *−s*, and *−t*.

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

−g Ignored; for Unix compatibility.

−i, −−inode
Print the inode number of each file to the left of the file name. If **−2** is selected, the inode number of the directory is printed on the second line. If **−D** is selected, the inode numbers are printed.

−k, −−kilobytes
If file sizes are being listed, print them in kilobytes. This overrides the environment variable POSIXLY_CORRECT.

−l, −−format=long, −−format=verbose
In addition to the name of each file, print 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.

−m, −−format=commas
List files horizontally, with as many as will fit on each line, separated by commas.

−n, −−numeric-uid-gid
List the numeric UID and GID instead of the names.

−p Append a character to each file name indicating the file type.

−q, −−hide-control-chars
Print question marks instead of nongraphic characters in file names.

−r, −−reverse
Sort directory contents in reverse order.

−s, −−size
Print the size of each file in 1K blocks to the left of the file name. If the environment variable POSIXLY_CORRECT is set, 512-byte blocks are used instead.

−t, −−sort=time
Sort directory contents by timestamp instead of alphabetically, with the newest files listed first.

−u, −−time=atime, −−time=access, −−time=use
Sort directory contents according to the files' last access time instead of the modification time. If the long listing format is being used, print the last access time instead of the modification time.

−x, −−format=across, −−format=horizontal
List the files in columns, sorted horizontally.

−A, −−almost-all
List all files in directories, except for '.' and '..'.

−B, −−ignore-backups
Do not list files that end with '~', unless they are given on the command line.

−C, −−format=vertical
List files in columns, sorted vertically.

−D, −−format=detailed
List a detailed description for each file. In addition to listing the long line format (*−l, −−format=long*), additional lines are listed with the file attributes, archive copies, and the times. Removable media files show the media type and the VSN.

sls −D (the detailed line list) prints its output as follows:

```
mickey.gif:
  mode: −rw−r−−r−−  links:   1  owner: root      group: other
  length:    319279  inode:     1407
  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 and the inode number.

The next line shows the file attributes and are formatted with descriptive text. Descriptions are as follows:

archdone; **archdone** indicates that the the archiver has completed processing the file; that is, there is no more work that the archiver can do on a file. Note: **archdone** does *not* indicate that the file has been archived.

archive -n; The file is marked never archive (super-user only).

damaged; The file is damaged

offline; The file is offline

stage -n; The file is marked never stage.

stage -a; The file is marked for associative staging.

release -a; This file is marked for release as soon as 1 copy is made.

release -n; The file is marked for never release.

release -p; The file is marked for partial release. **partial=***n***k** indicates that the first *n* kilobytes of disk space will be retained for this file. **offline**/**online** indicates the first *n* kilobytes of disk space are offline/online.

The archive copy line is displayed only if there is an active or stale copy. An example of the 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 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 the date and time when the archive copy was written to the media.

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. 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 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:

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 SAM-FS attributes were last changed.

creation: Time the file was created.

residence: Time the file changed from offline to online or vice versa.

The checksum line is displayed only if the file has a checksum-related attribute (*generate*, *use*, or *valid*) set. The format of the checksum line is as follows:

checksum: gen use val algo: 1

The above 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.

−F, −−classify
Append a character to each file name indicating the file type. For regular files that are executable, append a '*'. The file type indicators are '/' for directories, '@' for symbolic links, '|' for FIFOs, '=' for sockets, and nothing for regular files.

−G, −−no−group
Inhibit display of group information in a long format directory listing.

−L, −−dereference
List the files linked to by symbolic links instead of listing the contents of the links.

−N, −−literal
Do not quote file names.

−Q, −−quote-name
Enclose file names in double quotes and quote nongraphic characters as in C.

−R, −−recursive
List the contents of all directories recursively.

−S, −−sort=size
Sort directory contents by file size instead of alphabetically, with the largest files listed first.

−U, −−sort=none
Do not sort directory contents; list them in whatever order they are stored on the disk. This option is not called *−f* because the Unix **ls** *−f* option also enables *−a* and disables *−l*, *−s*, and *−t*. It seems useless and ugly to group those unrelated things together in one option. Since this option doesn't do that, it has a different name.

−X, −−sort=extension
Sort directory contents alphabetically by file extension (characters after the last '.'); files with no extension are sorted first.

−1, −−format=single-column
List one file per line.

−2 List in two line long format. In addition to listing the long line format (*−l, −−format=long*), a second line is listed with the file attributes, media requirements, and the creation time. Removable media files show the media type and the VSN. Non-checksum file attributes are formatted as a string of ten characters. The first character is O if the file is offline, or E if the file is damaged. The next nine characters are interpreted as three sets of three bits each. The first three represent the attributes that the **archive** command sets. The next three are set by **release**, and the final three are set by **stage**. An additional set of three bits is displayed following these; this additional set displays the checksum-related file attributes, set by the ssum command. The letter(s) displayed in these sets correspond to the options supplied to the respective commands.

sls −2 (the 2 line list) prints its output as follows:

```
-rwxrwxrwx    1 smith    dev      10876   May 16 9:42 part2
O----pan-- g-v lt
```

The first line of output is identical to that printed by **ls -l**.

This example 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).

The attributes are indicated as follows:

Dash 0 - offline/damaged status
- **O** the file is offline
- **P** the file is offline with partial online
- **E** the file is damaged
- − the file is on line

Dashes 1-3 - 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 will stay 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.

Dash 3 - not used

Dashes 4-6 - releaser attributes
- **n** Never release the file (only the super user can set this).
- **p** Partial release of the file (first portion is left on disk after release)
- **a** Release as soon as 1 copy is archived.
- − The attribute is not set.

Dash 7 - stage attribute
- **n** Direct access to the removable media (never stage on read).
- − The attribute is not set.

Dashes 8 and 9 are not currently used.

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

−w, −−width cols
Assume the screen is *cols* columns wide. The default is taken from the terminal driver if possible; otherwise the environment variable **COLUMNS** is used if it is set; otherwise the default is 80.

−T, −−tabsize cols
Assume that each tabstop is *cols* columns wide. The default is 8.

−I, −−ignore pattern
Do not list files whose names match the shell pattern *pattern* unless they are given on the command line. As in the shell, an initial '.' in a filename does not match a wildcard at the start of *pattern.*

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

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

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** program.

SEE ALSO

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

NAME

ssum − Set file checksum attributes

SYNOPSIS

ssum [−**d**] [−**f**] [−**g**] [−**u**] [−**a** *algorithm*] *filename*...

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

AVAILABILITY

LSCsamfs

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**). The algorithm to use for generating the value may be specified with the −**a** option. 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.

Files that volume overflow (a file that span multiple volumes) will not have checksums generated.

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.

−**a** *algorithm*

Specifies the algorithm used to generate the 128-bit checksum value. The simple checksum algorithm provided by LSC is the default if no algorithm is specified but the *generate* attribute is set. *algorithm* may be one of the following:

0 Use no algorithm.

1 Use a simple checksum algorithm that also factors in file length.

The checksum value is a 128 bit value. It is initialized with the size of the file. Subsequent 16 byte chunks of the file are added to the checksum value, with no carry

between bytes. The odd bytes at the end of the file are padded on the right with zero bytes and added to the checksum value in the same way.

Since only one checksum algorithm is currently used, the **-a** option may be deleted in future releases.

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

LSCsamfs

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), **mmap**(2), **write**(2)

NAME

archive_audit − Generate an archive audit

SYNOPSIS

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

AVAILABILITY

LSCsamfs

DESCRIPTION

archive_audit generates an audit of all archived files and removable media files (excluding archiver removable media files) in the SAM-FS 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

archiver(1M), **media**(5)

NAME

archiver − SAM-FS file archive daemon

SYNOPSIS

/etc/fs/samfs/archiver
/etc/fs/samfs/archiver [−**A**] [−**f**] [−**l**] [−**n***filesystem*] [−**v**] [*archive_cmd*]

AVAILABILITY

LSCsamfs

DESCRIPTION

archiver is responsible for automatically archiving SAM-FS files. **archiver** is started automatically a SAM-FS file system is mounted. Commands for controlling the archiver are read from the archiver commands file **/etc/fs/samfs/archiver.cmd**. This file does not have to be present for the archiver to execute. In this case, all files on the file system will be archived to the available removable media.

The second form is used to evaluate the archiver commands file *archive_cmd*. No archiving is performed. Information about the archiving operations are written to standard output. This form should be used each time the archiver commands file is changed, because any error found will prevent the archiver from running. If *archive_cmd* is not specified, the file **/etc/fs/samfs/archiver.cmd** will be evaluated. If **archiver.cmd** is not present **archiver** uses defaults to evaluate archiving.

Scheduling archive copies.

Finding files to archive.

Each file system is scanned by an individual 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 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 copy criteria are met, add the file to the archive request (ArchReq) for the Archive Set. The ArchReq contains a 'batch' of files that can be archived together.

If a file is offline, select the VSN to be used as the source for the archive copy. If the file copy is being rearchived, select that VSN.

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 should have been archived. (time_now - modification_time) -

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, begin composing the archive requests.

Composing archive requests.

If the ArchReq requires automatic 'owner' Archive Sets, separate the ArchReq by owner. Enter all the ArchReq-s into the scheduling queue.

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-s to mark the archive file boundaries where the properties differ. Sort the files within the archive file boundaries according to the 'sort' method. Mark the ArchReq as joined and sorted.

For ArchReq-s without a 'join' method required:
Sort the files according to the 'sort' method. Sorting the files will tend to keep the files together in the archive files. Mark the ArchReq as sorted. 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 ArchReq-s. All the online files will be archived together, and the offline files will be together.

If the Archive Set specifies multiple drives, divide the request for multiple drives.

The priority of each ArchReq created during this process is set to the highest file priority in the ArchReq.

Scheduling from the queue.

When an ArchReq is ready to be scheduled to an arcopy, the VSNs are assigned to the candidate ArchReq-s as follows:

The VSN that has most recently been used for the Archive Set is used if there is enough space for the ArchReq.

An ArchReq will be split into additional ArchReq-s as needed if it is too big for one VSN. Additional VSNs available to the Archive Set are selected in order of occurence in the catalog.

An ArchReq with a single file that is too large to fit on one VSN, and is larger than 'ovflmin' will have additional VSNs assigned as required. The additional VSNs are selected in order of decreasing size. This is to minimize the number of VSNs 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 VSN loaded
: the first VSN to be archived to is loaded in a drive

Files off linethe request contains off line files

Multiple archive VSNs
: the file being archived requires more than one VSN

Multiple stage VSNs
: the file being archived is offline on more than one VSN

Queue wait : seconds that the ArchReq has been queued

Stage VSN loaded
: the first VSN that contains offline files is loaded in a drive

Enter each ArchReq into the scheduling queue. When all ArchReq-s are in the queue, the queue is sorted by the scheduling priorityghest to lowest, to determine the order in which the ArchReq-s will be assinged to an arcopy.

Schedule only as many arcopy-s as drives allowed in a robot or allowed by the Archive Set. When all arcopy-s are busy, wait for an arcopy to complete. Repeat the scheduling sequence until all ArchReq-s are processed.

Assigning an ArchReq to an arcopy.

Step through each non-joined 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.

Send the list of VSNs and the list of files to the arcopy.

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**.

Sample default output:

```
Reading "example1.cmd"

Archive media:
default:mo
media:mo archmax:5000000
media:lt archmax:50000000

Archive devices:
device:mo20 drives_available:1 archive_drives:1
device:lt30 drives_available:1 archive_drives:1

Archive file selections:
Filesystem samfs1:
samfs1  File system data
    copy:1  arch_age:240
big  path:. minsize:512000
    copy:1  arch_age:240
all  path:.
    copy:1  arch_age:30


Archive sets:
all
    copy:1 media:mo
big
    copy:1 media:lt
samfs1
    copy:1 media:mo
```

OPTIONS

−A Turn on all list options.

−f List file system content. Sample output:

```
Filesystems:
samfs1 eq:11 mount:/sam1 interval:300 log_file:/tmp/archiver.log
regular files       52,  71.23%      6390780
offline             47,  64.38%      4293620
copy1               52,  71.23%      6390780
copy2                4,   5.48%            8
copy3                4,   5.48%            8
copy4                4,   5.48%            8

directories         21,  28.77%        86016
symbolic links       0,   0.00%
removable media      0,   0.00%
unknown              0,   0.00%
```

Column 2 is the number of files. Column 3 is the percent of the total number of files. Column 3 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:
   mo0002   mo  capacity:  637040 space:  561601
device:lt30 drives_available:1 archive_drives:1
  Catalog:
   LT0001   lt  capacity: 9939712 space: 9938400

.
.
.

Archive sets:
all
    copy:1 media:mo
    VSNs:
      mo0002
big
    copy:1 media:lt
    VSNs:
      LT0001
samfs1
    copy:1 media:mo
    VSNs:
      mo0002
```

OUTPUT FORMAT

The archiver can produce a log file containing information about files archived and unarchived. Here is an example:

```
A 1997/01/14 10:59:19 mo OPT003 samfs1.1 1a5.1 samfs1 27.1 4096 big/HiRes
A 1997/01/14 10:59:19 mo OPT003 samfs1.1 1a5.a samfs1 34.1 4096 big/QT/Sources
A 1997/01/14 10:59:19 mo OPT003 samfs1.1 1a5.13 samfs1 8.9 4096 big
A 1997/01/14 10:59:19 mo OPT003 samfs1.1 1a5.1c samfs1 33.1 4096 big/QT
A 1997/01/13 16:03:29 lt DLT000 big.1 7eed4.1 samfs1 13.7 477609472 big/f50
1 1997/01/13 16:03:29 lt DLT001 big.1 7fb80.0 samfs1 13.7 516407296 big/f50
2 1997/01/13 16:03:29 lt DLT005 big.1 7eb05.0 samfs1 13.7 505983404 big/f50
```

Field	Description
1	A for archived, 1-7 for additional sections following the A (section 0) entry, R for re-

archived, and 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.

SEE ALSO

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

NAME

arcopy − SAM-FS archive copy daemon

SYNOPSIS

/etc/fs/samfs/arcopy

AVAILABILITY

LSCsamfs

DESCRIPTION

arcopy is responsible for copying **SAM-FS** files to removable media. It is executed by **archiver**(1M). All required information is transmitted to **arcopy** through a pipe.

SEE ALSO

archiver(1M)

NAME

arfind − SAM-FS archive find daemon

SYNOPSIS

/etc/fs/samfs/arfind *file_system*

AVAILABILITY

LSCsamfs

DESCRIPTION

arfind is responsible for finding **samfs** file system files to be archived. It is executed by **archiver**(1M). The only argument is the name of the file system. All other required information is transmitted to **arfind** through a pipe.

SEE ALSO

archiver(1M)

NAME

auditslot − Audit slots in a robot

SYNOPSIS

/opt/LSCsamfs/sbin/auditslot [−**e**] *eq slot [slot...]*

AVAILABILITY

LSCsamfs

DESCRIPTION

auditslot will send a request to the robot specified by the equipment identifier *eq* to audit the media in the specified *slots*. The *slots* must be in use and occupied (that is, the media cannot be mounted in a drive). If *slot* is a two-sided optical media, 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 SAM-FS

SEE ALSO

export(1M), **import**(1M), **move**(1M), **mcf**(4), **robots**(1M)

NAME

build_cat − Build a media changer catalog file

SYNOPSIS

/opt/LSCsamfs/sbin/build_cat [−**s** *size*] [−**t** *media*] < *file* > | - *catalog*

AVAILABILITY

LSCsamfs

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:

slot number
The slot number within the catalog.

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 **media**(5)).

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.

OPTIONS

−**s** *size* Set the size of the catalog to *size* entries. 1000 is the default.

−**t** *media*
Set the media type of the catalog to *media* (see **media**(5). 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 enforcment of media type is performed.

STRANGE MEDIA

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

When building a catalog for strange 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), **media**(5), **robots**(1M)

NAME

chmed − Set or clear library catalog flags and values

SYNOPSIS

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

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

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

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

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

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

/opt/LSCsamfs/sbin/chmed -vsn *media vsn slot eq*

AVAILABILITY

LSCsamfs

WARNING

chmed sets or clears flags and values in a library catalog entry. These values are critical to the operation of SAM-FS 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 1024-byte blocks contained on the medium.

count is the the number of times a medium has been mounted since import, or the number of times a cleaning medium 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	slot contains cleaning medium
E	medium is bad
N	medium is strange
R	medium is read-only (software flag)
U	medium is unavailable (historian only)
W	medium is physically write-protected
b	medium has a bar code
c	medium is scheduled for recycling
i	slot in use
l	medium is labeled
o	slot is occupied
p	high priority medium

media specifies the media type. Valid values include (among others) **mo** and **lt**, for magneto-optical and DLT tape, respectively. See **media**(5) for the complete list of media types supported by SAM-FS.

slot gives the slot number in the given equipment which is to be operated on.

space is the number of 1024-byte blocks remaining unused on the medium.

specifier names the piece of media to be affected by the **chmed** command, in one of two forms: *media vsn* or *slot eq*.

time is the time the medium was last mounted in a drive. Several formats are allowed for *time*. Examples are: 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 medium to be affected.

DESCRIPTION

The first form sets (+*flags*) and the second clears (-*flags*) the flags for a particular VSN.

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

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

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

The final form sets the vsn and media type for a given slot in the specified robot.

STRANGE MEDIA

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

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

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

If you have many pieces of strange media, you can use build_cat to bulk-load a catalog.

EXAMPLES

```
chmed -RW lt TAPE0
chmed +c lt CYCLE
chmed -space 102300 lt TAPE0
chmed -time "Mar 23 10:15" lt TAPE0
chmed -time "Nov 28 1991 10:15" lt TAPE0
chmed -vsn lt TAPE1 5 30
```

SEE ALSO

build_cat(1M), **media**(5), **recycler**(1M), **robottool**(1M), **samu**(1M)

NAME

cleandrive − Clean drive in media changer

SYNOPSIS

/opt/LSCsamfs/sbin/cleandrive *eq*

AVAILABILITY

LSCsamfs

DESCRIPTION

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

SAM-FS supports 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. SAM-FS tracks the number of cleaning cycles used for each cleaning tape. If the media changer supports the export operation, SAM-FS 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 SAM-FS

SEE ALSO

mcf(4), **robots**(1M), **robottool**(1M)

NAME

devicetool − Graphical user interface for managing devices associated with SAM-FS

SYNOPSIS

/opt/LSCsamfs/sbin/devicetool

AVAILABILITY

LSCsamfs

DESCRIPTION

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

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 **SAM-FS**

SEE ALSO

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

NAME

dump_cat − Dump the media changer catalog file in text format

SYNOPSIS

/opt/LSCsamfs/sbin/dump_cat [−**o**] [−**V**] *catalog*

AVAILABILITY

LSCsamfs

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

−**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.

SEE ALSO

build_cat(1M), **robots**(1M)

NAME

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

SYNOPSIS

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

AVAILABILITY

LSCsamfs

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 SAM-FS file system to the sam-init 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 SAM-FS 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/fs/samfs/defaults.conf** file. **dump_log** is not intended for general use, and is provided as a utility to supply LSC, Inc. analysts with troubleshooting information when necessary.

FILES

/var/adm/log/fs_fifo_log
fifo buffer used by SAM-FS

/var/adm/log/fs_ioctl_log
ioctl buffer used by SAM-FS

NAME

export − Export media from a robot

SYNOPSIS

/opt/LSCsamfs/sbin/export *-s slot eq*

/opt/LSCsamfs/sbin/export *-v vsn eq*

AVAILABILITY

LSCsamfs

DESCRIPTION

export sends a request to the media changer specified by *eq* to place the media that belongs in *slot* or is labeled with *vsn* in to the mail-slot of the media changer. Either *slot* or *vsn* must be specified.

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

FILES

mcf	The configuration file for SAM-FS

SEE ALSO

import(1M), **build_cat**(1M), **dump_cat**(1M), **mcf**(4), **robots**(1M)

NAME

gnutar − GNU version of tar

SEE ALSO

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

NAME

import − Import media into robot

SYNOPSIS

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

/opt/LSCsamfs/sbin/import [−**v** *volser*] [−**b** *barcode*] −**m** *mediatype eq*

AVAILABILITY

LSCsamfs

DESCRIPTION

The first form of **import** will send a request to the media changer specified by *eq* to import media. The media will be placed in the first available slot of the catalog. If −**e** is specified all newly added media will be audited with EOD search, updating the catalog with actual capacity and space-remaining values. The −**v** option is only valid for GRAU, STK and IBM3494 (if running in shared mode (see **ibm3494**(7)) and will create a catalog entry with *volser* as the barcode. Physical import and export of media within the GRAU and STK are performed by utilities supplied by the vendor.

For STK only: Take *num* volumes from scratch pool *pool*. If *-l* is specified, list the new vsns on stdout.

The second form can only be used when *eq* is the equipment ordinal of the **historian**(7). This form adds an entry into the historian's catalog for the given *mediatype* and *barcode* or *volser*. At least one of the latter two must be present.

FILES

mcf	The configuration file for SAM-FS

SEE ALSO

export(1M), **mcf**(4), **robots**(1M)

NAME

itemize − Catalog optical disk or jukebox

SYNOPSIS

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

AVAILABILITY

LSCsamfs

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.

slot access_time count use ty vsn

Where *slot* is the slot number within the catalog, *access_time* is the last time this slot 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 slot 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 slot, the status of the slot is labeled, and the VSN is null.

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

NEEDS AUDIT
: The status of the slot 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
slot    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 SAM-FS

SYNOPSIS

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

AVAILABILITY

LSCgui

DESCRIPTION

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

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 slot, 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 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, slot 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 SAM-FS
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

load − Load media into a device

SYNOPSIS

/opt/LSCsamfs/sbin/load [−**w**] −**slot** n | −**vsn** *xxx eq*

AVAILABILITY

LSCsamfs

DESCRIPTION

load requests that the media from slot *n* or vsn *xxx* be loaded into device *eq*. The device specified by *eq* must be a removeable media drive, be in the "unavailable" state (see **set_state**(1M)), and be controlled by a media changer. If *eq* already has media loaded, it will be unloaded and the media put away before the new media is loaded. If *eq* is the equipment ordinal of a media changer, then the vsn is loaded into an available drive in the media changer. The drive that the vsn is loaded into will be chosen by SAM-FS.

Note: Loading media used by SAM-FS for archiving could result in the loss of the data contained on that media. LSC strongly recommends that archive media **NOT** be loaded in this manner.

OPTIONS

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

FILES

mcf The configuration file for SAM-FS

SEE ALSO

unload(1M), **set_state**(1M), **mcf**(4), **robots**(1M)

NAME

mount_samfs − Mount SAM-FS file systems

SYNOPSIS

mount −**F samfs** [*generic_options*] [−**o** *FSType-specific_options*] [−**O**] *special* | *mount_point*

mount −**F samfs** [*generic_options*] [−**o** *FSType-specific_options*] [−**O**] *special mount_point*

AVAILABILITY

LSCsamfs

DESCRIPTION

mount attaches a SAM-FS file system to the file system hierarchy at the *mount_point*, which is the pathname of a directory. If *mount_point* has any contents prior to the **mount** operation, these are hidden until the file system is unmounted.

If the first form of the command is used, **mount** will search **/etc/vfstab** to fill in the missing arguments, including the *FSType-specific_options*. See **mount**(1M).

If the second form of the command is used without any *FSType-specific_options*, the default is **rw**.

OPTIONS

See **mount**(1M) for the list of supported *generic options*.

−**o** Specify SAM-FS file system specific options in a comma-separated list with no intervening spaces. If invalid options are specified, a warning message is printed and the invalid options are ignored. The following options are available:

nosuid By default the file system is mounted with setuid execution allowed. Specifying **nosuid** causes the file system to be mounted with setuid execution disallowed.

forcedirectio If *forcedirectio* is specified, the default I/O mode is direct. This means data is transferred directly between the user's buffer and disk. The *forcedirectio* option should only be selected if the filesystem is used for large block aligned sequential I/O. See man directio(3C), setfa(1), sam_setfa(3) and sam_advise(3). The default I/O mode is buffered (uses the page cache).

high=*n* Set the high water mark for disk cache utilization to *n* percent. When the amount of space used on the disk cache reaches *n* percent, SAM-FS will start the **releaser** daemon (see **releaser**(1M)). The default is 80%.

low=*n* Set 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 **releaser** daemon will stop releasing disk space. The default is 70%.

weight_size=*x*
Set the size-based weighting factor for space releasing (see **releaser**(1M)) to *x*. The default value is 1.00. This is a floating point value. The parameter *weight* means *weight_size* for backwards compatibility.

weight_age=*x*
Set the age-based weighting factor for space releasing (see **releaser**(1M)) to *x*. The default value is 1.00. This is a floating point value.

maxcontig=*n* Set the maximum readahead and writebehind value to *n*. *n* is in units of 16k blocks. Maxcontig specifies the maximum number of bytes that can be read ahead or written behind by the filesystem. *n* is an integer from 1 to 512; the default is 8 (131072 bytes).

partial=*n* Set the default partial-release size for the filesystem to *n* kilobytes. The partial-release size is used to determine how many bytes at the beginning of a file

marked "partial release" should be retained on disk cache when the file is released. The user may override the default on a file-by-file basis by specifying a size when marking a file partial-release. (See **release**(1)). *n* is an integer from 8 to maxpartial; the default is 16.

maxpartial=*n*

Set the maximum partial-release size for the filesystem to *n* kilobytes. The partial-release size cannot be set larger than maxpartial. *n* is an integer from 0 to 102400; the default is 16.

partial_stage=*n*

Set the partial_stage size for the filesystem to *n* kilobytes. For a partial-release file, this value gives the offset in the file past which access will result in the entire file being staged to disk. *n* is an integer from 0 to maxpartial; the default is *partial.*

stage_n_window=*n*

Set the stage_n window size for the filesystem to *n* kilobytes. For a stage_n file, this is the size that is staged in to the disk cache at any one time. *n* is an integer from 64 to 2048000; the default is 256.

wr_throttle=*n*

Set the maximum number of outstanding write bytes to one file to *n* kilobytes. If *n* is set to 0, there is no limit. The default is 1/4 physmem.

notrace | trace

notrace disables filesystem tracing. *trace* enables filesystem tracing. The default is *trace*.

stripe=*n* Set the stripe size for the filesystem to *n* disk allocation units (DAU). *n* is an integer from 0 to 255. The default *n* is 1 on the *ms* equipment type filesystem. The default *n* is 0 on the *ma* equipment type filesystem. If *n* is 0, files are round-drobined on each slice. (See man **mcf**(4) for a description of the *ms* and *ma* equipment type filesystems.)

The following parameters are only supported by the *ma* equipment type filesystem. (See man **mcf**(4) for a description of the *ma* equipment type filesystem.)

qwrite If *qwrite* is specified, the filesystem enables simultaneous reads and writes to the same file from different threads. If *qwrite* is not specified, the filesystem disables simultaneous reads and writes to the same file. This is the mode defined by the UNIX vnode interface standard which gives exclusive access to only one writer while other writers/readers must wait. The *qwrite* option should be selected only if users of the filesystem handle multiple simultaneous transactions to the same file, such as database applications. This option will improve I/O performance by queuing multiple requests at the drive level.

shared_writer

Sets filesystem to type writer. For each filesystem, there should be only 1 system which has it mounted shared_writer. Directories and inodes are always written through to disk if shared_writer is set.

shared_reader

Sets filesystem to type reader. The filesystem must be mounted read-only to set shared_reader. There is no limit to the number of systems which can have the same filesystem mounted with shared_reader. Directories and inodes are always read from disk if shared_reader is set.

FILES

/etc/mnttab	Table of mounted file systems
/etc/vfstab	List of default parameters for each file system

SEE ALSO

mnttab(4), **mount**(1M), **mount**(2), **mountall**(1M), **release**(1), **releaser**(1M), **mcf**(4), **vfstab**(4)

NOTES

If the directory on 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.

NAME

move − Move media in a robot

SYNOPSIS

/opt/LSCsamfs/sbin/move *source-slot destination-slot eq*

AVAILABILITY

LSCsamfs

DESCRIPTION

move will send a request to the robot specified by *eq* to move the media in *source-slot* to *destination-slot*. The *source-slot* must be in use and occupied (that is, the media is not mounted) and *destination-slot* must not be occupied or in use. Some robots do not support moving media between storage slots.

FILES

mcf The configuration file for SAM-FS

SEE ALSO

export(1M), **import**(1M), **mcf**(4), **robots**(1M)

NAME

notify − SAM-FS exception notification daemon

SYNOPSIS

/etc/fs/samfs/notify

AVAILABILITY

LSCsamfs

DESCRIPTION

notify is a wrapper for user configureable scripts to handle unusual circumstances that arise in SAM-FS applications. All required information is transmitted to notify through a pipe

SEE ALSO

archiver(1M), **archiver.cmd**(4) **notify.sh**(4)

NAME

previewtool − Graphical user interface for displaying pending mount requests associated with SAM-FS

SYNOPSIS

/opt/LSCsamfs/sbin/previewtool

AVAILABILITY

LSCsamfs

DESCRIPTION

previewtool presents a graphical user interface for viewing pending mount requests for VSNs associated with SAM-FS. 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 SAM-FS

SEE ALSO

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

NAME

rearch − Mark archive entries to be rearchived.

SYNOPSIS

rearch [−**f**] [−**o**] [−**c** *n*] [−**m** *media*] [−**v** *vv...*] *filename...*

rearch [−**f**] [−**o**] [−**c** *n*] [−**m** *media*] [−**v** *vv...*] −**r** *dirname...* [*filename...*]

AVAILABILITY

LSCsamfs

DESCRIPTION

rearch marks archive entries for one or more files or directories to be 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 **media**(5).

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.

−**r** Recursively rearchive 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.

SEE ALSO

media(5)

NAME

recycler − SAM-FS media recycle command

SYNOPSIS

/opt/LSCsamfs/sbin/recycler [-c] [-d] [-v] [-x] [-X] *media-changer-family-set*|*archive-set*]

AVAILABILITY

LSCsamfs

DESCRIPTION

recycler is responsible for reclaiming expired archive copies from **SAM-FS** removable media. **recycler** is typically started from root's crontab at an off-peak time, but may also be invoked by root at any time from the command line. Commands controlling **recycler** are contained in the recycler commands file */etc/fs/samfs/recycler.cmd* and in some portions of the archiver commands file */etc/fs/samfs/archiver.cmd*. Default values are assumed by **recycler** if *recycler.cmd* file is not present. If *archiver.cmd* is not present, then recycling by archive set will not occur.

If *media-changer-family-set* or *archive-set* is specified, **recycler** will reclaim space from the specified media changer or archive set only; otherwise VSNs from each media changer and archive set may be reclaimed. Nothing is done in either case with a media changer or archive set whose current usage is less than its high-water mark.

media-changer-family-set is the family set name of the single media changer you wish to recycle. The family set name is given as the fourth field in the **mcf** file line defining the media changer.

archive-set is the name of the archive set, including the copy number, as given in the **archiver.cmd** file, of the single archive set you wish to recycle. For example, arset.1.

OPTIONS

The following options may either add or remove extra output in the recycler log in addition to the normal recycler processing.

−**c** Display the extrapolated capacity of each VSN. This is the medium's capacity assuming the compression observed on the medium so far continues for the rest of the medium. This option produces an additional line for each VSN headed "Alpha:".

−**C** Suppress listing of initial catalog(s).

−**d** Display messages during the VSN-selection phase of processing, indicating why each piece of media was or was not selected for recycling.

−**E** Causes the VSN section of the recycler's log to list only VSNs which are not 100% free.

−**s** Prevents listing of individual VSNs in the initial catalog section.

−**v** Display information about which files (or inode numbers, if no pathname can be calculated for the inode) are resident on media marked for recycling. These are the files which are on media which is being drained.

−**V** Suppresses listing of the VSN section.

−**x** Display messages for archive copies which are older than the time the medium on which the copies reside was labeled (expired archive copies). Such copies will error when staged; the data for those copies is irrecoverable. These archive copies must be unarchived. If any such copies are discovered, the recycler will refuse to continue. This is the default behavior. See the **-X** option below.

−**X** Inhibit displaying the messages for expired archive copies. This option is required if it is desired to have the recycler continue in the presence of expired archive copies. See the **-x** option above.

OPERATION

recycler operation is in two phases: VSN selection and VSN recycling. Selection is based on the amount of space used by expired archive copies as a percentage of total space on the VSN. For each

media changer and archive set being recycled, enough VSNs in that media changer or archive set with the highest percentages are selected to bring the media utilization in the media changer or archive set below the configured high-water-mark, provided that each VSN would contribute at least *VSN-minimum-percent-gain* (see **recycler.cmd(4)**) percent of its total space if it were recycled. If no such VSNs exist, the media changer or archive set cannot be recycled. Ties in expired space are resolved by selecting the VSN with the least amount of un-expired space.

There are a few disqualifying conditions for selecting VSNs. If a VSN contains data associated with a removable-media file, that VSN cannot be recycled. If a VSN is listed in the *recycler.cmd* file's **no_recycle** section, that VSN cannot be recycled.

Once VSNs have been selected, they are recycled. They are marked "recycling" so that no new archive copies are written onto them. All files with active archive copies on the VSNs are marked "to be re-archived". The archiver will move these copies to other VSNs. In subsequent runs, **recycler** will check these VSNs and post-process them when all valid archive copies have been relocated.

Finally, **recycler** checks to see if there are VSNs which were previously selected for recycling, but which have not yet been post-processed. If such VSNs exist, and they are now devoid of active archive copies, they are post-processed by invoking the **/etc/fs/samfs/recycler.sh** script with arguments including the generic media type ("tp" or "od"), and the VSN. The slot in the library and the library's equipment number where the VSN is located are also passed. The script can relabel the medium using either the original VSN or a new VSN, the medium can be exported from the library, or any other user-defined action can be taken.

In any case, **recycler.sh** should clear the "recycling" flag to indicate that recycling has completed on the VSN. **odlabel** and **tplabel** clear this flag once the medium has been relabeled.

RECYCLER OUTPUT

The recycler's output is in several sections. The first section describes the catalogs found for each media changer and archive set. First, a header listing the family set name or archive set name as well as the vendor, product and catalog pathname is displayed. Then, the capacity and remaining space for each VSN is shown in bytes with suffices k, M, G, and T denoting kilobytes, Megabytes, Gigabytes and Terabytes. Here, kilobytes means 1024 bytes, Megabytes 1024*1024 bytes, etc. 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 next section is a series of tables, one for each media changer and archive set which have associated VSNs. The name of the media changer or archive set is shown just to the right of "----Percent----". A VSN will be associated with only one media changer or archive set; attempts to assign a VSN to multiple archive sets will be flagged with "in multiple sets". The following fields are displayed:

Status — A phrase giving the VSN's recycle status. Phrases are shown in the table below.

Archives Count — The number of archive copies which are contained on this VSN.

Archives Bytes — The number of bytes of archive copies contained on this VSN.

Percent Use — The percentage of space in use on this VSN by current archive copies. This is the amount of data which would need to be moved if the VSN were selected for recycling.

Percent Obsolete — The percentage of space used on this VSN for which no archive copies were found. This is the space that can be reclaimed by recycling this medium.

Percent Free — The percentage of space remaining on this VSN.

The phrases in the Status column indicate that the VSN:

empty VSN	is empty of both expired and current archive images
full VSN	has no free space, but does have current archive images.
in multiple sets	matches multiple archive sets in the *archiver.cmd* file.
new candidate	was chosen for recycling this recycler run.
no-data VSN	contains only expired archive images and free space.
no_recycle VSN	was mentioned in the no_recycle section of *recycler.cmd*.
archive -n files	contains archive images for files now marked archive -n.
old candidate	was already marked for recycling before this recycler run.
request files	contains archive images for removeable media files.
partially full	contains some current archive images, as well as free space.
shelved VSH	is not currently located in any media changer.

The "Percent Use" value is the percentage of the medium used by current archive images. It is estimated by summing up the sizes of the archive copies on the medium. Because of compression, this value may overstate the amount of space actually used by these images.

The "Percent Free" value comes directly from the media changer catalog. It gives the percent of the medium's total capacity which is available to hold new archive images.

The "Percent Obsolete" value is calculated as 100% - "Use" - "Free". Because "Use" may overstate the actual space used (because of compression), the sum of "In use" + "Free" may exceed 100%, and thus "Obsolete" may be a negative value. Although aesthetically unpleasing, this does not cause any problems in the operation of the recycler.

For media types which support 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 medium (that is, capacity - space) to the logical number of tape blocks written to the medium. The latter value is kept in the catalog. This ratio is then used to adjust the "In use" value before it is displayed.

The first VSN displayed for each media changer 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
   slot                  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: /etc/fs/samfs/ad40
   Vendor: ADIC                 Product: Scalar DLT 448
   slot                 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/fs/samfs/archiver.cmd
   Vendor: SAM-FS               Product: Archive set
   slot                 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
```

recycler.cmd file:

```
logfile = /var/tmp/recycler.log
ad40 75 50
no_recycle mo ^OPT003
```

RECYCLING HISTORIAN MEDIA

The **recycler** will recycle media listed in the historian's catalog. The media listed in the historian catalog have been exported from a library or have been or are currently in a manually-mounted device.

recycler.sh is passed the media changer name "hy" for historian-catalog resident media so that it can cope with the possibility of the medium being recycled's residing in an off-site storage facility. Typically, recycler.sh will send e-mail to the administrator when this occurs, reminding the administrator to bring the off-site medium back on-site so that it can be reused. Media does not need to be on-site to be drained of archive copies, unless such a medium contains the only available archive copy of an off-line file.

RECYCLING BY ARCHIVE SET

The **recycler** can also recycle by archive set. It treats each archive set as a small media changer, which holds just the VSNs which the *archiver.cmd* file assigned to the archive set. The VSNs which are identified as belonging to a recycling archive set are removed from the recycler's version of the catalog for the media changer which physically contains the VSN. Thus, only the VSNs which are not part of an archive set remain in the media changer's catalog.

To enable recycling for a given archive set, it must have one of the recycling options specified in the *archiver.cmd* file. (see **archiver.cmd(4)**)

MESSAGES

A message like:

```
Jan 22 10:17:17 jupiter recycler[3400]: Cannot ioctl(F_IDSCF)
      Cannot find pathname for filesystem /samfs1 inum/gen 406/25
```

means that the recycler could not set the "rearchive" flag for a file. Normally, when this happens, the recycler will emit a message with the pathname, like:

```
Jan 22 10:17:17 jupiter 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 example message above, inode number 406) and generation number (here, 25) into a pathname in the /samfs1 filesystem.

The most likely of a number of reasons this occurs 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.

Previous versions of SAM-FS sometimes allowed deleted inodes to remain in the .inodes file, leading to so-called "orphan" inodes. These inodes also could not be resolved into a pathname, and would lead to a similar message.

SEE ALSO

archiver.cmd(4), **archiver**(1), **chmed**(1M), **mcf**(4), **odlabel**(1M), **recycler.cmd**(4), **recycler.sh**(4), **tplabel**(1M)

NAME

releaser − SAM-FS disk space release daemon

SYNOPSIS

/etc/fs/samfs/releaser x *mount_point low_water_mark weight-size* [*weight-age*]

AVAILABILITY

LSCsamfs

DESCRIPTION

releaser attempts to release the disk space of archived files on the file system mounted on *mount_point* until the *low_water_mark* is reached. **releaser** is started automatically by the SAM-FS file system when the disk cache utilization reaches the high-water mark. *weight-size* and *weight-age* are weighting factors used to prioritize release candidates (see **mount_samfs**(1M)).

Beginning with SAM-FS release 3.3.1, the **releaser** reads a releaser command file (*/etc/fs/samfs/releaser.cmd*) for parameters which override command-line arguments. (See **releaser.cmd**(4)). However, **x** must appear as the first argument for historical reasons.

ALGORITHM

releaser reads the SAM-FS .inodes file, building an ordered list of the files that can be released. The position of each file in the list is dependent on a priority calculated for each inode by the releaser (see the PRIORITY WEIGHTS section below.) 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 which could be released, the **releaser** stops. If the filesystem is still above high-water mark, the filesystem will restart **releaser**.

PRIORITY WEIGHTS

Each inode is assigned a priority based on its size and age. The size of the file (expressed in units of 4k-Byte 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 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 other method allows specification of weights for each of the access (*weight_age_access*), modification (*weight_age_modify*), and residence-change (*weight_age_residence*) ages. The sum of the product of the weight and corresponding age is the contribution of the age to the file's priority.

In all cases, the ages are expressed in minutes.

LOG

The *releaser.cmd* file can specify a logfile for each SAM-FS filesystem. If no command file exists, or if no "logfile" directives exist in the file, no logging will occur. The **releaser** will create this log file (if it does not exist) and append to it the following 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-init?    no
release files?          no
```

```
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 /sam1/250m
5131.5 (R: Tue Sep 22 17:39:47 CDT 1998) 9951 min, 156 blks /sam1/filecq
5095.5 (R: Tue Sep 22 17:39:49 CDT 1998) 9951 min, 120 blks /sam1/filecu
5062 (R: Tue Sep 22 18:38:50 CDT 1998) 9892 min, 116 blks /sam1/filebz
5039.5 (R: Tue Sep 22 17:40:01 CDT 1998) 9951 min, 64 blks /sam1/filedi
5036.5 (R: Tue Sep 22 17:37:34 CDT 1998) 9953 min, 60 blks /sam1/fileio
5035.5 (R: Tue Sep 22 17:40:13 CDT 1998) 9951 min, 60 blks /sam1/filedw
5032.5 (R: Tue Sep 22 17:38:08 CDT 1998) 9953 min, 56 blks /sam1/filejq
5031.5 (R: Tue Sep 22 17:39:56 CDT 1998) 9951 min, 56 blks /sam1/fileda
5024.5 (R: Tue Sep 22 17:38:00 CDT 1998) 9953 min, 48 blks /sam1/filejh
5024 (R: Tue Sep 22 17:38:22 CDT 1998) 9952 min, 48 blks /sam1/fileka
5023.5 (R: Tue Sep 22 17:40:07 CDT 1998) 9951 min, 48 blks /sam1/filedn
5019 (R: Tue Sep 22 17:40:44 CDT 1998) 9950 min, 44 blks /sam1/filefk
5015 (R: Tue Sep 22 17:40:28 CDT 1998) 9950 min, 40 blks /sam1/fileep
5011.5 (R: Tue Sep 22 17:40:14 CDT 1998) 9951 min, 36 blks /sam1/filedx
5011.5 (R: Tue Sep 22 17:39:58 CDT 1998) 9951 min, 36 blks /sam1/filede
5011 (R: Tue Sep 22 17:41:07 CDT 1998) 9950 min, 36 blks /sam1/filegk
5007.5 (R: Tue Sep 22 17:39:51 CDT 1998) 9951 min, 32 blks /sam1/filecw
5007 (R: Tue Sep 22 17:41:10 CDT 1998) 9950 min, 32 blks /sam1/filegr
5007 (R: Tue Sep 22 17:40:42 CDT 1998) 9950 min, 32 blks /sam1/filefg
5007 (R: Tue Sep 22 17:40:30 CDT 1998) 9950 min, 32 blks /sam1/filees
5004.5 (R: Tue Sep 22 17:38:14 CDT 1998) 9953 min, 28 blks /sam1/filejv
5004 (R: Tue Sep 22 17:38:57 CDT 1998) 9952 min, 28 blks /sam1/filelm
5002 (R: Tue Sep 22 18:38:54 CDT 1998) 9892 min, 56 blks /sam1/filecd
4996.5 (R: Tue Sep 22 17:38:06 CDT 1998) 9953 min, 20 blks /sam1/filejp
4995.5 (R: Tue Sep 22 17:39:57 CDT 1998) 9951 min, 20 blks /sam1/filedc
4992.5 (R: Tue Sep 22 17:37:24 CDT 1998) 9953 min, 16 blks /sam1/fileig
4992 (R: Tue Sep 22 17:39:06 CDT 1998) 9952 min, 16 blks /sam1/filelv
4986 (R: Tue Sep 22 18:38:50 CDT 1998) 9892 min, 40 blks /sam1/fileca
4982 (R: Tue Sep 22 17:36:54 CDT 1998) 9954 min, 5 blks /sam1/filehk
4981 (R: Tue Sep 22 17:41:09 CDT 1998) 9950 min, 6 blks /sam1/filegn
4980.5 (R: Tue Sep 22 17:40:15 CDT 1998) 9951 min, 5 blks /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
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 **releaser** was invoked with: The name of the .inodes file; the low-water mark; the size and age weight parameters; the equipment ordinal of the filesystem; the family set name of the filesystem; if started by sam-init or by the command line; if files should be released; and if each inode should be logged as encountered.

The next block of lines (---before scan---) shows the number of blocks currently free in the cache, and the number which would be free if the filesystem were exactly at the low-water mark. The goal of **releaser** is to increase **blocks_now_free** so that it is equal to or larger than **lwm_blocks**.

Next, under the heading ---scanning--- is a list of the files released by **releaser**. The first field on each line is the release priority. The next field is (*tag*: *date*), where *tag* is A, M or R, depending on if the *date* which follows represents the Access, Modify or Residency time. *date* is the most recent of the three dates listed. Next is "*mmm* min, *bbb* blks", where *mmm* is the age of the file in minutes, and *bbb* is the size in blocks. These two figures are multiplied by their respective weights and the sum taken to yield the release priority. The last field on the line is the file's pathname.

If any of *weight_age_access*, *weight_age_modify* or *weight_age_residence* is specified these lines only show the priority, size and pathname.

Under ---after scan--- the statistics accumulated by **releaser** during the previous scan pass are shown:

Statistic	Meaning
archnodrop	The number of inodes marked "archnodrop." These are files which are not to be released because the archiver is trying to keep them in cache.
already_offline	The number of inodes which were offline.
bad_inode_number	This field is not used, and will always be zero.
damaged	The number of inodes which were marked damaged.
extension_inode	The number of extension inodes found (used by volume overflow.)
negative_age	The number of inodes which had an age in the future. This is usually caused by PC's with incorrect clock settings acting as NFS clients.
nodrop	The number of inodes marked "release -n".
not_regular	The number of inodes which were not regular files.
number_in_list	The number of inodes which were on the **releaser**'s candidate list when **releaser** was finished scanning.
released_files	The number of files which **releaser** released.
too_new_residence_time	The number of inodes whose residence-change time was within 10 minutes of the current time.
too_small	The number of files which were too small to be released.
total_candidates	The number of inodes found which 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. You should run *samfsck* once to rescue any orphan inodes. If you've already run *samfsck* and this field remains non-zero, no further action is required.
zero_arch_status	The number of inodes which had no archive copies.
zero_inode_number	The number of inodes which had zero as their inode number.
zero_mode	The number of inodes which 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 10 minutes before the file is considered for release. This is to prevent a file which was recently staged in from being released.

If **releaser** selects a file as a release candidate, and immediately thereafter the file is accessed, the file may still be released by the filesystem, even though the file has been recently accessed. This happens because the filesystem 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

mount_samfs(1M) **release**(1), **releaser.cmd**(4),

NAME

robots, generic, stk, ibm3494, rs_server, rs_client − SAM-FS media changer daemons

SYNOPSIS

/etc/fs/samfs/robots *mshmid pshmid*
/etc/fs/samfs/generic *mshmid pshmid equip*
/etc/fs/samfs/stk *mshmid pshmid equip*
/etc/fs/samfs/ibm3494 *mshmid pshmid equip*
/etc/fs/samfs/rs_server *mshmid pshmid equip*
/etc/fs/samfs/rs_client *mshmid pshmid equip*

AVAILABILITY

LSCsamfs LSCstk LSCibm LSCremote

DESCRIPTION

robots starts and monitors the execution of the media changer (robot) control daemons for SAM-FS. **robots** is started automatically by **sam-init** if there are any media changers defined in the configuration file. (See **mcf**(4)). **robots** will start and monitor the correct daemon for all the media changers defined.

Each of the media changer daemons are responsible for monitoring the preview table for vsn's that are controlled by that daemon. If a request is found for one of its vsn's, the daemon will find an available drive under its control and move the media into that drive. Once the device is ready, the daemon will notify SAM-FS and the device will be assigned to the waiting process.

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**. *equip* is the equipment number of the device.

generic is the daemon that controls media changers that conform to the SCSI II standard for media changers, and it is the daemon that controls the GRAU ABBA media changers through the GRAU ACI interface. See **grauaci**(7) for more information.

stk is the daemon that controls the StorageTek (STK) media changers through the ACSAPI interface. See **stk**(7) for more information. This is an optional package (LSCstk) and not part of the standard LSCsamfs package.

ibm3494 is the daemon that controls the IBM 3494 tape libraries through the *lmcpd* interface. See **ibm3494**(7) for more information. This is an optional package (LSCibm) and not part of the standard LSCsamfs package.

rs_server and **rs_client** are the daemons that control SAM-Remote. See **sam_remote**(7) for more information. This is an optional package (LSCremote) and not part of the standard LSCsamfs package.

FILES

mcf The configuration file for SAM-FS

SEE ALSO

sam-init(1M), **mcf**(4), **acl2640**(7), **acl452**(7), **atlp3000**(7), **grauaci**(7), **stk**(7), **ibm3494**(7), **sam_remote**(7)

NAME

robottool − Graphical user interface for displaying information about and managing robot devices associated with SAM-FS

SYNOPSIS

/opt/LSCsamfs/sbin/robottool

AVAILABILITY

LSCsamfs

DESCRIPTION

robottool presents a graphical user interface for viewing information about and managing the robot devices associated with SAM-FS.

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 slots 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 slot is unavailable

l volume is labeled

c cleaning media

p slot 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, slot 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 SAM-FS

SEE ALSO

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

NAME

rpc.sam − SAM-FS RPC API server process

SYNOPSIS

/etc/fs/samfs/rpc.sam

AVAILABILITY

LSCsamfs

DESCRIPTION

rpc.sam is the RPC API (Application Programmer Interface) server process. It is initiated by **sam-init**.

rpc.sam uses the RPC program number that is paired with the RPC program name *samfs*. **rpc.sam** must run on the same machine as SAM-FS. 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-init − Initialize the SAM-FS system daemons

SYNOPSIS

/etc/fs/samfs/sam-init

AVAILABILITY

LSCsamfs

DESCRIPTION

sam-init initializes the SAM-FS 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-init**.

FILES

/etc/fs/samfs/	Location of SAM-FS daemons and configuration files
mcf	The configuration file for SAM-FS

SEE ALSO

mcf(4), **mount**(1M), **mount_samfs**(1M), **archiver**(1M), **generic**(1M), **scanner**(1M), **robots**(1M)

NOTES

sam-init 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-init**, use **ps**(1) to determine the process ID (*sam-init-pid*) of **sam-init** then enter the following:

kill -INT *sam-init-pid*

To shutdown **sam-init** 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-init | cut -c1-6‘’

NAME

sam_trace − Dump the SAM-FS trace buffer

SYNOPSIS

/opt/LSCsamfs/sbin/sam_trace [−**d** *corefile*] [−**n** *namelist*] [−**v**]

AVAILABILITY

LSCsamfs

DESCRIPTION

sam_trace dumps the contents of the trace buffer for the mounted SAM-FS 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.

−**v** Verbose option.

NOTE

sam_trace is a utility that is used to provide LSC, Inc. 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

samcmd − execute SAM-FS operator utility command

SYNOPSIS

samcmd *command*

AVAILABILITY

LSCsamfs

DESCRIPTION

samcmd executes a single SAM-FS 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. Succeeding arguments are the arguments for the **samu** command.

EXAMPLES

The following example mounts a medium from slot 2 in robot 30:

```
samcmd mtslot 30 2
```

The following example produces a detailed archiver display for filesystem samfs3 on standard output:

```
samcmd a samfs3
```

SEE ALSO

samu(1M)

NAME

samd − Utility to start or stop the SAM-FS daemons

SYNOPSIS

/opt/LSCsamfs/sbin/samd [start | stop]

AVAILABILITY

LSCsamfs

DESCRIPTION

samd is a utility to start up the sam-init master daemon or shut down the daemon.

OPTIONS

start Start up sam-init if the /etc/fs/samfs/mcf file exists and sam-init is not already running.

stop Kills sam-init.

SEE ALSO

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

NAME

samdev − Adds **/dev/samst** entries for media changers, optical disks and tape drives attached to the system

SYNOPSIS

/opt/LSCsamfs/sbin/samdev [-d]

AVAILABILITY

LSCsamfs

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 entries 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.

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 samfs file system

SYNOPSIS

samfsck [−**s** *scratch_dir*] [−**F**] [−**V**] *fs_name*

AVAILABILITY

LSCsamfs

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.

−**V** Turns on a verbose diplay of DEBUG information. This information is useful to LSC, Inc. analysts.

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/fs/samfs/mcf The configuration file for **samfs**

SEE ALSO

mcf(4)

NAME

samfsdump, samfsrestore − Dump or restore file control structure data

SYNOPSIS

samfsdump [**-dHqTvW**] **-f** *dump_file* **[** *file...* **]**

samfsrestore [**-dilRstTv2**] **-f** *dump_file* **[***file...* **]**

DESCRIPTION

samfsdump creates a dump file of the control structures 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, **samfsdump** creates a dump file of the control structures of the current relative directory (referenced as ".") and (recursively) its subdirectories (referenced as "./<subdirectory_name>").

samfsrestore uses the contents of the dump file to restore the control structures 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 **samfsdump** and **samfsrestore**, the dump file must be specified in **-f** *dump_file*, where *dump_file* specifies the name of the control structure data dump file to write or read, respectively. If a - (dash) is specified for the *dump_file*, **samfsdump** will write the dump file of control structure data to **stdout** or **samfsrestore** 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 **samfsdump** or before being read by **samfsrestore**.

samfsdump 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 which has no archive copy is dumped, a warning message will be issued (unless explicitly suppressed with the **-q** option) that this file will be marked damaged when restored. When that file is restored from the dump file, it will be marked damaged by **samfsrestore**.

samfsdump and **samfsrestore** require the super-user for execution. LSC, Inc., recommends that a site create **samfsdump** dumps on a periodic basis as part of a disaster recovery plan.

OPTIONS

-d Enable debugging messages. Useful only to LSC, Inc., 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, **samfsrestore** 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, symbolic links, and removable-media files 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 samfsdump), or were damaged during a restore because of having no archive image (for a samfsrestore). "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 samfsdump or samfsrestore. 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**.

-W This option is retained for compatibility. It used to enable the issuing of warning messages during the dump for those files which would be damaged should the dump be restored and is now allowed only for compatibility. These warning messages are now issued automatically by default. See the **-q** option above to suppress the issuing of these important warning messages.

-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.

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 **samfsrestore -t** to see how the names are stored.

NOTES

samfsdump output files compress to less than 25% of their original size.

ERRORS

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

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

"*file*: Warning! File will be damaged." during a **samfsdump** 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." during a **samfsdump** 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 **samfsrestore** means that the file was dumped with the "damaged" flag set.

"*file*: File is now damaged" during a **samfsrestore** means that the file was dumped when it had no archive images. Since **samfsdump**/**samfsrestore** do not dump file data, but rely on the file's data having been archived, the file no longer has any data associated with it, and is marked "damaged".

".: 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 **samfsdump** 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 **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" 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. **samfsrestore** 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. **samfsrestore** 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 /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

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

NAME

samgrowfs − Add to an existing SAM-FS file system

SYNOPSIS

samgrowfs [−**f** *mcf*] [−**V**] *fs_name*

AVAILABILITY

LSCsamfs

DESCRIPTION

samgrowfs grows a SAM-FS file system from the disk partitions that belong to the existing family set *fs_name*. *fs_name* is the family set name from the **mcf** file. Up to 200 disk partitions can be specified in the **mcf** file for a SAM-FS file system. The new partitions must be placed after the existing partitions for the family set *fs_name*.

In order to use **samgrowfs** you must have the file system unmounted. You must restart sam-init in order for it to re-read the mcf file before you remount the file system.

CAUTION

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.

OPTIONS

−**f** *mcf* Path name to the mcf file. The default is **/etc/fs/samfs/mcf**.

−**V** List configuration information, but do not execute the command.

EXAMPLE

The following example adds 2 partitions to an existing 1 partition filesystem. The **mcf** for the existing 1 partition filesystem with family set name **samfs1:**

```
samfs1              10   ms   samfs1
/dev/dsk/c0t3d0s7   11   md   samfs1   -   /dev/rdsk/c0t3d0s7
```

1) unmount the SAM-FS file systems

2) **kill -INT** *sam-init's pid*

3) Change the **mcf** and add the 2 new partitions for filesystem with family set name **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) Enter the commands:

```
samgrowfs samfs1
mount samfs1
```

FILES

/etc/fs/samfs/mcf The configuration file for **samfs**

SEE ALSO

sammkfs(1M), **mcf**(4)

NAME

sammkfs − Construct a new SAM-FS file system

SYNOPSIS

sammkfs [−**a** *allocation_unit[***k***]*] [−**i** *ninodes*] [−**r** *inode_file*] [−**V**] *fs_name*

AVAILABILITY

LSCsamfs

DESCRIPTION

sammkfs makes 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. Up to 200 disk partitions can be specified in the *mcf* file for a samfs file system.

CAUTION

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.

OPTIONS

−**a** *allocation_unit*

Specifies the number of bytes in units of 1024 byte blocks to be allocated to a disk allocation unit (DAU). A DAU is represented by 1 bit in the bit maps. The default *allocation_unit* is 16k. If the filesystem contains striped groups, a DAU is *allocation_unit* ∗ the number of members in the striped group. A DAU is the minimum disk space allocated. This means the minimum disk space allocated in a striped group is allocation_unit ∗ number of members in the group. DAU is >= 16k and < 65536k on the *ma* equipment type filesystem. DAU is 16k, 32k, or 64k on the *ms* equipment type filesystem. (See man **mcf**(4) for a description of the *ms* and *ma* equipment type filesystems).

−**i** *ninodes* Specifies the number of inodes to be preallocated for this filesystem.

−**r** *inode_file*

Restore using the **.inodes** file. If specified, the directories and files in *inode_file* are restored. When *fs_name* is mounted, all the directories and files are offline. If an archived directory or file has been modified and not yet rearchived, it is marked damaged, offline, and all archive copies are marked stale. After the filesystem is mounted, the **undamage** command can be used to unstale and undamage the directory or file.

−**V** List configuration information, but do not execute the command.

FILES

/etc/fs/samfs/mcf The configuration file for SAM-FS

NOTES

Data alignment refers to matching the allocation unit of the RAID controller with the allocation_unit of the filesystem. A mismatched alignment causes a read-modify-write operation for I/O that is less than the block size. The optimal alignment formula is:

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 64k, then the optimal allocation_unit is 64 ∗ 7 = 448. The format command can be used to determine the number of data disks: divide the capacity by the size of the disks.

SEE ALSO

samgrowfs(1M), **undamage**(1M), **mcf**(4),

NAME

samncheck − generate pathnames vs. i-numbers for SAM-FS file systems

SYNOPSIS

samncheck *mount_point i-number* [*i-number* ...]

AVAILABILITY

LSCsamfs

DESCRIPTION

samncheck generates a pathname in the SAM-FS 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

samset − Change SAM-FS environment

SYNOPSIS

samset *[keyword [parameter...]]*

AVAILABILITY

LSCsamfs

DESCRIPTION

samset is used to change or display variables that control SAM-FS 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 SAM-FS startup. These values come from the **defaults.conf** file. **samset** allows you to change *keywords* while SAM-FS is running. Any changes made remain effective only during the current instance of SAM-FS; values revert to the defaults in **defaults.conf** at the next startup.

The following *keywords* are supported:

attended yes

attended no

attended tells SAM-FS 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 SAM-FS 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 SAM-FS 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.

SAM-FS 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).

stage_checking *x*

Enables levels of stage recovery. Each level includes the previous levels. Values for *x* are:

0 No stage recovery enabled.

1 Retry position if tar header is not found.

2 Verify tape is at expected position before each read.

3 Verify tape is at expected position after each read.

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 SAM-FS 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 LSC, Inc., 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 LSC, Inc., 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 LSC, Inc., 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.

mounts Log media mount requests.

opens Log open and close of removable media devices.

robot_delay
Causes all media changers to delay 60 seconds during startup.

staging This should be used only by LSC, Inc., analysts to trace file staging.

stageall This should be used only by LSC, Inc., analysts to trace stageall processing.

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. LSC, Inc., 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;
```

LSC, Inc., does not recommend setting this option indiscriminately, as large output files are quickly produced.

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.

stage_ck Output stage recovery detail messages.

stage Stage operations.

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), **media**(5)

NAME

samtool − Graphical user interface for starting SAM-FS administrative tools

SYNOPSIS

/opt/LSCsamfs/sbin/samtool

AVAILABILITY

LSCsamfs

DESCRIPTION

samtool contains programs that help you manage and view information about the robots, devices, and pending mount requests associated with SAM-FS.

samtool displays three icons, one for each of the available SAM-FS 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 SAM-FS

SEE ALSO

devicetool(1M), **previewtool**(1M), **robottool**(1M), **mcf**(4)

NAME

samu − SAM-FS operator utility

SYNOPSIS

samu [−**d** *c*] [−**r** *i*] [−**c** *string*] [−**f** *cmd-file*]

AVAILABILITY

LSCsamfs

DESCRIPTION

samu is a curses-based operator interface for SAM-FS. It has a number of displays that show the status of devices managed by SAM-FS and allows the operator to control 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,A,J,M
	Next inode	I
	Next sector	S
	Next equipment	T,U
control-b	Previous file system	:a,a
	Page backward	c,h,o,p,s,t,u,v,A,J,M
	Previous inode	I
	Previous sector	S
	Previous equipment	T,U
control-d	Half-page forward	c,p,s,u,A,J,M
	Next robot catalog	v
	Page forward	h,S
	Page arcopies forward	a
control-u	Half-page backward	c,p,s,u,A,J,M
	Previous robot catalog	v
	Page backward	h,S
	Page arcopies backward	a
control-k	Advance display format	A,I,S
	Select (manual,robotic,both,priority)	p
	Advance sort key	v

Toggle path display		n,u

The sort selections for the v display are: slot, count, usage, VSN, access time, barcode, label time.

DISPLAYS

The following displays are available:

a	Display archiver status	A	Archiver shared memory
c	Display configuration	C	Memory
h	Display help information	F	Optical disk label
m	Display mass-storage status	I	Inode
n	Display staging activity	J	Preview shared memory
o	Display optical disk status	L	Shared memory tables
p	Display mount request preview	M	Shared memory
r	Display removable media	N	Mount table
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	Kernel tables

COMMANDS

The following commands may be entered after a colon (:).

Archiver commands:

arrestart	Restart archiver
arrun	Start archiving
artrace [option ...]	Set trace options

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

Device commands:

down	eq	Mark equipment down
idle	eq	Idle equipment
maxcontig	eq contig	Set maximum contiguous blocks
off	eq	Off equipment
on	eq	On equipment
partial	eq size	Set default size in kB left online after release
readonly	eq	Mark equipment read-only
ro	eq	Mark equipment read-only
thresh	eq high low	Set disk thresholds
unavail	eq	Mark equipment unavailable
unload	eq	Unload mounted media/magazine

Robot commands:

audit	eq	Audit jukebox
import	eq	Import cartridge from mailbox
export	eq slot	Export cartridge to mailbox
mtslot	eq slot	Mount cartridge in drive
mtvsn	eq vsn	Mount cartridge in drive

Miscellaneous commands:

clear	vsn [slot]	Clear mount request
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
!shell-command		Run shell command

SEE ALSO

curses(3), **media**(5)

NAME

scanner − SAM-FS daemon for manually-mounted devices

SYNOPSIS

/etc/fs/samfs/scanner *mshmid pshmid*

AVAILABILITY

LSCsamfs

DESCRIPTION

scanner monitors the manually-mounted devices. It will periodically check each device for newly inserted media. If **scanner** 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 SAM-FS file system is notified and the device is assigned to the request.

scanner 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

set_state − Set device state

SYNOPSIS

/opt/LSCsamfs/sbin/set_state [**-w**] *state eq*

AVAILABILITY

LSCsamfs

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 SAM-FS. 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 SAM-FS. 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 SAM-FS file system and most SAM-FS 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 SAM-FS. 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 SAM-FS. 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 SAM-FS

SEE ALSO

load(1M), **unload**(1M), **mcf**(4), **robots**(1M)

NAME

setsyscall − Change system call number in SAM-FS library

SYNOPSIS

/opt/LSCsamfs/sbin/setsyscall *syscall* [*library_file*]

AVAILABILITY

LSCsamfs

DESCRIPTION

setsyscall is used to set the system call number *syscall* in the SAM-FS library **/opt/LSCsamfs/lib/libsam.so**.

syscall must be the same value as entered in **/etc/name_to_sysnum** on the "samsys" line.

The optional argument may be used if the library file name is not **/opt/LSCsamfs/lib/libsam.so**.

FILES

/etc/name_to_sysnum
/opt/LSCsamfs/lib/libsam.so

NAME

stageall − SAM-FS associative staging daemon

SYNOPSIS

/etc/fs/samfs/stageall

AVAILABILITY

LSCsamfs

DESCRIPTION

stageall 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

tplabel − Label tape

SYNOPSIS

tplabel [−**w**] [−**V**] [−**b** *blksize*] [−**erase**] −**vsn** *vvvvvv* −[**new** | **old** *vv...*] [−**slot** *n*] *device*

DESCRIPTION

tplabel labels the tape on the specified *device* (a device name or equipment ordinal). 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 *device* is a robot, the slot in which the desired media is stored must be specified by −**slot** *n*. The slot number is treated as hexadecimal if the number starts with 0x, as octal if starting with a 0 (zero), otherwise as decimal.

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

−**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**] [−**o**] [−**c** *n*] [−**m** *media*] [−**v** *vv...*] *filename...*

unarchive [−**f**] [−**o**] [−**c** *n*] [−**m** *media*] [−**v** *vv...*] −**r** *dirname...* [*filename...*]

AVAILABILITY

LSCsamfs

DESCRIPTION

unarchive deletes archive entries for one or more files or directories. 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 deleted. The default is all copies.

If −**m** *media* is specified, archive copies on the specified media are deleted. See **media**(5).

If −**v** *vv...* is specified, −**m** must also be specified. In this case, archive copies on the specified VSN *vv...* are deleted.

OPTIONS

−**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

media(5)

NAME

undamage − Undamage and unstale archive entries

SYNOPSIS

/opt/LSCsamfs/sbin/undamage [−**f**] [−**c** *n*] [−**m** *media*] [−**v** *vv...*] *filename...*

/opt/LSCsamfs/sbin/undamage [−**f**] [−**c** *n*] [−**m** *media*] [−**v** *vv...*] −r *dirname...* [*filename...*]

AVAILABILITY

LSCsamfs

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 **media**(5).

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.

−**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

media(5)

NAME

unload − Unload media from a device

SYNOPSIS

/opt/LSCsamfs/sbin/unload [−**w**] *eq*

AVAILABILITY

LSCsamfs

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 SAM-FS 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 SAM-FS

SEE ALSO

auditslot(1M), **historian**(7), **load**(1M), **set_state**(1M), **mcf**(4), **robots**(1M)

NAME

unrearch − Mark archive entries to be not rearchived.

SYNOPSIS

unrearch [−**f**] [−**o**] [−**c** *n*] [−**m** *media*] [−**v** *vv...*] *filename...*

unrearch [−**f**] [−**o**] [−**c** *n*] [−**m** *media*] [−**v** *vv...*] −**r** *dirname...* [*filename...*]

AVAILABILITY

LSCsamfs

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 **media**(5).

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.

−**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

media(5)

NAME

sam_advise − Set attributes on a file

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/include/lib.h**"

int sam_advise(const int *fildes***, const char** ∗*ops***);**

DESCRIPTION

sam_advise( ) provides advise about expected behavior of the application when accessing data in the file associated with the open file descriptor, *fildes*. **sam_advise( )** provides advise for a file using a SAM-FS ioctl call. The last caller of sam_advise() sets the advice for all applications using the file.

ops is the character string of options, for example: "dw". Individual options are described below.

OPTIONS

d Return the file attributes on the file to the default, i.e. the I/O is reset to virtual memory I/O. 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.

r Advises the system to use direct (raw) I/O (see man directio(3C) for Solaris 2.6 and above). The default I/O is virtual memory (cached) I/O.

p Advises the I/O buffers used for directio are locked (see man mlock). This means the system does not lock the I/O buffers used for directio. 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 directio. 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

sam_setfa(3), **directio**(3C), **mount_samfs**(1M), **mcf**(4)

NAME

sam_archive − Set archive attributes on a file or directory

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/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 SAM-FS 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

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.

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/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/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

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/include/catalog.h**"

int sam_closecat(int *cat_handle***);**

DESCRIPTION

sam_closecat( ) closes the robot catalog specified by *cat_handle*. *cat_handle* is a catalog "handle" obtained from a previous call 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 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/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/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 SAM-FS graphical user interfaces and in the SAM-FS 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 SAM-FS shared memory segment.
EINVAL	The size of the SAM-FS shared memory segment is incorrect. You may need to recompile your program with the current version of SAM-FS.
ENOENT	Access to the SAM-FS shared memory segment has failed; possibly SAM-FS isn't running.
EMFILE	The SAM-FS 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 SAM-FS 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/LSCsamfs/lib -lsam [** *library* **...]**

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:

tab (/) ;
l l .
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/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/include/catalog.h**"

int sam_getcatalog(int *cat_handle***, uint** *start_slot***, uint** *end_slot***, struct sam_cat_ent** ∗*buf***, size_t** *entbufsize***);**

DESCRIPTION

sam_getcatalog() obtains a range of catalog entries from the robot catalog 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_slot* and *end_slot*. *start_slot* must be less than or equal to *end_slot*, and must be in the range of valid slot numbers for the robot. *buf* is a pointer to an array of **sam_cat_ent** structures, into which the catalog entry information is placed. This array should be larger 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;          /* Type of slot */
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.
EINVAL	The buffer size provided is invalid, or **start_slot** or **end_slot** is invalid. (Either **start_slot** is less than zero, **end_slot** is greater than the number of slots in the catalog, or **start_slot** is greater than **end_slot**.)
EINTR	A signal was caught during the **sam_getcatalog()** function.

SEE ALSO

sam_opencat(3), **sam_closecat**(3)

NAME

sam_opencat − Open a catalog to read entries

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/include/catalog.h**"

int sam_opencat(const char ∗*path*, **struct sam_cat_tbl** ∗*buf*, **size_t** *bufsize*);

DESCRIPTION

sam_opencat( ) opens the robot catalog pointed to by *path*. The string which *path* points to is limited to 31 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 robot catalog file must be readable by the user. The user may have MAX_CAT catalogs open simultaneously.

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 */
```

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. Otherwise, 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 are true:

EMFILE	The user already has MAX_CAT catalogs open, or the process has too many open files.
EACCES	A component of the path prefix denies search permission.
EFAULT	*path* or *buf* points to an illegal address.
EINTR	A signal was caught during the function.
EINVAL	*bufsize* is set to an invalid value.
ELOOP	Too many symbolic links were encountered in translation
ENOMEM	Not enough memory to memory map the catalog file.

SEE ALSO

sam_closecat(3), **sam_getcatalog**(3)

NAME

sam_readrminfo − Get removable-media file status

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/LSCsamfs/lib -R/opt/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/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/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/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 SAM-FS 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. *n* is an integer between 8 and 4194256. Not valid with sam_stage *n* option. Also 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_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.

SEE ALSO

release(1), **ssum**(1), **sam_ssum**(3)

NAME

sam_request − Create a removable-media file

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/LSCsamfs/lib -R/opt/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/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:

tab (%) ;
l l l .
%%/∗ 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 ∗/
tab (%) ;
l2 l3 l2 .
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.**

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:

tab (%) ;
l2 l2 l2 .
%%/* 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*.

SEE ALSO

request(1), **mcf**(4)

NAME

sam_ssum − Set checksum attributes on a file

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/include/lib.h**"

int sam_ssum(const char ∗*path***, const char** ∗*ops***);**

DESCRIPTION

sam_ssum( ) sets the checksum attributes on a file using a SAM-FS 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 LSC 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 LSC-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/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/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 SAM-FS 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.

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

stageall(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/LSCsamfs/lib -R/opt/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/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 */

/* 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/LSCsamfs/lib -lsam [** *library* **...]**

#include </opt/LSCsamfs/include/stat.h**>**

int sam_vsn_stat(const char ∗*path***, int** *copy***, struct sam_vsn_stat** ∗*buf***, size_t** *bufsize***);**

DESCRIPTION

sam_vsn_stat( ) obtains information about the VSNs for 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.

copy is the archive copy number (1 to 4).

buf is a pointer to a **sam_vsn_stat( )** 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. This should be equal to sizeof(struct sam_vsn_stat).

The contents of the structure pointed to by *buf* include the following MAX_VSNS 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 which 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*.

SEE ALSO

sam_stat(3)

NAME

sam_archive − Set archive attributes on a file or directory

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/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 SAM-FS 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

archive(1), **sam_archive**(3), **sam_initrpc**(3X), **sam_closerpc**(3X)

NAME

sam_closerpc − Perform RPC shutdown for SAM-FS RPC API library

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/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 SAM-FS RPC API library

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/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 SAM-FS server and creates an RPC client handle. In essence, this routine sets up the connection to the SAM-FS host machine, required for other API calls in the **libsamrpc** library.

rpchost is the hostname of the SAM-FS 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 host-name of the SAM-FS 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 SAM-FS RPC API server process, **rpc.sam**, which runs on the SAM-FS 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/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/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 SAM-FS 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

release(1), **sam_release**(3), **sam_initrpc**(3X), **sam_closerpc**(3X)

NAME

sam_setfa − Set release attributes on a file or directory

SYNOPSIS

cc [*flag* **...]** *file* **... -L/opt/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/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 release attributes on a file or directory by sending its request to the SAM-FS 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

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/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/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 SAM-FS 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

stageall(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/LSCsamfs/lib -lsam [** *library* **...]**

#include "**/opt/LSCsamfs/include/stat.h**"
#include "**/opt/LSCsamfs/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 SAM-FS RPC server, **rpc.sam**.

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

ReservedVSNs − SAM-FS archiver VSN reservation data

SYNOPSIS

/etc/fs/samfs/.archive/ReservedVSNs

AVAILABILITY

LSCsamfs

DESCRIPTION

/etc/fs/samfs/.archive/ReservedVSNs is the data file that the archiver uses to keep the list of VSNs reserved to Archive Sets. When the archiver initializes, it reads this file to determine previous VSN reservations. If this file is not found, a warning message is sent to the syslog.

The file consists of lines of ASCII text in the following form:

media VSN ArchiveSet/owner/filesystem date time

The fields **ArchiveSet**, **owner**, and **filesystem** may be empty depending on the options in the archiver command file.

The date and time tell when the reservation was made and are informational only.

As a VSN is reserved to an archive set during archiving, a reservation line is appended to the file.

During VSN assignment, the label time of candidate VSNs are compared to the reservation time of the VSN. If the label time is more recent than the reservation time, the reservation is removed and the VSN may be repooled or reused.

The ReservedVSNs file is edited to make reservations for relabeled VSNs. A relabeled VSN is identified by a label time that is more recent than the reservation time.

If such an instance occurs, lines referring to previous Archive Set reservations are commented out. In addition, poorly formatted lines are removed.

If a reserved VSNs backup file (/var/adm/samfs/archiver/ReservedVSNs.bak) does not exist, the current reserved VSNs file becomes the backup. In any case, the edited file becomes the new reserved VSNs file. Note that this editing is only done each time the archiver initializes.

EXAMPLE

```
mo B02009 t1428a.1// 1997/12/07 08:28:41
mo B02010 t1428b.1// 1997/12/07 08:28:42
mo B02012 t1428c.1// 1997/12/07 08:28:42
```

NOTE

If this file is removed, all VSN reservations are lost.

SEE ALSO

archiver(1M)

NAME

SamGUI.rsc − SAM GUI resource configuration file

SYNOPSIS

/etc/fs/samfs/SamGUI.rsc

DESCRIPTION

The GUI resource configuration file contains settings and options for the SAM-FS 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 **media**(5) 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), **media**(5)

NOTES

Whenever a new version of SAM-FS GUI tools is 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 − samfs file archiver commands

SYNOPSIS

archiver.cmd

AVAILABILITY

LSCsamfs

DESCRIPTION

Commands for controlling the archiver are read from *archiver.cmd*. 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 VSNs.

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 file system data, (directories, symbolic links, 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)).

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 age of a file.

The archiver command file consists of several types of lines:

General commands
Archive Set assignments
Archive Copy definitions
Archive Set Copy parameters section
VSN associations section

Each of these lines consists of one or more fields separated by white space. Note, in particular, that equal-signs (=) must be surrounded on both sides 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).

General commands.

General commands definitions are identified by the '=' character in the second field or no additional fields.

delay = *time*
Delay for *time* before starting the archiver scan. This delay can be used to wait for all robots to initialize.

Example:

delay = 3m

datadir = *dirname*

Set the name of the directory that will be used for the archiver's data files to *dirname*. Except for **ReservedVSNs**, The data files are used only during archiver execution.

The name of the default data directory is: */etc/fs/samfs/.archive*.

drives = *robot count*

Set the number of drives to use for archiving on *robot* (the robot family set name as defined in the mcf) to *count*. The archiver will use only *count* number of drives in *robot* to create archive copies. This command prevents the archiver from using all drives in a robot and possibly interfering with staging.

The default value is the actual number of drives in the robot.

Example:

drives = gr50 3

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.

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 file system 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 the date, time, media, VSN, 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: */etc/fs/samfs/ar_notify.sh*.

ovflmin = *media minimum_size*

Set the minimum size of a file which will require more than one VSN for media *media* to *minimum_size*. Files to be archived that are smaller than this size will be placed on only a single VSN of the media. Files that are larger than this size will be allowed to overflow one VSN to at least one (but not more than eight) VSN.

If not specified, volume overflow will not take place.

trace = *filename* **[** *event event1 ...* **]**

Set the name of the archiver trace file to *filename*. The trace log file contains a line for each event being traced. In addition, the line contains selectable elements. *option* is the type of event to trace, or element to include in the trace line. To exclude the event, prefix the event 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 queue**.
alloc	Memory allocations.
ipc	Inter process communication.
err	Non-fatal program errors.
files	File actions.
misc	Miscellaneous.
oprmsg	Operator messages.
proc	Process initiation and completion.
queue	Queue contents when changed.
fatal	Fatal syslog messages.
cust	Customer notification syslog or notify file messages.

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 default is no trace file. The default events are:
cust, **err**, **fatal**, **misc**, **proc**. The default message elements program[pid] and time are always included and can't be deselected.

wait

The archiver will not begin archiving until it receives SIGUSR1. This is a mechanism to allow other activities to be performed before archiving begins.

The default is no waiting.

Archive Set assignments

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 command. The Archive Set name is the first field, followed by the path relative to the SAM-FS file system mount point. The path may be quoted with the '"' character. 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 the Archive Set named **no_archive** may be done to prevent the files from ever being archived. Note that since all local archive set assignments are processed after global assignments, if you have local assignments you must make the no_archive assignment locally for each filesystem.

The Archive Set assignments may be followed by Archive Copy definitions.

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 for all files matching the file characteristics on this Archive Set definition. *attributes* may be any of 'a' always, 'n' never, or 'p' partial.

-stage *attributes*
Set the "stage" attributes for all files matching the file characteristics on this Archive Set definition. *attributes* may be any of 'a' associative, 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.

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 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 age is 4 minutes.

The next field is the age of the file when the copy is unarchived. The default is to never unarchive the copy.

Archive set 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.

-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.

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 VSNs associated with an Archive Set for archiving. When a group of files is to be archived at the same time, a VSN with enough space for all the files will be selected for use. This action may cause VSNs to not be filled to capacity.

Selecting this parameter causes the archiver to attempt to fill VSNs 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.

-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 VSN.

direct Direct copy. Copy files directly from the offline VSN to the archive VSN without using the cache. Source VSN and destination VSN must be different and two drives are available.

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 which will require more than one VSN in this Archive Set to *minimum_size*. Files to be archived that are smaller than this size will be placed on only a single VSN of the media. Files that are larger than this size will be allowed to overflow one VSN to at least one VSN.

If not specified, the **ovflmin** value for the media will be used.

-priority age *value*

Set the "Archive age" property multiplier for files in this Archive Set to *value*.

-priority archive_immediate *value*

TP **-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 VSNs" property multiplier for files in this Archive Se t to *value*.

-priority archive_loaded *value*

Set the "Archive VSN loaded" property multiplier for files in this Archive Set t o *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*

-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 VSN loaded" property multiplier for files in this Archive Set to *value*.

-priority stage_overflow *value*
Set the "Multiple stage VSNs" property multiplier for files in this Archive Set to *value*.

-reserve [set | dir | user | group | fs]
This parameter specifies that the VSNs 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 VSNs. A so-called "ReserveName" is assigned to VSNs 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.

This option may be set for all Archive Sets by the global command **reserve**.

-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 age (modification time).

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.

The following archive set parameters control recycling by archive set. If none of the following parameters are set for an archive set, the archive set will not be recycled by the recycler. VSNs which comprise that archive set (unless also assigned to other archive sets) will be recycled by the robot 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 VSNs of useful data. Note that the actual number of VSNs selected for recycling may also be dependant on the -recycle_vsncount parameter.

The default is to not limit the amount of data which will be moved to recycle the archive set.

-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 VSNs associated with the archive set. When the utilization of those VSNs 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.

The default is to not ignore.

-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 to not send any mail.

-recycle_mingain *percent*
This option limits selection of VSNs for recycling to those which would increase their free space by *percent* or more. VSNs 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 VSNs the recycler will schedule for rearchiving so as to clear VSNs of useful data. Note that the actual number of VSNs selected for recycling may also be dependant on the -recycle_dataquantity parameter.

The default is to not limit the number of VSNs selected from the archive set.

VSN pool definitions section.

Collections of VSNs 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.

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.

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 VSN is needed for an Archive Set Copy, each VSN of each robot 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 VSN 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), **arcopy**(1M), **arfind**(1M), **mcf**(4), **recycler**(1m), **regcmp**(3G), **ar_notify.sh**(4), **ReservedVSNs**(4)

NAME

defaults.conf − Setting default values for SAM-FS

SYNOPSIS

/etc/fs/samfs/defaults.conf

DESCRIPTION

The defaults configuration file allows the site to set certain default values for SAM-FS. The **defaults.conf** file is read when **sam-init** is started; it cannot be changed while **sam-init** is running nor when a SAM-FS filesystem is mounted. Temporary changes to the SAM-FS environment values can be made using **samset**(1M).

The **defaults.conf** file consists of a list of keyword/value pairs (*keyword = value*) which set site-definable defaults. Note that equal-signs (=) must be surrounded on both sides by white space. All keyword/values are case-sensitive and must be entered as shown. Values can be either unquoted strings (where string values are expected), or integers in decimal (123), octal, (0123) or hex (0x123) format.

Keywords are:

optical This is used to set the default media type when a generic optical disk (**od**) is requested. See **media**(5) for the accepted media types. (A string value is expected.) The default is **mo**.

tape This is used to set the default media type when a generic tape (**tp**) is requested. See **media**(5) for the accepted media types. (A string value is expected.) The default is **lt**.

log This is the value used for the facility code used for issuing log messages. See **syslog**(3) for the list of accepted values. The default is **LOG_LOCAL7**.

labels If a tape library has a barcode reader, the system can set the tape label equal to the first or the last characters of the barcode (uppercased). The value of **labels** is **barcodes** (for use first part of barcode as label), **barcodes_low** (for use last part of barcode as label) or *read* (for read label from tape). When **labels** is set to **barcodes** or **barcodes_low**, any tape mounted for a write operation that is write enabled, unlabeled and has a readable barcode will have a label written on the tape before the write is started. The default is **read**.

exported_media

If **exported_media** is set to **unavailable**, then media exported from a media changer is set unavailable in the historian. If **exported_media** is set to **available**, then media exported from a media changer is left available in the historian. The default is **available**. See **historian**(7).

attended

If **attended** is **yes**, this assumes an operator is available to mount media that is not flagged as unavailable by the historian. If **attended** is **no**, then any request for media known to the historian will be rejected (unless already mounted). The default is **yes**.

timeout This is used to set the timeout (in seconds) for direct access to removable media. If a process fails to issue an I/O request to the device within this time, the device will be removed from job assignment and the process will receive an ETIME on the next I/O to the device. A value of zero will disable this timeout. The default is 60 seconds.

debug This keyword sets the default for the debug flags used by the SAM_FS daemons for logging messages. The value side of this keyword is a space separated list of debug options. See **samset**(1M) for a list of options. The default is **logging**.

devlog *eq [event ...]* devlog manipulates the device log event flags for device *eq*. These flags control which events get written to the device log files. See **samset**(1M) for a list of options. The default is **none**.

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. The default is 600 seconds (10 minutes).

inodes This keyword is still accepted for backwards compatibility, but it has no effect. See man samfs.cmd(4).

previews
This sets the number of outstanding mount requests. Care should be taken when changing this value. Each entry takes about 500 bytes of shared memory. Together with the **stages** value (see below), this defines the size of the preview shared memory segment. The default is 100.

samrpc If the value is **on**, the RPC API server process, **rpc.sam**, will be automatically started when SAM-FS is started. If the value is **off**, or if this keyword does not appear in the file, **rpc.sam** will not be automatically started.

stage_retries
This keyword is still accepted for backwards compatibility, but it has no effect. See man samfs.cmd(4).

stages This is the number of stage requests that can be outstanding at one time. Each entry takes about 50 bytes. See **previews** above. The default is 1024.

tp_mode
This is the mode (see **chmod**(2)) set for tape drive device nodes when not under control of SAM-FS. When SAM-FS is controlling the drive, the mode bits are 0660.

stale_time
Any request for removable media waiting this number of minutes will be sent an error. Setting this to zero will disable this function. The default is 30 minutes.

*dev*_**delay**
This is the dismount time in seconds for device type *dev*. After a media is loaded on this device type, this time must elapse before the media is dismounted and another media is mounted. The default value for delay is 30 seconds. See **mcf**(4) for the accepted device types.

*dev*_**unload**
This is the unload wait time in seconds for device type *dev*. This is the time the media changer daemons will wait after the device driver returns from a SCSI unload command. This gives the media changer time to eject the media, open the door, etc, before the daemon will command the changer to remove the media. This value should be set to the longest time needed for the worst case media changer configured. See the example file (**/opt/LSCsamfs/examples/defaults.conf**) supplied with the installation for the default values. Any device not in the example file defaults to zero.

*dev*_**blksize**
This is the default blocksize for tapes of type *dev*. The released default blocksizes are as follows:

tp_blksize = 128

lt_blksize = 128

st_blksize = 128

vt_blksize = 128

xt_blksize = 16

dt_blksize = 16

se_blksize = 128

d3_blksize = 256

d2_blksize = 1024

ib_blksize = 256

i7_blksize = 128

so_blksize = 1024

at_blksize = 128

sg_blksize = 256

fd_blksize = 256

The value for blocksize can be one of 16, 32, 64, 128, 256, 512, 1024, or 2048. This value is multiplied by 1024 to arrive at the actual blocksize. The default may be overridden when manually labeling a tape (see **tplabel**(1M)). The default is used when no blocksize is specified or during automatic labeling when **labels = barcodes** has been specified. See **mcf**(4) for the accepted device types.

operator

The name of the group that will be granted operational privileges within the GUI tools (previewtool, devicetool, and robottool) and command queues. Only one group name may be specified. Users must have their effective group ID set to this group in order to gain operational privileges.

oper_privileges

By default, members of the operator group will not have the following privileges: media labeling, performing slot 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. Using **oper_privileges** keyword, along with the following values will allow operator group members access to those privileges. Multiple values must be separated by a space:

label Allow labeling of media.

slot Allow mounting, unloading, and movement of media within a library.

fullaudit
Ability to perform a full library audit.

state Ability to change the device state. Operator group members can ON devices, regardless of this setting.

clear Ability to clear media mount requests.

options Ability to change display options, such as the media displayed in the previewtool.

refresh Ability to turn off automatic display refresh.

all Grants all the above privileges.

EXAMPLES

Here is a sample configuration file.

```
optical = mo
debug = logging debug timing
tape = lt
log = LOG_LOCAL7
timeout = 30
stages = 750
stage_retries = 3
inodes = 2048
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
```

SEE ALSO

request(1), **samset**(1M), **tplabel**(1M), **chmod**(2), **syslog**(3) **mcf**(4), **media**(5), **historian**(7)

NAME

dev_down.sh − SAM-FS device down notification script

SYNOPSIS

/etc/fs/samfs/dev_down.sh

AVAILABILITY

LSCsamfs

DESCRIPTION

/etc/fs/samfs/dev_down.sh is a script which is executed by robots(4) when a device is marked "down" or "off".

DEFAULT FILE

As released, **/opt/LSCsamfs/examples/dev_down.sh** contains a script which will mail root with the relevant information.

EXAMPLE

The following is an example **/etc/fs/samfs/dev_down.sh** file:

```
#!/bin/sh
#

#   /etc/fs/samfs/dev_down.sh - Take action in the event
#       a device is marked down by SAM.
#
#   arguments:  $5: device identifier
#
# Change the email address on the following line to send
# email to the appropriate recipient.
/usr/ucb/mail -s "SAM-FS Device downed" root <<EOF
‘date‘
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

robots(4)

NAME

inquiry.conf − SCSI inquiry strings to SAM-FS device type

SYNOPSIS

/etc/fs/samfs/inquiry.conf

DESCRIPTION

The inquiry configuration file maps a SCSI device into the SAM-FS device type.

The **inquiry.conf** file contains the vendor identification and product identification reported by a SCSI device in response to an inquiry command. Each entry is made up of three quoted fields separated by a comma and/or white space and optionally followed by a comment:

"*vendor id*", "*product id*", "*SAM-FS name*" *#comment*

vendor id and *product id* are the vendor identification (8 characters) and product identification (16 characters) as reported in the inquiry data. *SAM-FS name* is the SAM-FS device name (see below).

Trailing spaces do not need to be supplied in either id field. Any occurrence of " (double quote) , (comma) or \ (back slash) in any id field should be prefaced with the escape character \ (back slash). Blank lines and lines beginning with # are ignored.

SAM-FS Device Names

acl2640
ACL 2640 tape library

acl452
ACL 4/52 tape library

adic448
ADIC 448 tape library

ampexd2
Ampex D2 tape

ampex410
Ampex 410 Media Changer

ampex810
Ampex 810 tape library

archdat
Archive Python 4mm DAT

atlp3000
ATL P3000 tape library

cyg1803
Cygnet Jukebox 1803

dlt2000
Digital Linear Tape (both 2000 and 4000 series)

dlt2700
Digital Linear Tape (both 2000 and 4000 series)

docstor
DISC automated library

exb210
Exabyte 210 tape library

exb8505
Exabyte 8505 8mm cartidge tape

fujitsu_128
Fujitsu Diana4 128 track tape

grauaci
GRAU media library

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

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

rap4500
Laser Magnetic RapidChanger 4500

rsp2150
Metrum RSP-2150 VHS video tape

sonyait
Sony AIT tape

sonydms
Sony Digital Mass Storage tape library

sonydtf
Sony DTF tape

speclog
Spectra Logic Libraries

stk4280
StorageTek 4280 Tape

stk9490
StorageTek 9490 Tape

stk9840
StorageTek 9840 Tape

stkapi
StorageTek API library

stkd3
StorageTek D3 Tape

stk97xx
StorageTek 97*xx* Media Libraries

EXAMPLES

Here is a sample configuration file. The existence of a device in this sample file does not imply that the device is supported by SAM-FS.

```
"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
```

SEE ALSO

mcf(4)

NOTES

Whenever a new version of SAM-FS is installed, the existing **inquiry.conf** file is copied to **inquiry.conf.MMDDYY** for reference and backup purposes.

WARNINGS

During device identification, the vendor and product id's 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.

This interface is supplied to circumvent problems that occur when hardware vendors change the values returned for vendor and product id's. For example, some hardware vendors will return a different value for product id if the hardware is supplied by an OEM. Mapping untested hardware to a SAM-FS name is not supported by LSC.

NAME

mcf − Master configuration file for SAM-FS

SYNOPSIS

/etc/fs/samfs/mcf

DESCRIPTION

The **mcf** file defines the devices and family sets used by SAM-FS. The format of the **mcf** file is:

equipment identifier	*equipment ordinal*	*equipment type*	*family set*	*device state*	*additional parameter*

equipment identifier, *equipment ordinal* and *equipment type* are required for each entry. Comments begin with a '#' and blank lines are ignored. The fields in the file are white space-separated. A '-' is used to indicate no entry in a field. *equipment identifier* can be no longer than 127 characters. *equipment ordinal* can range from 1 to 65535. The *equipment ordinal* should be kept low in order to keep the internal tables used by SAM-FS small. Valid *device states* are: 'on', 'off', 'unavail', or 'down'. *device state* defaults to 'on'.

The following *equipment types* are disk cache family set. The *equipment identifier* is used in **mount**(1M) processing as the 'device to mount' (the first field in /etc/vfstab) for the mount point (see **vfstab**(4)). *family set* is required and is used to define the magnetic disks that make up the family set.

ms Disk cache family set for **md** magnetic disk.
ma Disk cache family set consisting of one or more **mm** and one or more **mr** magnetic disks.
ma Disk cache family set consisting of one or more **mm** and one or more **gx** magnetic disks.

The valid *equipment types* are magnetic disk. This defines a magnetic disk that is part of a family set. *equipment identifier* is the path to the ''special file'' (such as /dev/dsk/c?t?d?s?). The *family set* is required and must match a *family set* defined on a **ms** entry. *additional parameter* is required and is the path to the ''raw special file'' (such as /dev/rdsk/c?t?d?s?).

md Magnetic disk that is part of a **ms** disk cache family set. This device is allocated with small and large daus.
mm Magnetic disk that is part of a **ma** disk cache family set. This device is allocated with the meta data.
mr Magnetic disk that is part of a **ma** disk cache family set. This device is allocated with the file data.
gx Magnetic disk that is part of a **ma** disk cache family set. This device is allocated with the file data. The x is a decimal number 0 .. 127. x identifies a striped group of devices. These devices must be the same size. It is not possible to samgrowfs a striped group. However, it is possible to add additional striped groups.

The following *equipment types* define libraries and removable media.

rb Generic SCSI robot (libraries automatically configured by SAM-FS)
hp HP Optical libraries
ml DLT2700 Tape libraries
cy Cygnet libraries
dm Sony DMS library
ds DocuStore Optical Disk libraries
me Metrum and Mountain Gate libraries
ac Odetics ACL libraries
eb EXABYTE libraries
ad ADIC libraries
a1 Ampex ACL libraries
a2 Ampex DST810 library
sl Spectra Logic libraries

s9 STK 97xx series
ic IBM 3570 Media Changer

These are the ‘‘SCSI-robotic’’ media loaders. If you use "rb", SAM-FS will set the appropriate type based on the SCSI vendor code. *equipment identifier* is the path to the ‘‘special file’’ (such as /dev/samst/c?t?u?) for the samst device driver (see **samst**(7)). *family set* is required and is used to associate devices with the robot. *additional parameter* is required and is the full path name of the library catalog file. This file is used to store information about each piece of media in the library.

gr This is the GRAU ABBA media changer. *equipment identifier* is the path to the ‘‘parameters file’’ for *grauaci* (see **grauaci**(7)). *family set* is required and is used to associate devices with the robot. *additional parameter* is required and is the full path name of the library catalog file. This file is used to store information about media in the library.

fj This is the Fujitsu Koala media changer. *equipment identifier* is the path to the ‘‘parameters file’’ for *fujitsulmf* (see **fujitsulmf**(7)). *family set* is required and is used to associate devices with the robot. *additional parameter* is required and is the full path name of the library catalog file. This file is used to store information about media in the library.

im This is the IBM 3494 interface. *equipment identifier* is the path to the ‘‘parameters file’’ for *ibm3494* (see **ibm3494**(7)). *family set* is required and is used to associate devices with the media changer. *additional parameter* is required and is the full path name of the library catalog file. This file is used to store information about media in the changer.

sk This is the STK ACSLS interface. *equipment identifier* is the path to the ‘‘parameters file’’ for *stk* (see **stk**(7)). *family set* is required and is used to associate devices with the media changer. *additional parameter* is required and is the full path name of the library catalog file. This file is used to store information about media in the changer.

hy This is the SAM-FS historian. *equipment identifier* is the string "historian". *family set* must be set to "-". *additional parameter* is required and is the full path name of the catalog file. This file is used to store information on the media being handled by the *historian* (see **historian**(7)).

od Generic optical disk (disks automatically configured by SAM-FS)
o2 12 inch WORM
wo 5 ¼ inch optical WORM
mo 5 ¼ inch erasable optical
mf IBM Multi Function
This is the set of optical devices. If you use "od", SAM-FS will set the appropriate type based on the SCSI vendor code. *equipment identifier* is the path to the ‘‘special file’’ (such as /dev/samst/c?t?u?) for the samst device driver (see **samst**(7)). The *family set* is used to associate the device with the robot which has the same *family set*. If the family set is ’-’, then the device is assumed to be operator mounted.

tp Generic tape (tapes automatically configured by SAM-FS)
lt Digital linear tape
st STK 3480
vt Metrum VHS (RSP-2150)
xt 8mm Exabyte (850x) tape
dt 4mm DAT tape
se STK 9490
sg STK 9840

d3 STK D3
d2 AMPEX DST
fd Fujitsu M8100
ib IBM 3590
i7 IBM 3570
so SONY DTF
at SONY AIT

This is the set of tape drives. If you use "tp", SAM-FS will set the appropriate type based on the SCSI vendor code (does not apply to the AMPEX DST (see **dst**(7))). *equipment identifier* is the path to the ''raw'' device (such as /dev/rmt/?bn). You must specify the BSD no-rewind path (see **mtio**(7)). If the device supports compression, then that path should be specified for better tape usage. The *family set* is used to associate the device with the robot which has the same *family set*. If the family set is '-', then the device is assumed to be operator mounted. The *additional parameter* is required if the *equipment identifier* is not of the form "/dev/rmt/*" (the standard 'st' device driver) and is the path to the ''special file'' (such as /dev/samst/c?t?u?) for the samst device driver (see **samst**(7)). While SAM-FS has access to a tape device no other user should be allowed access the device. Since the tape driver (see **st**(7)) supplies a number of paths to a single device, the system administrator should change the access permissions for all such paths to not allow access (see **chmod**(1)). SAM-FS will 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 will be set to the value of ''tp_mode'' in the defaults.conf file (see **defaults.conf**(4)).

ss This is the SAM-Remote server. The *equipment identifier* is the path to the server configuration file (see **sam_remote**(7)). The *family set* is required and is used by the clients to associate the device with the server of the same *family set* name.

sc This is the SAM-Remote client. The *equipment identifier* is the path name to the client configuration file (see **sam_remote**(7)). The *family set* is required and is used by the clients to associate the device with the server of the same *family set* name. *additional parameter* is required and is the full path name of the client's catalog file.

rd This is the SAM-Remote pseudo-device. The *equipment identifier* is the path to the pseudo-device (such as /dev/samrd/rd*). The *family set* is required and is used by the clients to associate the device with the server of the same *family set* name.

SEE ALSO

chmod(1), **build_cat**(1M), **dump_cat**(1M), **mount**(1M), **mount_samfs**(1M), **defaults.conf**(4), **inquiry.conf**(4), **vfstab**(4), **mtio**(7), **dst**(7), **fujitsulmf**(7), **grauaci**(7), **historian**(7), **ibm3494**(7), **samst**(7), **sam_remote**(7), **st**(7), **stk**(7)

NAME

notify.sh − SAM-FS exception notification script template

SYNOPSIS

/etc/fs/samfs/notify.sh

AVAILABILITY

LSCsamfs

DESCRIPTION

/etc/fs/samfs/notify.sh is a template for a script executed by SAM-FS applications when abnormal or exceptional events are encountered. It is intended as a means to allow site specific control of these events.

/etc/fs/samfs/notify.sh is executed with the following arguments:

The first two arguments are the program and process id of the program causing the exception.

The third 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 fourth argument is the message number as found in the message catalog.

The fifth argument is the text of the translated message string.

DEFAULT FILE

As released, **/etc/fs/samfs/notify.sh** is a script which will log the event to **syslog** using the **/usr/bin/logger** command.

Additionally, the **emerg**, **alert**, **crit**, and **err** keywords generate mail to the root account, echoing the message string.

For the benefit of the archiver, two **warning** messages have special behaviour keyed on message numbers 4010 and 4011. Two directories are created in **/etc/fs/samfs/.archive** to handle the exceptions of no VSNs or no space of an archive set. These directories are named NoVsns and NoSpace, respectively, and will be populated with zero-sized files with the same name as their corresponding archive sets. This is to prevent repeated email and reporting to syslog when these conditions occur. It is the administrator's responsibility to remove these files when the specific condition no longer exists.

EXAMPLE

See **/opt/LSCsamfs/examples/notify.sh**.

SEE ALSO

archiver(1M), **notify**(1M), **archiver.cmd**(4)

NOTES

An existing copy of **/etc/fs/samfs/ar_notify.sh** will be preserved when SAM-FS is upgraded. However, an existing copy of **/opt/LSCsamfs/examples/notify.sh** will not.

NAME

preview.cmd − SAM-FS preview commands file

SYNOPSIS

/etc/fs/samfs/preview.cmd

AVAILABILITY

LSCsamfs

DESCRIPTION

Archive and stage requests which could not be immediately satisfied go to the preview area for future consideration. If no preview command file (**preview.cmd**) present requests are satisfied in the first in first out order. A user can control the scheduling of preview requests, thus overriding the default FIFO behavior, by entering commands in preview.cmd.

preview.cmd contains commands for modifying preview requests priorities. The commands allow user to increase priority for specific VSNs and change archive request priorities based on the filesystem states regarding High Water Mark (HWM) and Low Water Mark (LWM). These commands are read by **sam-init** at startup and all values specified are stored in the shared memory. The commands cannot be changed while **sam-init** is running.

These commands are given one per line. Note that equal-signs (=) must be surrounded on both sides by white space. Comments begin with a # and extend through the end of the line. There are two type of commands: global and filesystem specific. Global commands are always apply to all filesystems. Filesystem specific 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

The following commands are available.

vsn_priority = *value*

This command is global and indicates the *value* priority will increase by for VSNs marked as high priority VSNs (see **chmed** (1M)). The default value for **vsn_priority** is 0.

age_priority = *factor*

This command is global and indicates the *factor* to be applied to the time (in seconds) request is waiting in preview area to be satisfied. This parameter being more than 1.0 will increase weight of the time in calculating the total priority, while less that 1.0 will decrease time factor. See below how total priority is calculated. The default value for **age_priority** is 1.

fs = *filesystem-family-set-name*

This command tells that the following commands apply to the indicated *filesystem-family-set-name* only.

lwm_priority = *value*

This command is filesystem specific and indicates the *value* priority will increase by for archiving requests vs. staging when filesystem is below LWM level. The default value for **lwm_priority** is 0.

lhwm_priority = *value*

This command is filesystem specific and indicates the *value* priority will increase by for archiving requests vs. staging when filesystem crossed LWM up, but still lower than HWM level, which means that filesystem is filling up. The default value for **lhwm_priority** is 0.

hlwm_priority = *value*

This command is filesystem specific and indicates the *value* priority will increase by for archiving requests vs. staging when filesystem crossed HWM down, but still higher than Low Water Mark level, which means that releaser wasn't able to free enough space to get filesystem below LWM. The default value for **hlwm_priority** is 0.

hwm_priority = *value*

This command is filesystem specific and indicates the *value* priority will increase by for archiving requests vs. staging when filesystem crossed HWM level up, which means that releaser is running at this point. The default value for **hwm_priority** is 0.

The total preview request priority is the sum of all priorities and calculated as follows:

priority = vsn_priority + wm_priority + age_priority * time_in_sec

All priorities are floating point numbers.

EXAMPLES

This example file sets **vsn_priority** and **hwm_priority** for the samfs1 filesystem. Any other filesystems not specified here will use the default priority for HWM. All filesystem will use the default priorities for LWM and state between LWM and HWM.

```
vsn_priority = 1000.0
fs = samfs1
hwm_priority = 100.0
```

Next example provides priority factors for all filesystems, except samfs3, which sets the HWM priority explicetely

```
hwm_priority = 1000.0
hlwm_priority = 500.0
lhwm_priority = 100.0
fs = samfs3
hwm_priority = 200.0
```

SEE ALSO

sam-init(1M), **chmed**(1M)

NAME

recycler.cmd − SAM-FS recycler commands file

SYNOPSIS

/etc/fs/samfs/recycler.cmd

AVAILABILITY

LSCsamfs

DESCRIPTION

Commands for controlling **recycler** are read from **/etc/fs/samfs/recycler.cmd**. These commands are given one per line. Note that equal-signs (=) must be surrounded on both sides by white space.

logfile = *filename*

Set the name of recycler's log file to *filename*, showing the robots' 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 **recycler(1m)** for more information.

no_recycle *media-type VSN-regexp [VSN-regexp...]*

Disallow the recycler from recycling the VSNs which match the *media-type* and the regular expression(s).

robot-family-set robot-high-water VSN-minimum-percent-gain
options

This command establishes the high-water mark for the media utilization in the indicated *robot*, specified as an integer percentage of total capacity. 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 **recycler** from selecting any candidates from the specified media changer. **mail** *mailaddress* - which will cause **recycler** to mail a message to the indicated *mailaddress* when a robot's media utilization exceeds the high-water mark. Ommission of *mailaddress* prevents any mail from being sent. *robot-family-set* is the name given as the fourth field in the **mcf** file line defining the media changer for which you wish to set the parameters.

ARCHIVER'S COMMAND FILE

The archiver's command file, */etc/fs/samfs/archiver.cmd*, can also specify recycling parameters for archive sets. Each archive set which has recycling parameters applied in *archiver.cmd* will be considered as a small robot containing just the VSNs which the archiver assigns to the archive set. See **archiver.cmd**(4) for more information. It is not legal to specify an archive set name in the *recycler.cmd* file.

DEFAULT FILE

If there is no /etc/fs/samfs/recycler.cmd file, then, for each robot, a line is constructed:

robot 95 50 ignore mail root

and logging is disabled.

EXAMPLE

The following is an example /etc/fs/samfs/recycle.cmd file:

```
logfile = /var/adm/recycler.log
lt20 75 60 ignore
hp30 90 60 mail root
gr47 95 60 ignore mail root
no_recycle lt DLT.*
```

The results of **recycler** operation are found in /var/adm/recycler.log. Three robots are defined with various high-water marks. The first robot 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 robot is recycled (that is, VSNs are emptied of valid archive images and relabeled) and root is sent e-mail when the robot exceeds the 90% high-water mark. The third robot 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

recycler(1M) **archiver.cmd**(4)

NAME

recycler.sh − SAM-FS recycler post-process shell escape

SYNOPSIS

/etc/fs/samfs/recycler.sh

AVAILABILITY

LSCsamfs

DESCRIPTION

/etc/fs/samfs/recycler.sh is a script which is executed by the recycler when it has finished draining a VSN of all known active archive images.

DEFAULT FILE

As released, **/etc/fs/samfs/recycler.sh** contains a script which will mail root with the relevant information.

EXAMPLE

The following is an example **/etc/fs/samfs/recycle.sh** file:

```
#!/bin/csh

#   /etc/fs/samfs/recycler.sh - post-process a VSN after
#   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 number 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/LSCsamfs/sbin/chmed -R $5 $2
    /opt/LSCsamfs/sbin/chmed -W $5 $2
    /opt/LSCsamfs/sbin/${1}label -vsn $2 -old $2 \
        -slot $3 $4
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

recycler(1M), **tplabel**(1M), **odlabel**(1M)

NAME

releaser.cmd − SAM-FS releaser commands file

SYNOPSIS

/etc/fs/samfs/releaser.cmd

AVAILABILITY

LSCsamfs

DESCRIPTION

Commands for controlling **releaser** are read from **/etc/fs/samfs/releaser.cmd**. These commands are given one per line. Note that equal-signs (=) must be surrounded on both sides by white space. 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

The following commands are available.

fs = *filesystem-family-set-name*

This command tells the releaser the the following commands apply to the indicated *filesystem-family-set-name* only.

no_release

Tells the releaser to not actually release any files. This option is useful when you are tuning the priority weights. See also **display_all_candidates**.

logfile = *filename*

Set the name of releaser's log file to *filename*.

display_all_candidates

Show all inodes as they are encountered. For each, display the releaser priority. Useful in conjunction with the **no_release** option to judge the effect of changing the priority weights.

The following weights are used to calculate the release priority of each file in the filesystem. Each file's priority is composed of two parts: age-priority and size-priority.

There are two ways of calculating the age-priority. The first method multiplies a *weight_age* times the most recent of the following ages: access, modify, and residence-change. ("access age", for example, is defined as the current time minus the file's last-access time. The value of age is in units of 60-second minutes.)

The second method allows you to specify separate weights for each of these ages.

Finally, the age-priority calculated above is added to the size-priority. The size-priority is simply the size of the file (in 4k-Byte blocks) times the *weight_size* factor.

The following are detailed descriptions of the weights. Note that if you specify *weight_age* for a given filesystem, then you may not specify *weight_age_access*, *weight_age_modify*, or *weight_age_residence*, and the converse. Each weight is a floating-point value between 0.0 and 1.0, inclusive.

weight_age = *float*

Sets the weight factor for the overall age of the file to *float*. *weight_age* 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.

weight_age_access = *float*

Sets the weight factor for the access age of the file to *float*. *weight_age_access* is multiplied by the the file's access age (expressed in minutes). This product, added to the sum of the products of the modify and residence-change ages times their respective weights, becomes the age

component of the file's release priority.

weight_age_modify = *float*
Sets the weight factor for the modify age of the file to *float*. *weight_age_modify* is multiplied by the the file's modify age (expressed in minutes). This product, added to the sum of the products of the modify and residence-change ages times their respective weights, becomes the age component of the file's release priority.

weight_age_residence = *float*
Sets the weight factor for the residence-change age of the file to *float*. *weight_age_residence* is multiplied by the 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 times their respective weights, becomes the age component of the file's release priority.

weight_size = *float*
Sets the weight factor for the size of the file to *float*. *weight_size* is multiplied by the size of the file in 4k-Byte blocks to arrive at the size component of the file's release priority.

EXAMPLES

This example file sets **weight_age** and **weight_size** for the samfs1 filesystem. No releaser log will be produced. Any other filesystems not specified here will use the weights specified in */etc/vfstab*.

```
fs = samfs1
weight_age = .45
weight_size = 0.3
```

Next, this example provides weights for all filesystems, regardless of the settings in */etc/vfstab*. All filesystem releaser runs will be logged to */var/adm/releaser.log*.

```
weight_age = 1.0
weight_size = 0.03
logfile = /var/adm/releaser.log
```

This example specifies weights and log files for each filesystem.

```
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
```

This example is identical in function to the above, but uses global specification of **weight_size**.

```
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

mount_samfs(1M) **release**(1),

NAME

samfs.cmd − SAM-FS mount commands file

SYNOPSIS

/etc/fs/samfs/samfs.cmd

AVAILABILITY

LSCsamfs

DESCRIPTION

Commands for controlling **samfs** mount parameters are read from **/etc/fs/samfs/samfs.cmd**. These commands serve as defaults, and can be superceded by parameters on the mount command. These commands are given one per line. Note that equal-signs (=) must be surrounded on both sides by white space. 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

The following commands are available.

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 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*.

forcedirectio

If *forcedirectio* is specified, the default I/O mode is direct. This means data is transferred directly between the user's buffer and disk. The *forcedirectio* option should only be selected if the filesystem is used for large block aligned sequential I/O. See directio(3C), setfa(1), sam_setfa(3) and sam_advise(3). The default I/O mode is buffered (uses the page cache).

stage_retries = *n*

This is the number of stage retries attempted per archive copy, when particular errors are encountered. The number of stage retries may not exceed 20. Setting the number of stage retries to zero prevents a retry from being initiated. The default is 3.

high = *n*

Set the high water mark for disk cache utilization to *n* percent. When the amount of space used on the disk cache reaches *n* percent, SAM-FS will start the **releaser** daemon (see **releaser**(1M)). The default is 80%.

low = *n* Set 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 **releaser** daemon will stop releasing disk space. The default is 70%.

weight_size = *x*

Set the size-based weighting factor for space releasing (see **releaser**(1M)) to *x*. The default value is 1.00. This is a floating point value. The parameter *weight* means *weight_size* for backwards compatibility.

weight_age = *x*

Set the age-based weighting factor for space releasing (see **releaser**(1M)) to *x*. The default value is 1.00. This is a floating point value.

readahead = *n*

Set the maximum readahead and writebehind value to *n*. *n* is in units of kilobytes. Readahead

specifies the maximum number of bytes that can be read ahead or written behind by the filesystem. *n* is an integer from 16 to 8192; the default is 128 (131072 bytes). Note, this parameter replaces the mount option maxcontig.

flush_behind = *n*
Set the maximum flush behind value to *n*. *n* is in units of kilobytes. Modified pages which are being written sequentially are written to disk asyncronously to help the Solaris VM layer keep pages clean. *n* is an integer from 16 to 8192; the default is 64 (65536 bytes).

stage_flush_behind = *n*
Set the maximum stage flush behind value to *n*. *n* is in units of kilobytes. Stage pages which are being staged are written to disk asyncronously to help the Solaris VM layer keep pages clean. *n* is an integer from 16 to 8192; the default is 64 (65536 bytes).

partial = *n*
Set the default partial-release size for the filesystem to *n* kilobytes. The partial-release size is used to determine how many bytes at the beginning of a file marked "partial release" should be retained on disk cache when the file is released. The user may override the default on a file-by-file basis by specifying a size when marking a file partial-release. (See **release**(1)). *n* is an integer from 8 to maxpartial; the default is 16.

maxpartial = *n*
Set the maximum partial-release size for the filesystem to *n* kilobytes. The partial-release size cannot be set larger than maxpartial. *n* is an integer from 0 to 102400; the default is 16.

partial_stage = *n*
Set the partial_stage size for the filesystem to *n* kilobytes. For a partial-release file, this value gives the offset in the file past which access will result in the entire file being staged to disk. *n* is an integer from 0 to maxpartial; the default is *partial.*

stage_n_window = *n*
Set the stage_n window size for the filesystem to *n* kilobytes. For a stage_n file, this is the size that is staged in to the disk cache at any one time. *n* is an integer from 64 to 2048000; the default is 256.

wr_throttle = *n*
Set the maximum number of outstanding write bytes to one file to *n* kilobytes. If *n* is set to 0, there is no limit. The default is 1/4 physmem.

notrace | trace
notrace disables filesystem tracing. *trace* enables filesystem tracing. The default is *trace*.

stripe = *n*
Set the stripe size for the filesystem to *n* disk allocation units (DAU). *n* is an integer from 0 to 255. The default *n* is 1 on the *ms* equipment type filesystem. The default *n* is 0 on the *ma* equipment type filesystem. If *n* is 0, files are roundrobined on each slice. (See **mcf**(4) for a description of the *ms* and *ma* equipment type filesystems.)

The following parameters are only supported by the *ma* equipment type filesystem. (See **mcf**(4) for a description of the *ma* equipment type filesystem.)

qwrite If *qwrite* is specified, the filesystem enables simultaneous reads and writes to the same file from different threads. If *qwrite* is not specified, the filesystem disables simultaneous reads and writes to the same file. This is the mode defined by the UNIX vnode interface standard which gives exclusive access to only one writer while other writers/readers must wait. The *qwrite* option should be selected only if users of the filesystem handle multiple simultaneous transactions to the same file, such as database applications. This option will improve I/O performance by queuing multiple requests at the drive level.

shared_writer

Sets filesystem to type writer. For each filesystem, there should be only 1 system which has it mounted shared_writer. Directories and inodes are always written through to disk if shared_writer is set.

shared_reader

Sets filesystem to type reader. The filesystem must be mounted read-only to set shared_reader. There is no limit to the number of systems which can have the same filesystem mounted with shared_reader. Directories and inodes are always read from disk if shared_reader is set.

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

directio(3C), **mount_samfs**(1M), **mcf**(4), **release**(1), **setfa**(1), **sam_setfa**(3), **sam_advise**(3)

NAME

media − List of media supported by SAM-FS

AVAILABILITY

LSCsamfs

DESCRIPTION

The following media are supported by SAM-FS:

Tapes

dt	4mm DAT tape
fd	Fujitsu M8100 128track
ib	IBM 3590
i7	IBM 3570
lt	digital linear tape (DLT)
so	Sony DTF
st	STK 3480
se	STK 9490
sg	STK 9840
d3	STK Redwood SD-3
tp	default tape type set by "tape = *xx*" in defaults.conf
vt	Metrum VHS tape
xt	Exabyte 8mm tape
d2	Ampex DST310 (D2) tape
at	Sony AIT tape

Optical disks

mo	5 1/4 in. erasable optical disk (see note)
o2	12 in. WORM optical disk
od	default optical disk type set by "optical = *xx*" in defaults.conf
wo	5 1/4 in. WORM optical disk

NOTE ON MAGNETO-OPTICAL DISKS

The current release of SAM-FS supports M-O disks with 512, 1024, and 2048 byte sectors.

SEE ALSO

defaults.conf(4)

NAME

acl2640 − The ACL2640 Automated Tape Library

AVAILABILITY

LSCsamfs

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 SAM-FS. Later version of SAM-FS will support automatic cleaning.

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 SAM-FS

SEE ALSO

export(1M), **import**(1M), **mcf**(4), **robots**(1M), **robottool**(1M)

NAME

acl452 − The ACL 4/52 Automated Tape Library

AVAILABILITY

LSCsamfs

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 SAM-FS. Auto-load may be used during initial loading of cartridges as long as SAM-FS 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. SAM-FS 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 in SAM-FS since it must wait for the library to become online before any commands may be issued.

FILES

mcf	The configuration file for SAM-FS

SEE ALSO

export(1M), **import**(1M), **move**(1M), **mcf**(4), **robots**(1M), **robottool**(1M)

NAME

atlp3000 − The ATL P3000 Automated Tape Library

AVAILABILITY

LSCsamfs

DESCRIPTION

The ATL P3000 tape library supports 326 DLT tape cartridges and up to 16 DLT tape drives. The library has an import/export unit that may be used to import or export up to 12 cartridges into or out of the library.

CONFIGURATION

The ATL P3000 should be configured with auto-clean and auto-load disabled and with bar-code-labels enabled when running SAM-FS.

IMPORT/EXPORT MEDIA

To import media, follow the ATL P3000 User's Guide on loading the load port. Then issue the **import**(1M) command or function the import button in **robottool**(1M).

To export media, use the **export**(1M) command or the export button in **robottool** to move media into the export unit then follow the instructions in the ATL P3000 User's Guide.

FILES

mcf The configuration file for SAM-FS

SEE ALSO

export(1M), **import**(1M), **mcf**(4), **robots**(1M), **robottool**(1M)

NAME

dst − The AMPEX DST interface

AVAILABILITY

LSCdst

DESCRIPTION

Using AMPEX DST tape drives with SAM-FS requires the installation of the AMPEX DST Tape Device Driver supplied by AMPEX *BEFORE* the LSCdst 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

SAM-FS requires version 3.4 of the AMPEX DST Tape Device Driver; earlier versions will not work.

SAM-FS 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 Koala LMF Automated Tape Library

AVAILABILITY

LSCsamfs

DESCRIPTION

fujitsulmf is the SAM-FS interface to the Fujitsu Koala 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 Koala 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/fs/samfs/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/fs/samfs/lmf50     50      fj      fj50   -       /etc/fs/samfs/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 SAM-FS is not running and **sam-init** has been shut-down (see **sam-init**(1M)).

FILES

mcf The configuration file for SAM-FS.

/opt/LSCsamfs/lib/liblmf2.so

The LMF library supplied by Fujitsu.

SEE ALSO

build_cat(1M), **dump_cat**(1M), **export**(1M), **import**(1M), **mcf**(4), **robots**(1M), **libmgr**(1M)

NAME

grauaci − The GRAU Automated Tape Library through the ACI

AVAILABILITY

LSCsamfs

DESCRIPTION

grauaci is the SAM-FS interface to the GRAU ABBA library. This interface utilizes the DAS/ACI 2.0 interface supplied by GRAU. For more information on DAS/ACI, see the DAS/ACI 2.0 Interfacing Guide and the DAS Administration Guide. Both manuals are supplied by GRAU.

CONFIGURATION

It is assumed that the site has the DAS server configured and operating with the ABBA library. In the DAS configuration file for this client, the 'avc' (avoid volume contention) and the dismount parameters should both be set true.

The "equipment identifier" field in the **mcf** file, (see **mcf**(4)), 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. All keywords and values are case-sensitive and must be entered as shown.

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, where:

drivename
is the drivename as configured in the DAS configuration file.

path is the pathname to the device. This name must match the "equipment identifier" of an entry in the **mcf** file.

If the ABBA library contains different media types, then there must be a separate media changer for each of the media types. Each media changer will have a unique client name in the DAS configuration, a unique library catalog and a unique parameters file.

EXAMPLE

Here are sample parameters files and mcf entries for a GRAU library supporting DLT tape and HP optical drives.

```
#
# This is file: /etc/fs/samfs/grau50
#
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/fs/samfs/grau60
#
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 a GRAU library - DLT
#
/etc/fs/samfs/grau50    50      gr      gr50   -       /etc/fs/samfs/gr50
/dev/rmt/0cbn           51      lt      gr50   -       /dev/samst/c2t5u0
/dev/rmt/1cbn           52      lt      gr50   -       /dev/samst/c2t6u0

#
# Sample mcf file entries for a GRAU library - HP optical
#
/etc/fs/samfs/grau60    60      gr      gr60   -       /etc/fs/samfs/gr60
/dev/samst/c1t1u0       61      od      gr60   -
```

IMPORT/EXPORT

Since the physical adding and removing of media in the GRAU library is done with DAS 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. *grauaci* will verify that DAS 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 the GRAU. **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 SAM-FS is not running and **sam-init** has been shut-down (see **sam-init**(1M)).

FILES

mcf The configuration file for SAM-FS.

/opt/LSCsamfs/lib/libaci.so
The ACI library supplied by GRAU.

SEE ALSO

build_cat(1M), **dump_cat**(1M), **export**(1M), **import**(1M), **mcf**(4), **robots**(1M), **libmgr**(1M)

NAME

historian − The SAM-FS historian

AVAILABILITY

LSCsamfs

DESCRIPTION

historian is the SAM-FS daemon that keeps track of media that has been exported from a media changer and media that has been mounted on manually mounted devices.

CONFIGURATION

The **historian** acts like a media changer but since there are no devices associated with it, has no family set name. If there is no historian configured in the mcf file (see **mcf**(4)) one will be created as:

```
historian    n+1   hy   -   -   /etc/fs/samfs/sam_historian
```

Where *n+1* is the highest equipment number defined in the mcf file plus 1.

The catalog for the historian will be created with 1000 entries when the historian first starts and can grow during execution. Each time the catalog fills, 1000 entries of 120 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 SAM_FS file system.

Two configuration parameters in the defaults.conf file (see **defaults.conf**(4)) affect the way the historian will react to requests for media or requests to add media to its 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's catalog will be sent back to the file system with an error (ESRCH). Any request for media currently mounted on a manually mounted drive will be accepted no matter what the state of the attended or unavailable flags are.

EFFECT ON SAM_FS

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's catalog.

export (see **export**(1M)) is used to remove entries from the historian's catalog. You may export by slot or vsn.

CATALOG

The **historian** 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 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 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 SAM-FS.
defaults.conf	Default information.
/etc/fs/samfs/sam_historian	Default historian catalog file.

SEE ALSO

build_cat(1M), **dump_cat**(1M), **export**(1M), **defaults.conf(4), mcf**(4), **robots**(1M), **robottool**(1M)

NAME

ibm3494 − The IBM3494 interface through lmcpd

AVAILABILITY

LSCibm

DESCRIPTION

ibm3494 is the SAM-FS 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 SAM-FS 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**. 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         # From mtlib output
access=private
```

```
category = 5

# The mcf file entries.
#
# IBM 3494 library
#
/etc/fs/samfs/ibm60     60      im      ibm3494e -      /etc/fs/samfs/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. 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. 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 SAM-FS is not running and **sam-init** has been shutdown (see **sam-init**(1M)).

FILES

mcf	The configuration file for SAM-FS.
/etc/ibmatl.conf	Configuration file used by **lcmpd**.
/opt/LSCsamfs/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), **robots**(1M), **robottool**(1M)

NAME

samst − Driver for SCSI media changers, optical drives and non-motion I/O for tape drives.

SYNOPSIS

samst*@target,lun:a*

AVAILABILITY

LSCsamfs

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, and 2048 byte sector sizes for optical media. Tape drives are only supported for non-motion SCSI commands such as inquiry and mode sense. 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 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 SAM-FS is installed, the existing **samst.conf** file is copied to **samst.conf.MMDDYY** for reference and backup purposes.

NAME

ssi_so − The StorageTek (STK) ACSAPI client daemon.

AVAILABILITY

LSCstk

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 (**/etc/fs/samfs/ssi.sh**) is used by the stk daemon (see **stk**(1M)), to start a ssi_so daemon. In general, most sites should not need to change the variables within this script.

SEE ALSO

robots(1M), **stk**(7)

NAME

stk − The StorageTek (STK) interface through ACSAPI

AVAILABILITY

LSCstk

DESCRIPTION

stk is the SAM-FS interface to the STK libraries. This interface utilizes the ACSAPI interface supplied by STK. The LSCstk package installs the libraries and daemons for the client side of the API. For more information on ACSAPI and interfacing the STK libraries, see the documentation supplied with the STK hardware and server side daemons.

CONFIGURATION

It is assumed that the site has the server daemons (CSI and ACSLM) configured and operating with the STK 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/values 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 will be 50004. Please note that if you are running co-hosted ACSLS 5.3 or higher, the default value will not work (try a higher port, 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 STK. 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 STK 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. SAM has defaults for indexes 0-12 which were current at the time of release. Only the indexes that are missing or incorrect need be supplied. The defaults are as follows:

index	type	capacity	
0	3480	210 Mb	215040
1	3490E	800 Mb	819200
2	DD3A	10 Gb	10485760
3	DD3B	25 Gb	26214400
4	DD3C	50 Gb	52428800
5	DD3D	0 Gb	0
6	DLTIII	10 Gb	10485760
7	DLTIV	20 Gb	20971520
8	DLTIIIXT	15 Gb	15728640
9	STK1R	20 Gb	20971520
10	STK1U	0 Gb	0
11	EECART	1.6 Gb	1677721
12	JCART	0 Gb	0

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 STK 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.

acs is the ACS number for this drive as configured in the STK library.

lsm is the LSM number for this drive as configured in the STK library.

panel is the PANEL number for this drive as configured in the STK library.

drive is the DRIVE number for this drive as configured in the STK library.

EXAMPLE

Here is a sample parameters file and mcf entries for a STK library.

```
#
# This is file: /etc/fs/samfs/stk50
#
hostname = acsls_server_name
portnum = 50004
access = some_user  # No white space allowed in the used_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)         #a comment
capacity = (0=215040, 1=819200, 5=10485760)

The mcf file entries, which reference this configuration file are:

#
# Sample mcf file entries for a STK library
#
/etc/fs/samfs/stk50  50  sk  sk50   -   /etc/fs/samfs/sk50_cat
/dev/rmt/0cbn        51  st  sk50   -
/dev/rmt/1cbn        52  st  sk50   -
```

IMPORT/EXPORT

Since the physical adding and removing of media in the STK 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 volser 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

There are two utilities used to maintain the library catalog used by the STK. **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. Use **import** to populate the catalog with the volumes allowed by ACSLM. 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 SAM-FS is not running and **sam-init** has been shut-down (see **sam-init**(1M)).

FILES

mcf	The configuration file for SAM-FS.
/etc/fs/samfs/ssi.sh	A shell script used to start **ssi_so**.
/etc/fs/samfs/ssi_so	A shared object version of the SSI daemon supplied by STK.
/opt/LSCsamfs/lib/stk/*	The libraries needed by the API interface supplied by STK.
/etc/fs/samfs/stk_helper	A program to issue commands for the STK ACSAPI

SEE ALSO

build_cat(1M), **dump_cat**(1M), **export**(1M), **import**(1M), **robots**(1M), **robottool**(1M), **mcf**(4), **ssi_so**(7)