Sun QFS and Sun Storage Archive Manager 5.3 Reference Manual

Part No: E22976
June 2012

120621@25097

Contents

Preface

Sun QFS and Sun Storage Archive Manager Reference Manual provides man pages relating to the Sun QFS and Sun Storage Archive Manager 5.3 release.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Description	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`**
		`Password:`

aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX system prompt and superuser prompt for shells that are included in the Oracle Solaris OS. Note that the default system prompt that is displayed in command examples varies, depending on the Oracle Solaris release.

TABLE P–2 Shell Prompts

Shell	Prompt
Bash shell, Korn shell, and Bourne shell	`$`
Bash shell, Korn shell, and Bourne shell for superuser	`#`
C shell	`machine_name%`
C shell for superuser	`machine_name#`

CHAPTER 1

User Commands (Man Pages Section 1)

This chapter provides section 1 man pages for Sun QFS and Sun Storage Archive Manger.

alterfile(1)

```
NAME
     alterfile - Alters file content

SYNOPSIS
     alterfile [-o offset] [-v v_value] [-x x_value] file ...

AVAILABILITY
     SUNWsamtp

DESCRIPTION
     The alterfile command changes one byte of a file.  More than
     one file can be specified as input.

OPTIONS
     This command accepts the following arguments:

     -o offset Alters the byte at this offset in the file.  If
               not specified, the offset is a random number
               within the file.

     -v v_value
               Changes the byte to v_value.  If not specified,
               the byte is set to 0.

     -x x_value
               Changes the byte by exclusive OR-ing the byte with
               x_value.

     file ...  Names one or more files to be changed.
```

archive(1)

```
NAME
     archive - Sets archive attributes and schedules files for
     immediate archiving

SYNOPSIS
     archive [-C] [-I] [-d] [-f] [-n] [-w] [-W] [-c copy_no]
     filename ...

     archive [-C] [-I] [-d] [-f] [-n] [-w] [-W] [-c copy_no] -r
     dirname ...[filename ...]

AVAILABILITY
     SUNWsamfs

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

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

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

OPTIONS
     This command accepts the following arguments:

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

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

     -I       Support inconsistent archive copies. This means
              that an archive copy can be created even if the
              file is modified while it is being copied to the
              media. By default, the archive copy is disallowed
              if the file is inconsistent, that is, if the file
              is modified while it was being copied to the
              media.  Note, the file cannot be staged if the
              copy is marked inconsistent; however, after a

              samfsrestore, the inconsistent flag is removed
              from the archive copy and the file can be staged.
```

```
                 Inconsistent archiving is useful for databases,
                 however caution is advised because it a file can
                 be staged from an inconsistent copy after the file
                 is restored using samfsrestore.

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

       -f        Suppresses error messages.

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

                 This option cannot be specified for a file that
                 has the checksum use attribute set.  This
                 attribute is set by using the ssum(1) command -u
                 option.  For more information on ssum(1), see the
                 ssum(1) man page.

                 If the archiver file system examination method has
                 been set to scandirs, setting this option on a
                 directory will prevent the archiver from examining
                 the directory, and all of its subdirectories.
                 This behavior should only be used for directory
                 trees that have all archive copies made for all
                 files.  And, no changes should be made to any of
                 the subdirectories or files.

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

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

                   archive -r .
                   archive -r -w .

       -W        Waits for a file to have all its required archive
                 copies before completing.  This option cannot be
                 specified on the command line in conjunction with
                 the -w, -d, or -n options.  Note that it may take
                 a long time for a file to be archived.

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

                   archive -r .
                   archive -r -W .

       -c copy_no
                 Specify 1, 2, 3, or 4 for copy_no.  If one or more
                 -c options are specified, only those archive
                 copies (copies 1, 2, 3, or 4) are affected.  The
                 -c option may only be used with the -w and -r
                 options.

                 If used without any other options (or just the -r
                 option), archive copy copy_no will be made
                 immediately.

                 If used with the -w option, (with or without the
                 -r option), the command will wait for the archive
                 copy copy_no to be made.

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

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

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

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

       archive -d -r .

  SEE ALSO
       ssum(1), stage(1), release(1)
```

dvt(1)

```
NAME
     dvt - sequentially write and read a file.

SYNOPSIS
     dvt  [-c  block_count]  [-C]  [-d]  [-e   error_limit]   [-g
     stripe_group]   [-o   rw|w|r]   [-O  byte_offset]  [-p]  [-P
     a|f|o|0(zero)]  [-q  queue_size]   [-R   read_threads]   [-s
     block_size] [-v] [-W write_threads] filename

AVAILABILITY
     SUNWsamtp

DESCRIPTION
     dvt writes to a disk file and then reads the disk file.  The
     time  required for the transfer(s) is measured, and the read
     and write transfer rates are computed.  The read  optionally
     verifies the written data.

OPTIONS
     -c  block_count
          The number of blocks to be written/read.

     -C   Specifies that I/O should be buffered through the  page
          cache.

     -d   Specifies direct I/O:  I/O  should  not  use  the  page
          cache.

     -e  error_limit
          Specifies  the  maximum  number  of  data   miscompares
          allowed to occur before the test is stopped.

     -g  stripe_group
          Specifies the number of the  striped  group  where  the
          file  is  to be preallocated.  Stripe_group is a number
          0..127 corresponding to a set of gXXX  devices  in  the
          SAM-QFS master configuration file.  This option applies
          only to filesystems configured with stripe groups.

     -i  stride
          Set the I/O offset stride to stride.

     -o  rw|w|r
          The option rw means write the file and  then  read  the
          file.   This  is the default.  The option w means write
          the file.  The option r means  read  an  existing  file
          written by dvt.

     -O  byte_offset
          Offset initial file I/O by byte_offset bytes.

     -p   Specifies the file should be preallocated.  This option

          applies only to SAM-QFS filesystems.

     -P  a|f|o|0
```

The data pattern. a is an ascending pattern which uses the 64 bit byte offset as the pattern. f is an ascending pattern which uses the 64 bit byte offset as the pattern, plus inserts 16 characters of the filename at 8K boundaries. o writes all 1's. 0 (the numeral zero) writes all 0's.

-q queue_size
Specifies the number of the entries that will be outstanding in the work queue. The queue_size should be larger than read_threads.

-R read_threads
Specifies the number of the threads that will be outstanding for read. If read_threads is less than write_threads, write_threads will be used.

-s size
The block size in bytes. If the size has the suffix k, the block size is in units of kilobytes. The size must be at least 1064 bytes to hold the parameter block header.

-v Data will be verified on the read pass. The times and transfer rates will include this comparison time.

-W write_threads
Specifies the number of the threads that will be outstanding for write.

SEE ALSO

pdvt(1)

sam_advise(3), sam_setfa(3)

mcf(4)

genfile(1)

NAME

genfile - generate files of random data

SYNOPSIS

genfile [-D] [-R] [-S seed] [-c] [-d dirname] [-f] [-g] [-s minsize[-maxsize]] [-v] filename...

AVAILABILITY

SUNWsamtp

DESCRIPTION

genfile generates and checks files of random data. The files consist of records of 2113 random integers. The record also includes the name of the file and the record number. In addition, a file header is written before the data records. The header contains the file name, the status of the file (stat(2)), the random number seed and the data

```
     length.  This format allows the  file  data  to  be  checked
     later knowing only the file name.

     The file is written using  a  buffer  size  of  41  records.
     Using 41 and 2113 (which are prime numbers), avoids perform-
     ing I/O in integer multiples of sector and block sizes.

     File names may be generated by using the regular  expression
     "range"  construction  [x-y].  When such a filename argument
     is used, each range construction, from  right  to  left,  is
     successively incremented.

     For example, the file name file[A-C][0-9] generates the file
     names fileA0 through fileC9.

     Note, you need to "shell escape" the range constructions.

     If one of the files to be  generated  already  exists,  that
     file name is skipped.

OPTIONS
     -D   Set "directio" on the file.

     -R   Allow rewriting of an existing file.

     -Sseed
          Set the random number seed.

     -c   Read and check files.

     -ddirname
          Use dirname as a prefix to the file names.

     -f   Do not report errors.
User Commands                                         genfile(1)

     -g   Generate files.  This is the default action.

     -ssize
          Generate file with data of  length  size.   The  actual
          length  of the file will be size plus the length of the
          file header (168 bytes on SPARC, 156 on  x86)  and  the
          length  of the file name plus one and rounded up to the
          next multiple of four.  For example if size  is  speci-
          fied  as 10 and the file name is /var/file2, the actual
          length of the file would be 190 bytes on a SPARC  plat-
          form and 178 bytes on an x86 platform.

     -sminsize-maxsize
          Generate files with a random data length  between  min-
          size  and maxsize.  The actual file length includes the
          header and file name lengths as above.

     -v   Verbose output.
```

pdvt(1)

```
NAME
     pdvt - POSIX Device Verification Tool

SYNOPSIS
     pdvt  [-b]  [-B]  [-c  block_count]  [-C]  [-d]   [-D]   [-e
     data_error_limit] [-E io_error_limit] [-f] [-g stripe_group]
     [-G sync_file] [-h] [-i stride[k|m|g]] [-k  offset[k|m|g|t]]
     [-l  loops]  [-m  buffer[k|m|g|t]]  [-o  rw|wr|r|w] [-p] [-P
     a|o|0|r]  [-q  queue_size]  [-r]   [-R   read_threads]   [-s
     block_size[k|m]]   [-S  file_size[k|m|g|t]]  [-T]  [-u]  [-v
     log_mask] [-V] [-W write_threads] [-z  seconds]  filename  [
     filename_out ]

AVAILABILITY
     Oracle Corporation

DESCRIPTION
     The POSIX  Device  Verification  Test  ("PDVT")  uses  POSIX
     threads,  or  pthreads,  to test RAID device and file system
     I/O performance.

     PDVT uses the concept  of  thread  pools.   PDVT  (the  boss
     thread) creates a specified number of worker threads.  These
     worker threads survive for the duration of the program.  The
     PDVT  boss  thread  creates  I/O  requests  and  puts  these
     requests  on  a  work  queue.   Worker  threads  remove  I/O
     requests  from  the  work  queue  and  process them.  When a
     worker thread completes an I/O request, it  removes  another
     one from the work queue if the queue is not empty.

OPTIONS
     -b   Perform backwards I/O.

     -B   Print I/O buffer information.

     -c  block_count
          The number of blocks to be written/read.

     -C   Specifies that I/O use the page cache.

     -d   Specifies direct I/O:  I/O  should  not  use  the  page
          cache.

     -D   Live dangerously and allow PDVT to write to slices that
          start at cylinder 0.

     -e  data_error_limit
          Specifies the maximum number of data  miscompares  that
          can occur before the test is stopped.

     -E  io_error_limit
          Specifies the maximum number of  I/O  errors  that  can

          occur before the test is stopped.

     -f   Set read process to follow an active writer.
```

```
-g  stripe_group
     Specifies the number of the  striped  group  where  the
     file  is  to be preallocated.  Stripe_group is a number
     0..n where n matches the gXXX devices configured in the
     SAM-QFS master configuration file.  This option is only
     processed if -p is specified and  it  only  applies  to
     SAM-QFS filesystems with configured stripe groups.

-G sync_file
     Threads  will  wait  to  perform  I/O  until  sync_file
     exists.   Once  test  is complete the sync_file will be
     removed.

-h   Print usage information.

-i  stride[k|m|g]
     Number of bytes to stride

-k offset[k|m|g|t]
     Byte offset to start writing or reading from.

-l  loops
     Number of times to loop before closing the file.   This
     option is useful for testing cache performance.

-m bufsize[k|m|g|t]
     Buffer size for random data pattern.  Pattern is  based
     on  file  name and can therefore regenerate the data if
     using the same file name.  Used in combination with P r
     option.

-o  rw|wr|r|w
     Operation options:

     rw - Read and write a file at the same time

     wr - Write then read a file

     r  - Read a file

     w  - Write a file

-p   Specifies the file should be preallocated.  Striping is
     also  permitted  if striped groups have been configured
     as part of the filesystem.  This option only applies to
     SAM-QFS filesystems.

-P  a|o|0|r

     The data pattern:

     a - ascending pattern based on file offset

     o - ones pattern

     0 - zeros pattern

     r - random pattern (requires -m option)
     -q  queue_size
          Specifies the number of the entries that will  be  out-
          standing  in  the work queue.  The queue_size should be
          larger than read_threads.

     -r   Perform random I/O

     -R  read_threads
          Specifies the number of the threads that will  be  out-
          standing  for  read.   If  read_threads  is  less  than
          write_threads, write_threads will be used.

     -s  block_size[k|m]
          Size of blocks to be written/read.

     -S  file_size[k|m|g|t]
          Size of file to be written/read.  File size  and  block
          count are mutually exclusive.

     -T   Start timing at memory allocation instead of I/O start.

     -u   Unlink file when finished.

     -v log_mask
          Set logging mask to print info, errors, results  and/or
          debugging   information.   Interpreted   in  octal  by
          default, and has a default  value  of  03  (print  test
          results and error messages).

          001 - Print test results (on by default)

          002 - Print error messages (on by default)

          004 - Print debugging information during test

          010 - Print lots of debugging information during test

          020 - Print read and/or write return values

     -V   Data will be verified on the read pass.  The times will
          include this comparison time.

     -W  write_threads
          Specifies the number of the threads that will  be  out-
          standing for write.

     -z  seconds
          Used in combination with f option to specify number  of
          seconds  to  sleep  before  retrying  a  read  request.
          Default is 50000000 nanoseconds.

EXAMPLES
     Example 1: Write a file using 4 write threads and  8  queued
     requests

     pdvt -o w -W 4 -q 8 -s 16m -S 4g /path/file.dat

     Example 2: Write a file with a random buffer for data verif-
```

ication

```
pdvt -P r -m 4g -o w -W 4 -q 8 -s 16m -S 4g /path/file.dat
```

Example 3: Reading previous file with data verification

```
pdvt -P  r  -m  4g  -V  -o  r  -R  4  -q  8  -s  16m  -S  4g
/path/file.dat
```

NOTES

The output of PDVT will vary based on the log level selected. Test results and errors will be displayed by default, but setting the log level to something like 7 will print the test results as well as the test configuration.

Output for results and information are comma separated values that can be imported into a spreadsheet.

Results: operation,block size(KB),fsize(KB),write threads,read threads,queue size, wall time,user time,system time,throughput(MB/sec)

Using the log level 10 will result in a significant amount of output showing what PDVT is doing. Additionally, log level 4 will print out each buffer as it is compared with a data verification read.

The random data feature uses the file name provided on the command line to produce a seed that is fed to the srand48(3C) function call. In order to verify a file once it is written, the file name must be the same as given on the write command.

Performance of a verification read should not be taken into consideration as the throughput represents the time not only to read a file, but also to verify each buffer. The

verification process can add many seconds to a read.

SEE ALSO

dvt(1)

sam_advise(3), sam_setfa(3)

mcf(4)

release(1)

NAME

release - Releases disk space and sets release attributes

SYNOPSIS

```
release [-a] [-d] [-f] [-n] [-p] [-s partial_size] [-V]
filename ...

release [-a] [-d] [-f] [-n] [-p] [-s partial_size] [-V]
```

```
     -r dirname ...  [filename ...]

AVAILABILITY
     SUNWsamfs

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

     Files that are symbolic links are silently ignored.

     If the -a, -d, -n, -p, or -s options are specified, only the
     attribute is set; the disk space is not released, however if
     the partial attribute is reset, all blocks are released for
     an offline regular file.

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

OPTIONS
     This command accepts the following arguments:

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

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

               If the partial attribute is reset, all blocks are
               released for an offline regular file. If the
               partial blocks are to be retained, specify the -p
               or -s option with the -d option.

     -f        Suppresses error messages.

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

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

               By default, the minimum size of the portion
               retained on disk is controlled by the
```

```
                    -o partial=nk option on the mount_samfs(1M)
                    command.  This amount can be adjusted by using the
                    -s option on the release command.

                    NOTE: Even through only a portion of the file is
                    retained on disk, the amount of disk space
                    consumed is equal to one DAU. So, for example, if
                    the partial size is set to 16K and the DAU size is
                    256K, even though only 16K of data remains after a
                    partial release, the actual disk space used is
                    256K.

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

                    This option cannot be specified under the
                    following circumstances:

                    o This option cannot be specified for a file that
                      has the checksum use attribute set.  This
                      attribute is set by using the ssum(1) command -u
                      option.

                    o The stage(1) command's -n option enables the
                      never-stage attribute.

                      For more information on the stage(1) command,
                      see the stage(1) man page.

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

                    that disk space are retained.

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

                    This option cannot be specified under the
                    following circumstances:

                    o This option cannot be specified for a file that
                      has either the checksum-generate or checksum-use
                      attributes set.  These attributes are set by
                      using the ssum(1) command's -g or -u options,
                      respectively.

                    o The stage(1) command's -n option enables the
                      never-stage attribute.
```

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

               Symbolic links that are encountered when this
               option is in effect are not traversed.

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

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

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

SEE ALSO
     archive(1), ssum(1), stage(1).

     mount_samfs(1M).
```

request(1)

```
NAME
     request - Creates a removable-media file

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

     request [-f file_id] [-g group] [-i information]
     [-l vsnfile] -m media [-N] [-n version] [-o owner] [-s size]
     file

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The request command creates a removable-media file, which is
     a file that resides only on one or more removable media
     cartridges.  Such a file does not reside in online magnetic
     disk storage.  Removable media files allow you to read data
     from tape or magneto-optical cartridges directly to memory.
     Creating removable media files allows you to use cartridges
     in an automated library without having them under the
     control of SAM-QFS. In addition, removable media files can
```

also be used for disaster recovery purposes.

A removable media file can be written to more than one volume if the file is large. This creates a volume overflow file.

The -m media option to this command specifies the media type and is a required option.

The Volume Serial Name (VSN) for the removable media file specifies the cartridges to which the removable media file will be written. The VSNs can be specified in one of two ways:

o By specifying the -v vsn option. If you specify the VSN using this option, you can also use the -p option to supply the position of the removable media file on the media. You must be superuser to specify the -p option.

o By specifying the -l vsnfile option.

Note that you cannot specify both the -v option and the -l option on the same request command line.

When an application writes to a removable media file by using the open(2) system call with the oflag argument set to O_WRONLY, O_RDWR, O_CREAT, or O_TRUNC, the SAM-QFS file

system is updated to reflect the data's position on the cartridge. Subsequent read access using the open(2) call with oflag set to O_RDONLY results in a read of the data written during creation.

OPTIONS

This command accepts several options. In the following list, the options are grouped according to function.

General Options

The following general options can be used for any type of removable-media cartridge:

-l vsnfile
    Specifies the name of the file that contains the list of VSNs.

    Within a vsnfile, VSNs must be specified one per line. Each line must contain a VSN name. The vsn cannot be more than 6 characters in length for a tape or 31 characters in length for magneto-optical media.

    You can also specify the position within the vsnfile. If specifying the position, begin each line with the VSN name, followed by a space character, and followed by a decimal or hexadecimal number that indicates the position on the medium. If specifying in hexadecimal, precede the position indicator by Ox characters.

```
                    Each VSN in the vsnfile must reside in a local
                    automated library.

                    This option cannot be specified in conjunction
                    with the -v option.

          -m media  Specifies the media type.  For media, specify a
                    media type as described on the mcf(4) man page.
                    This is a required option.

          -p position1[/position2/position3/ ...]
                    A number that specifies the position of the
                    removable media file on the cartridge.  This
                    option must be specified in conjunction with the
                    -v option.  The number of positions specified must
                    match the number of vsns specified on the -v
                    option.

                    The position can be specified in decimal or
                    hexadecimal.  To specify hexadecimal, precede the
                    position with 0x.  If specified, the media is

                    positioned to position on each VSN.  The number of
                    positions specified must match the number of VSNs.

                    Note that SAM-QFS utilities usually print the
                    position of the file on the medium in hexadecimal.
                    You must be superuser to specify a position.

          -s size   Specifies 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.

          -v vsn1[/vsn2/vsn3/ ...]
                    Specifies one or more VSNs to which the removable
                    media will be written.  The vsn cannot be more
                    than 6 characters in length for a tape or 31
                    characters in length for magneto-optical media.

                    If more than one VSN is specified, separate them
                    with slash characters (/).

                    If you want to specify the position on the media,
                    use the -p position argument in conjunction with
                    this argument.

                    Each vsn specified must reside in a local
                    automated library.

                    This option cannot be specified in conjunction
                    with the -l option.

          file      Specifies the name of the file to be written to
                    removable media.  This can be a full path name.
                    The file must reside in a SAM-QFS file system.
                    After the removable-media file is created,
                    subsequent access to the file results in access to
                    the specified removable-media cartridge.
```

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

     The following option can be used only if the removable media
     file is being written to tape media:

     -N   Specifies that the media is a foreign tape.  That is,
          the tape was not written in a SAM-QFS environment.  The
          tape must be barcoded, write protected, opened for read
          access only, and positioned to 0.

Magneto-Optical Media Options
     The following options can be used only if the removable
     media file is being written to magneto-optical media:

     -f file_id
               Specifies 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 in which file is greater
               than 31 characters, no default exists, and the -f
               argument is required.

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

     -o owner  Specifies the owner.  Can be up to 31 characters.
               The default is the current user.  For
               magneto-optical disk files that are to be used to
               read archive images, the owner specification must
               be -o sam_archive.

     -g group  Specifies the group identifier.  Can be up to 31
               characters.  The default is the user’s current
               group.  For magneto-optical disk files that are to
               be used to read archive images, the group
               specification must be -g sam_archive.

     -i information
               Specified a user information string.  The
               information string is written in the file’s label
               at creation time.  Can be up to 159 characters.

EXAMPLES
     Example 1.  This example command is used to recover data
     from a tape-resident archive file at position 286
     hexadecimal on DLT volume YYY:

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

Example 2. This example command shows how to specify multiple VSNs:

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

Example 3. This example has the same effect as the command line in Example 2, but it uses the -l option:

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

File vsns is as follows:

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

SEE ALSO

basename(1).

open(2).

mcf(4).

NOTES

Removable-media files are not supported over NFS.

schproj(1)

NAME

schproj - change file project attribute

SYNOPSIS

```
schproj [ -fhR ] project filename...

schproj -R [ -H | -L | -P ] project filename...
```

AVAILABILITY

SUNWsamfs

SUNWqfs

DESCRIPTION

schproj sets the project attribute of files and directories. The project can be specified as either the project name or the numeric value from the project database /etc/project. The owner of a file or directory can change the project of a file or directory to any project of which the owner could be a member. The superuser can change the project of any file or directory to any project name or numeric value from the project database /etc/project.

OPTIONS

-f Force. Do not report errors.

-h If the file is a symbolic link, this option changes the project of the symbolic link. Without this option, the

```
               project of the file referenced by the symbolic link is
               changed.

          -H   If the file specified on the command line is a sym-
               bolic  link  that references a directory, this option
               changes  the  project  of  the  directory referenced
               by  the symbolic link and all the files in the file
               hierarchy below it. If a symbolic  link is  encountered
               when  traversing the file hierarchy, the project of the
               target file  is  changed,  but  no recursion takes
               place.

          -L   If the file is a symbolic link, this option changes the
               project of the file referenced by the symbolic link. If
               the file specified on the command line, or encountered
               during the traversal of the file hierarchy, is a sym-
               bolic link that references a directory, then this
               option changes the project of the directory referenced
               by  the  symbolic  link and all files in the file
               hierarchy below it.

          -P   If the file specified on the command line or encoun-
               tered during the traversal of a file hierarchy is a
               symbolic link, this option changes the project of the

               symbolic link. This option does not follow the symbolic
               link to any other part of the file hierarchy.

          Specifying more than one of the  mutually-exclusive  options
          -H,  -L,  or  -P is not considered an error. The last option
          specified determines the behavior of schproj.

          -R   Recursive.  schproj descends through the directory and
               any subdirectories, setting the specified project as it
               proceeds.  When a symbolic link is encountered, the
               project of the target file is changed unless the -h  or
               -P option is specified. However no recursion takes
               place unless the -H or -L option is specified.

     SEE ALSO
          sls(1), sfind(1), project(4).
```

sdu(1)

```
     NAME
          sdu - Summarizes disk usage

     SYNOPSIS
          sdu [-a] [-b] [-c] [-D] [-h] [--help] [-k] [-l] [-L] [-m]
          [-s] [--si] [-S] [--version] [-x] [file ...]

     DESCRIPTION
          This man page describes the GNU version of the du(1) command
          as enhanced by Oracle Corporation for the SAM-QFS file system.
          The sdu command displays the amount of disk space used by
          each file argument.  If file is a directory, the command
          returns disk space information for each subdirectory of
          file.  If file is a removable media file, the command
          returns 0 for the size of that file.

          By default, the space is returned in 1K blocks, but if the
          POSIXLY_CORRECT environment variable is set, 512-byte blocks
          are reported.  The sdu command displays actual disk blocks
          for online SAM-QFS files.  It also displays an estimate of
          disk blocks (based on file size) for offline SAM-QFS files.
          To get actual disk block usage for both online and offline
          files, use the du(1) command.

     OPTIONS
          This command accepts the following options:

          -a        Displays counts for all files, not just
                    directories.  Equivalent to specifying --all.

          -b        Displays sizes in bytes.  Equivalent to specifying
                    --bytes.

          -c        Writes a grand total of all of the arguments after
                    all arguments have been processed.  This can be
                    used to determine the disk usage of a directory
                    with some files excluded.  Equivalent to
                    specifying --total.

          -D        Dereferences symbolic links that are command line
                    arguments.  Does not affect other symbolic links.
                    This is helpful for determining the disk usage of
                    directories like /usr/tmp if they are symbolic
                    links.  Equivalent to specifying
                    --dereference-args.

          -h        Displays sizes in human-readable format.  For
                    example, 1K, 234M, 2G.  Equivalent to specifying
                    --human-readable.

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

          -k        Displays sizes in kilobytes.  This overrides the
                    environment variable POSIXLY_CORRECT.  Equivalent
                    to specifying --kilobytes.

          -l        Counts the size of all files, even if they have
                    appeared already in another hard link.  Equivalent
                    to specifying --count-links.

          -L        Dereferences symbolic links.  That is, the command
                    shows the disk space used by the file or directory
                    that the link points to instead of the space used
                    by the link.  Equivalent to specifying
                    --dereference.

          -m        Uses 1024-kilobyte blocks, not 512, regardless of
                    the POSIXLY_CORRECT environment variable setting.
                    Equivalent to specifying --megabytes.
          -s        Displays only a total for each argument.
                    Equivalent to specifying --summarize.

          --si      Like -h, but size is displayed in base 10 units.

          -S        Counts the size of each directory separately, not
                    including the sizes of subdirectories.  Equivalent
                    to specifying --separate-dirs.

          --version Displays version information on standard output
                    then exits successfully.

          -x        Skips directories that are on different file
                    systems from the one that the file being processed
                    is on.  Equivalent to specifying
                    --one-file-system.

          file      Specifies the file or the path to the file being
                    analyzed.  The size is written.  If no file is
                    specified, the current directory is used.  If more
                    than one file is specified, separate each with a
                    space character.
```

segment(1)

```
NAME
     segment - Sets segment file attributes

SYNOPSIS
     segment [-d] [-f] [-s stage_ahead] [-V] -l segment_size
     filename ...

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

AVAILABILITY
     SUNWsamfs

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

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

     The segment command is not supported in SAM-QFS shared file
     systems.

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

     -f        Suppresses errors.

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

               -l 1024k

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

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

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

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

NOTES

The file system disables quotas at mount time if any of the following files in the file system's root directory are segmented:

- .quota_a

- .quota_g

- .quota_u

The -drives directive in the archiver.cmd file specifies the number of drives to use for archiving and staging.

The mmap function cannot be carried out on a segmented file. Because of this, a segmented file cannot be an executable binary.

Segmentation of files is not supported on a SAM-QFS shared file system.

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

     o  More than one drive is available.

     o  The -drives directive is in effect.

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

     o  The file was archived as striped.

     o  More than one drive is available.

     o  The -drives directive is in effect.

SEE ALSO
     stage(1), archive(1), archiver.cmd(4)
```

setfa(1)

```
NAME
     setfa - Sets file attributes

SYNOPSIS
     setfa [-A allocahead[k|m|g]] [-B] [-d] [-D] [-f]
     [-g stripe_group] [-l length[k|m|g]] [-L length[k|m|g]]
     [-q] [-r dirname] [-s stripe] [-V] filename ...

AVAILABILITY
     SUNWsamfs

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

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

OPTIONS
     This command accepts the following options:

     -A allocahead
               Specifies the number of bytes to be allocated
               ahead of a write to the file.  The n must be an
               integer and must be greater than or equal to one
               kilobyte and less than 4 terabytes.  The n is
               rounded down to units of kilobytes.  The integer
               specified may be followed by k (for kilobytes), m
               (for megabytes), or g (for gigabytes).  For
               example:

               -A 1m
```

```
                   This option is only valid for a regular file. This
                   option should be used when writing large files
                   where more sequential allocation is desired. Note,
                   when the file is closed the blocks are reset to
                   the size of the file.

          -B       Permanently clears the direct I/O attribute for
                   this file.  This means that data is transferred
                   indirectly between the user’s buffer and disk
                   through the system’s buffer cache.

                   For more information, see the directio(3C) man
                   page.  The SAM-QFS buffered I/O attribute is
                   persistent, remaining until the attribute is reset
                   or the file is destroyed.

          -d       Resets all file attributes to the default
                   attributes.  When -d is specified, attributes are

                   first reset to the default, then other
                   attribute-setting options are processed.

          -D       Permanently sets the direct I/O attribute for this
                   file.  This means that data is transferred
                   directly between the user’s buffer and disk.  This
                   attribute should only be set for large,
                   block-aligned, sequential I/O.  The default I/O
                   mode is buffered (uses the page cache).  Direct
                   I/O is not used if the file is currently memory
                   mapped.

                   For more information, see the directio(3C) man
                   page.  The SAM-QFS direct I/O attribute is
                   persistent, remaining until the attribute is reset
                   or the file is destroyed.

          -f       Supresses errors.

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

                   Note that the stripe_group attribute is inherited.
                   It is possible to create a directory and set a
                   stripe group for that directory.  Then, all files
                   created in that directory are allocated on the
                   specified stripe group.  In the following example,
                   files created in audio are allocated on striped
                   group 0, and files created in video are allocated
                   on stripe group 1:

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

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

-L length Specifies the number of bytes to be preallocated
          to the file.  The n must be an integer.  The

          integer specified may be followed by k (for
          kilobytes), m (for megabytes), or g (for
          gigabytes).  For example:

          -L 1g

          This option is only valid for a regular file.  The
          L option allocates using standard allocation. This
          means striping is supported. This also means the
          file can be extended.  The L and l options are
          mutually exclusive.

-q        Specifies that this file will be linked to the
          pseudo character device driver, samaio, for the
          purpose of issuing async I/O. Note, this option
          also sets Direct I/O and qwrite. Setting this
          option may result in greater performance.  This
          option is not valid when applied against certain
          system files and directories such as lost+found.

-r dirname
          Recursively performs the operation (setting file
          attributes) for any files contained in the
          specified dirname or its subdirectories.  If no
          filename is specified, a -r dirname must be
          specified.

-s stripe Specifies the number of allocation units to be
          allocated before changing to the next unit.  If
          stripe is 1, the file is striped across all units
          with 1 disk allocation unit (DAU) allocated per
          unit.  If  stripe is 0, the file is allocated on
          one unit until that unit has no space.  The
          default stripe is specified when the file system
          is mounted.  For more information, see
          mount_samfs(1M)).

          An Invalid argument message is generated if a
          stripe > 0 is specified and mismatched stripe
          groups exist.  A stripe group is said to be
          mismatched if all striped groups do not have the
          same number of partitions.  Striping across
          mismatched stripe groups is not allowed.

-V        Enables the verbose display.  A message is written
```

```
               for each file on which attributes are set.

     filename  Specifies the file for which attributes are being
               set.  If no -r dirname is specified, a filename
               must be specified.  If -r dirname is specified, a
               filename specification is optional.

SEE ALSO
     archive(1), release(1), ssum(1), stage(1).

     mount_samfs(1M).

     directio(3C).
```

sfind(1)

```
NAME
     sfind - Searches for files in a directory hierarchy

SYNOPSIS
     sfind [path ...]  [expression]

AVAILABILITY
     SUNWqfs

     SUNWsamfs

DESCRIPTION
     The sfind(1) command contains Oracle Corporation extensions to
     the GNU find(1) command.  The extensions support the
     features of files that reside in Sun QFS or SAM-QFS file
     systems.

     The sfind command searches the directory tree rooted at each
     path by evaluating the specified expression from left to
     right, according to the rules of precedence.  The search
     continues until the outcome is known (the left hand side is
     false for and operations, true for or), at which point the
     sfind command moves on to the next file name.  For more
     information on the rules of precedence, see the OPERATORS
     section of this man page.

     The Oracle Corporation extensions to this command include the
     addition of several tests that reference characteristics
     specific to files that reside in a Sun QFS or SAM-QFS file
     system.  These tests are as follows:

     -admin_id n, -any_copy_archive_i, -any_copy_d, -any_copy_r,
     -any_copy_s, -any_copy_u, -any_copy_v, -archdone,
     -archive_C, -archive_d, -archive_I, -archive_n, -archived,
     -archpos n, -archpos1 n, -archpos2 n, -archpos3 n, -archpos4
     n, -copies n, -copy n, -copy_archive_i n, -copy_d n,
     -copy_r n, -copy_s n, -copy_u n, -copy_v n, -damaged,
     -is_setfa_D, -is_setfa_g, -is_setfa_s, -mt media_type,
     -mt1 media_type, -mt2 media_type, -mt3 media_type,
     -mt4 media_type, -offline, -online, -ovfl, -ovfl1, -ovfl2,
     -ovfl3, -ovfl4, -partial_on, -project pname, -release_a,
     -release_d, -release_n, -release_p, -rmedia, -rmin n,
     -rtime n, -sections n, -sections1 n, -sections2 n,
     -sections3 n, -sections4 n, -segment n, -segment_a,
     -segment_i, -segment_s, -segmented, -segments n, -setfa_g n,
     -setfa_s n, -ssum_g, -ssum_u, -ssum_v, -stage_a, -stage_d,
     -stage_n, -verify, -vsn pattern, -vsn1 pattern,
     -vsn2 pattern, -vsn3 pattern, -vsn4 pattern, -xmin n,
     -xtime n.

     For a comprehensive lists of tests, see the TESTS section of
     this man page.

     This command accepts the following options:

     path      Specifies the path to the directory to be
               searched.  If no path is specified, the sfind
               command searches the current directory.  If a path
               is specified, the path must appear on the command
               line to the left of the expression argument.  If
               specifying more than one path, separate each with
               a space character.

     expression
               An expression composed from arguments described in
               the OPTIONS, TESTS, ACTIONS, and OPERATORS
               sections of this man page.  If no expression is
               specified, -print is used.

               The expression must begin with one of the
               following characters:

               - A dash (-)
               - An opening parenthesis (()
               - A closing parenthesis ())
               - A comma (,)
               - An exclamation point (!)

               Any arguments to the left of the preceding
               character list are assumed to be paths to search.
               Any arguments to the right of the preceding
               character list are assumed to be part of the
               expression.

               An expression can be constructed from the
               following:

               o  Options, which affect overall operation rather
                  than the processing of a specific file.
                  Options always return true.  For a list of
                  possible options, see the OPTIONS section.

               o  Tests, which return a true or false value.  For
                  a list of possible tests, see the TESTS
                  section.

               o  Actions, which have side effects.  Actions
                  return a true or false value.  If expression
                  contains no actions other than -prune, the
```

```
                 -print action is performed on all files for
                 which the expression is true.  For a list of
                 possible actions, see the ACTIONS section.

              o  Operators, which separate options, tests, and
                 actions.  For a list of possible operators, see

                 the OPERATORS section.  The -and operator is
                 assumed if no operator is specified.

     OPTIONS
        An expression can contain one or more options.  The options
        always return true.  The available options are as follows:

        option    Action

        -daystart Measures times for -amin, -atime, -cmin, -ctime,
                  -mmin, and -mtime from the beginning of today
                  rather than from 24 hours ago.

        -depth    Processes each directory's contents before the
                  directory itself.

        -follow   Dereferences symbolic links.  Implies the -noleaf
                  option.  For more information, see the -noleaf
                  information that follows in this list.

        -maxdepth levels
                  Descends at most levels levels of directories
                  below the command line arguments.  The levels
                  argument muse be a nonnegative integer.  If you
                  specify -maxdepth 0, the tests and actions are
                  applied to the command line arguments only.  For
                  more information, see the TESTS and ACTIONS
                  sections of this man page.

        -mindepth levels
                  Prevents any tests or actions from being performed
                  at levels less than levels.  The levels argument
                  must be a nonnegative integer.  If you specify
                  -mindepth 1, all files except the command line
                  arguments are processed.  For more information,
                  see the TESTS and ACTIONS sections of this man
                  page.

        -noleaf   Supresses optimization.  When specified, the
                  command does not assume that directories contain 2
                  fewer subdirectories than their hard link count.
                  This option is needed when searching file systems
                  that do not follow the UNIX directory-link
                  convention.  Such file systems include CD-ROM or
                  MS-DOS file systems or AFS volume mount points.

                  Each directory on a typical UNIX file system has
                  at least 2 hard links:  its name and its . entry.
                  If subdirectories are present, each of those has a
                  .. entry linked to that directory.  When the sfind
                  command examines a directory, after it has statted
                  2 fewer subdirectories than the directory's link
```

```
                    count, it assumes that the rest of the entries in
                    the directory are not directories.  That is, the
                    rest of the entries are 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.

          -test_segments
                    For a segmented file, applies sfind tests to each
                    individual data segment and to the index inode.
                    If a sfind test returns true for a data segment or
                    for a segmented file's index inode, sfind writes
                    the file path, a slash, and the segment number.
                    The number zero is written for the index inode's
                    segment number.

                    The following options always automatically enable
                    the -test_segments option: -segment n, -segment_i,
                    -segment_s.

                    If this option is not specified, and the tests are
                    applied to a segmented file, then the tests are
                    applied at the file-level, and they are aggregated
                    over all data segments.

                    This option has no effect on test results when
                    applied to unsegmented files.

                    For more information, see the TESTS section of
                    this man page.

          -version  Writes the sfind command's version number to
                    standard error.

          -ractive  If the WORM feature is active writes the path of
                    files which are retained and the retention period
                    has not expired.

          -rover    If the WORM feature is active writes the path of
                    files whose retention period has expired.

          -rafter   date
                    If the WORM feature is active writes the path of
                    files whose retention period ends after the given
                    date.  The date is specified with traditional
                    format CCYYMMDDHHMM. CC is the century, YY is the
                    year, MM is the month, DD is the day, HH is the
                    hour, and MM is minute(s).

          -rremain time
                    If the WORM feature is active writes the path of
                    files with retention periods with at least <time>

                    left.  The time is a duration specified as a
                    combination of years, days, hours, and minutes
                    given as a string "MyNdOhPm" where M, N, O, P are
                    arbitrary non-negative integers. y, d, h, m
                    represent the number of years, days, hours, and
```

```
                   minute(s) for the search.

          -rlonger <time>
                   If the WORM feature is active writes the path of
                   files with retention periods longer than <time>.
                   The time is a duration specified as a combination
                   of years, days, hours, and minutes given as a
                   string "MyNdOhPm" where M, N, O, P are arbitrary
                   non-negative integers. y, d, h, m represent the
                   number of years, days, hours, and minute(s) for
                   the search.

          -rpermanent
                   If the WORM feature is active writes the path of
                   files whose retention period is permanent.

          -xdev     Prevents the command from descending directories
                   on other file systems.

       TESTS
          An expression can contain one or more tests.  Many tests
          accept a numeric argument, n.  The numeric arguments can be
          specified with a preceding plus sign (+) or minus sign (-),
          as follows:

          n Format  Meaning

          +n        Greater than n.

          -n        Less than n.

           n        Exactly n.

          The available tests are as follows:

          test      Condition

          -admin_id n
                   File has admin id number n.

          -amin n   File was last accessed n minutes ago.

          -anewer file
                   File was last accessed more recently than file was
                   modified.  The -anewer test affects the -follow
                   option only if the -follow option comes before (is
                   to the left of) the -anewer test on the command

                   line.

          -any_copy_archive_i
                   File’s copy is marked to be archived immediately.

                   For a segmented file, if the -test_segments option
                   is not in effect, this test evaluates to true if
                   the segmented file’s index inode’s copy is marked
                   to be archived immediately or one of the file’s
                   data segment’s copy is marked to be archived
                   immediately.
```

```
-any_copy_d
          File has an archive copy that is damaged.

          For a segmented file, if the -test_segments option
          is not in effect, this test evaluates to true if
          the segmented file's index inode has an archive
          copy that is damaged or if at least one of the
          file's data segments has an archive copy that is
          damaged.

-any_copy_r
          File has an archive copy marked for rearchiving by
          the rearch(1M) command or by the recycler.

          For a segmented file, if the -test_segments option
          is not in effect, this test evaluates to true if
          the segmented file's index inode has an archive
          copy marked for rearchiving or if at least one of
          the file's data segments has an archive copy
          marked for rearchiving.

-any_copy_s
          File has an archive copy that is stale.

          For a segmented file, if the -test_segments option
          is not in effect, this test evaluates to true if
          the segmented file's index inode has an archive
          copy that is stale or if at least one of the
          file's data segments has an archive copy that is
          stale.

-any_copy_u
          File has an unarchived copy.

          For a segmented file, if the -test_segments option
          is not in effect, this test evaluates to true if
          the segmented file's index inode has an unarchived
          copy or if at least one of the file's data
          segments has an unarchived copy.

-any_copy_v
          File has an archive copy that is verified.

          For a segmented file, if the -test_segments option
          is not in effect, this test evaluates to true if
          the segmented file's index inode has an archive
          copy that is verified or if all of the file's data
          segments have an archive copy that is verified.

-archdone File has completed archive processing.  The
          archiver has no further work to do on the file at
          this time.  Note that this does not mean that the
          file has been archived.

          For a segmented file, if the -test_segments option
          is not in effect, this test evaluates to true for
          a segmented file if and only if all of the file's
          data segments have completed archive processing.
```

```
                    This test does not evaluate a segmented file’s
                    index inode to see if it has completed archive
                    processing.

          -archive_C
                    File has had the equivalent of archive -C run
                    against it, so the concurrent archiving is
                    enabled.  For more information on the -C option to
                    the archive(1) command, see the archive(1) man
                    page.

          -archive_d
                    File has had the equivalent of archive -d run
                    against it, so the archiver handles it according
                    to system defaults.  For more information on the
                    -d option to the archive(1) command, see the
                    archive(1) man page.

          -archive_I
                    File has had the equivalent of archive -I run
                    against it, so the inconsistent archiving is
                    supported.  For more information on the -I option
                    to the archive(1) command, see the archive(1) man
                    page.

          -archive_n
                    File has had the equivalent of archive -n run
                    against it, so it will never be archived.  For
                    more information on the -n option to the
                    archive(1) command, see the archive(1) man page.

          -archived File is archived.

                    For a segmented file, if the -test_segments option

                    is not in effect, this test evaluates to true if
                    all of the file’s data segments are archived.
                    This test does not evaluate a segmented file’s
                    index inode to see if it has been archived.  The
                    following sfind command finds files on /sam6 whose
                    index inode has been archived:

                    sfind /sam6 -archived -segment_i -print

                    The preceeding sfind command identifies only index
                    inodes that have been archived; it does not yield
                    any information regarding whether a segmented
                    file’s data segments have been archived.

          -archpos n
                    File has at least one archive copy at position n.
                    Note that n may be preceded by + or -, and
                    specified in decimal, or hexadecimal if preceded
                    by "0x".  This position is the position prior to
                    the decimal point in sls output or the archiver
                    log.  If n is a path starting with "d" or "f" it
                    is interpreted as a relative path to a disk
                    archive file.
                    For a segmented file, if the -test_segments option
                    is not in effect, this test evaluates to true if
                    the file's index inode has at least one archive
                    copy at position n or if at least one of the
                    file's data segments has at least one archive copy
                    at position n.

          -archpos1 n
          -archpos2 n
          -archpos3 n
          -archpos4 n
                    File has the indicated copy number (1-4) at
                    position n.

                    For a segmented file, if the -test_segments option
                    is not in effect, this test evaluates to true if
                    the file's index inode has the indicated archive
                    copy at position n or if at least one of the
                    file's data segments has the indicated archive
                    copy at position n.

          -atime n  File was last accessed n*24 hours ago.

          -cmin n   File status was last changed n minutes ago.

          -cnewer file
                    File status was last changed more recently than
                    file was modified.  The -cnewer test is affected

                    by the -follow option only if the -follow option
                    comes before (is to the left of) the -cnewer test
                    on the command line.

          -copies n File has n archive copies.

                    For a segmented file, if the -test_segments option
                    is not in effect, this test evaluates to true if
                    each of the file's data segments have n archive
                    copies.  This test does not evaluate a segmented
                    file's index inode to see if it has n archive
                    copies.

          -copy n   File has an archive copy number n.

                    For a segmented file, if the -test_segments option
                    is not in effect, this test evaluates to true if
                    each of the file's data segments have an archive
                    copy number n.  This test does not evaluate a
                    segmented file's index inode to see if it has an
                    archive copy number n.

          -copy_archive_i n
                    File's copy n is marked to be archived
                    immediately.

                    For a segmented file, if the -test_segments option
                    is not in effect, this test evaluates to true if
                    the segmented file's index inode's copy n is
                    marked to be archived immediately or if at least
```

```
                     one of the file’s data segment’s copy n is marked
                     to be archived immediately.

          -copy_d n  File has an archive copy number n that is damaged.

                     For a segmented file, if the -test_segments option
                     is not in effect, this test evaluates to true if
                     the segmented file’s index inode has an archive
                     copy number n that is damaged or if at least one
                     of the file’s data segments has an archive copy
                     number n that is damaged.

          -copy_r n  File has an archive copy number n marked for
                     rearchiving by the rearch(1M) command or by the
                     recycler.

                     For a segmented file, if the -test_segments option
                     is not in effect, this test evaluates to true if
                     the segmented file’s index inode has an archive
                     copy number n marked for rearchiving or if at
                     least one of the file’s data segments has an
                     archive copy number n marked for rearchiving.

          -copy_s n  File has a stale archive copy number n.

                     For a segmented file, if the -test_segments option
                     is not in effect, this test evaluates to true if
                     the segmented file’s index inode has a stale
                     archive copy number n or if at least one of the
                     file’s data segment has a stale archive copy
                     number n.

          -copy_u n  File’s archive copy number n is unarchived by the
                     unarchive(1M) command.

                     For a segmented file, if the -test_segments option
                     is not in effect, this test evaluates to true if
                     the segmented file’s index inode’s archive copy
                     number n is unarchived or if at least one of the
                     file’s data segment’s archive copy number n is
                     unarchived.

          -copy_v n  File has an archive copy number n that is
                     verified.

                     For a segmented file, if the -test_segments option
                     is not in effect, this test evaluates to true if
                     the segmented file’s index inode has an archive
                     copy number n that is verified or if all of the
                     file’s data segments have an archive copy number n
                     that is verified.

          -ctime n   File 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 file system of type type.  Possible
          file system types differ among the different UNIX
          versions and include, but are not limited to, the
          following:  ufs, 4.2, 4.3, nfs, tmp, mfs, S51K,
          and S52K.  You can use the -printf action with its
          %F argument to obtain the types of your file
          systems.  For more information on -printf, see the
          ACTIONS section.

-gid n    File has n for its numeric group ID.

-group gname
          File belongs to group gname.  A numeric group ID

          is allowed.

-ilname pattern
          Like -lname, but the match is case insensitive.

-iname pattern
          Like -name, but the match is case insensitive.
          For example, a pattern of fo* and F?? both match
          file names Foo, FOO, foo, fOo, and so on.

-inum n   File has inode number n.

          For a segmented file, if the -test_segments option
          is not in effect, this test evaluates to true if
          any of the file’s data segments or its index inode
          have inode number n.

-ipath pattern
          Like -path, but the match is case insensitive.

-iregex pattern
          Like -regex, but the match is case insensitive.

-is_setfa_D
          File has had its directio set using the setfa -D
          command.  For more information on the setfa -D
          command, see the setfa(1) man page.

-is_setfa_g
          File has had its stripe group number set using the
          setfa -g command.  For more information on the
          setfa -g command, see the setfa(1) man page.

-is_setfa_s
          File has had its stripe width set using the
          setfa -s command.  For more information on the
          setfa -s command, see the setfa(1) man page.

-links n  File has n links.

-lname pattern
          File is a symbolic link whose contents match shell
```

```
                    pattern pattern.  The metacharacters do not treat
                    the slash character (/) or the period character
                    (.) 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 the
                    -prune action.  For more information, see the
                    example in the -path test in this list.

          -newer file
                    File was modified more recently than file.  The
                    -follow option affects the -newer test only if the
                    -follow option comes before (is to the left of)
                    the -newer test on the command line.

          -nouser   No user corresponds to the file’s numeric user ID.

          -nogroup  No group corresponds to the file’s numeric group
                    ID.

          -offline  File is offline.

                    For a segmented file, if the -test_segments option
                    is not in effect, this test evaluates to true if
                    the file’s index inode is offline or if all of the
                    file’s data segments are offline.

          -online   File is online.

                    For a segmented file, if the -test_segments option
                    is not in effect, this test evaluates to true if
                    the file’s index inode is online and all of the
                    file’s data segments are online.

          -ovfl     File has at least one archive copy that has
                    sections on more than one VSN; this condition is
                    known as volume overflow.

                    For a segmented file, if the -test_segments option
                    is not in effect, this test evaluates to true if
                    the file's index inode has at least one archive
                    copy that has sections on more than one VSN or if
                    at least one of the file's data segments has an

                    archive copy that has sections on more than one
                    VSN.

          -ovfl1
          -ovfl2
          -ovfl3
          -ovfl4    File has an archive copy that has sections on more
                    than one VSN for the indicated copy number (1-4).

                    For a segmented file, if the -test_segments option
                    is not in effect, this test evaluates to true if
                    the file's index inode has an archive copy that
                    has sections on more than one VSN for the
                    indicated copy number or if at least one of the
                    file's data segments has an archive copy that has
                    sections on more than one VSN for the indicated
                    copy number.

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

          -path pattern
                    File name matches shell pattern pattern.  The
                    metacharacters do not treat the slash (/) or the
                    period (.) specially.  For example, the following
                    line writes an entry for a directory called
                    ./src/misc (if one exists):

                    sfind . -path './sr*sc'

                    To ignore a whole directory tree, use the -prune
                    action rather than checking every file in the
                    tree.  For example, the following command skips
                    the directory src/emacs for all files and
                    directories under it and it writes the names of
                    the other files found:

                    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.

          -project pname
                    File belongs to project pname.  A numeric project
                    ID is allowed.

          -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 having the
                    release(1) command with its -d option run against
                    it, and thus has the default release handling.

          -release_a
                    File has had the equivalent of having the
                    release(1) command with its -a option run against
                    it, and thus will be released immediately after
                    being archived.

          -release_n
                    File has had the equivalent of having the
                    release(1) command with its -n option run against
                    it, and thus will never be released.

          -release_p
                    File has had the equivalent of having the
                    release(1) command with its -p option run against
                    it, and thus will be partially released.

          -rmedia   File is a removable media file.

          -rmin n   File’s residence was changed n minutes ago.

          -rtime n  File’s residence was changed n*24 hours ago.

          -sections n
                    File has at least one archive copy that has
                    sections on n VSNs.

                    For a segmented file, if the -test_segments option
                    is not in effect, this test evaluates to true if
                    the file’s index inode has at least one archive
                    copy that has sections on n VSNs or if at least
                    one of the file’s data segments has an archive
                    copy number n that has sections on n VSNs.

          -sections1 n
          -sections2 n
          -sections3 n

          -sections4 n
                    File has an archive copy that has sections on n
                    VSNs for the indicated copy number (1-4).

                    For a segmented file, if the -test_segments option
                    is not in effect, this test evaluates to true if
                    the file’s index inode has at least one archive
                    copy that has sections on n VSNs for the indicated
                    copy number or if at least one of the file's data
                    segments has an archive copy number n that has
                    sections on n VSNs.

          -segment n
                    Data segment or index inode has segment number n.

                    Index inodes always have segment number 0.  Data
                    segments are numbered sequentially starting with
                    1.

                    This test always causes sfind to run as if the
                    -test_segments option were in effect.

          -segment_a
                    File or directory has had the segment attribute
                    set.

                    If the -test_segments option is also in effect,
                    then this test evaluates to true for index inodes
                    and data segments in addition to files and
                    directories that have had the segment attribute
                    set.

          -segment_i
                    Item is an index inode.

                    This test always causes sfind to run as if the
                    -test_segments option were in effect.

          -segment_s
                    Item is a data segment.

                    This test always causes sfind to run as if the
                    -test_segments option were in effect.

          -segmented
                    Item is a segmented file.

                    If used in conjunction with the -test_segments
                    option, this test evaluates to true for index
                    inodes and data segments.

          -segments n

                    Segmented file has n data segments.

          -setfa_g n
                    File's stripe group was set to n using the command
                    setfa -g n.  For more information on the setfa -g
                    command, see the setfa(1) man page.

          -setfa_s n
                    File's stripe width was set to n using the command
                    setfa -s n.  For more information on the setfa -s
                    command, see the setfa(1) man page.

          -size n[unit]
                    File uses n 512-byte blocks.  To specify another
                    size, use the unit suffix.  The possible unit
                    specifiers are as follows:

                    unit      Meaning

                    b or c    Bytes.

                    k         Kilobytes.

                    m         Megabytes.

                    g         Gigabytes.

                    t         Terabytes.

                    For example, the following specifications are
                    equivalent:

                    -size 3
                    -size 1536b

                    The -size test does not count indirect blocks, but
                    it does count blocks in sparse files that are not
                    actually allocated.

          -ssum_g   File has had the equivalent of the ssum(1) command
                    with its -g option 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 the ssum(1) command
                    with its -u option run against it, and thus will
                    have a checksum value verified (used) when it is
                    staged.

          -ssum_v   File has a valid checksum value.

                    For a segmented file, if the -test_segments option

                    is not in effect, this test evaluates to true if
                    all of the file’s data segments have valid
                    checksum values.  This test does not evaluate a
                    segmented file’s index inode to see if it has a
                    valid checksum value.

          -stage_a  File has had the equivalent of the stage(1)
                    command with its -a option run against it, and
                    thus will have associative staging behavior.

          -stage_d  File has had the equivalent of the stage(1)
                    command with its -d option run against it, and
                    thus will have the default staging behavior.

          -stage_n  File has had the equivalent of the stage(1)
                    command with its -n option 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.  For c, specify one of the
          following:

          Type c    Meaning

          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).

-verify   File has the verify attribute set.  See the

          ssum(1) man page for more information on the
          verify attribute.

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

-vsn1 pattern
-vsn2 pattern
-vsn3 pattern
-vsn4 pattern
          File has an archive copy on a volume with VSN
          matching shell pattern pattern for the indicated
          copy (1-4).

-xmin n   File’s data was created n minutes ago.

-xtime n  File’s data was created n*24 hours ago.

-xtype c  The same as -type unless the file is a symbolic
          link.  For symbolic links, the -xtype test checks
          the type of the file that the -type test does not
          check.  For c values, see the -type test in this
          list.

          For symbolic links, the following occurs:
                 o  If the -follow option has not been specified,
                    the test returns true if the file is a link to
                    a file of type c.

                 o  If the -follow option has been specified, the
                    test returns true if c is l.

     ACTIONS
        An expression can contain one or more actions.  The
        available actions are as follows:

        action    Result

        -exec command ;
                  Executes the specified command.  True if 0 status
                  is returned.  All arguments to the right of the
                  -exec keyword are assumed to be arguments to
                  command until an argument consisting of a
                  semicolon (;) 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 the find(1) command.  Both
                  of these constructions might need to be escaped
                  with a backslash character (\) or quoted to

                  protect them from expansion by the shell.

        -fprint file
                  True.  Writes the full file name to file file.  If
                  file does not exist when sfind is run, it is
                  created.  If file 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.  Similar to the -print0 action, but it
                  writes to file like -fprint.

        -fprintf file format
                  True.  Similar to the -printf action, but it
                  writes to file, using format, like the -fprint
                  action.  For information on possible format
                  option, see the -printf format action.

        -ok command ;
                  Executes the specified command, like the -exec
                  action, but it asks the user first (on the
                  standard input).  If the user response does not
                  start with y or Y, command is not run, and the
                  return value is false.

        -print    True.  Writes the full file name, followed by a
                  newline, to standard output.

        -print0   True.  Writes the full file name, followed by a
                  null character, to standard output.  This allows
```

file names that contain newlines to be interpreted correctly by programs that process the sfind output.

-printf format

True. Writes format to standard output, interpreting both backslash (\) escape and percent character (%) directives. Field widths and precisions can be specified as with the printf(3C) C library function. Unlike the -print action, the -printf action does not add a newline at the end of the string.

Two lists follow. The escapes are listed first, and the directives are listed after the escapes.

Esc	Result
\a	Alarm bell.
\b	Backspace.
\c	Stops printing from this format immediately.
\f	Form feed.
\n	Newline.
\r	Carriage return.
\t	Horizontal tab.
\v	Vertical tab.
\\	A literal backslash (\).

A backslash character (\) followed by any other character is treated as an ordinary character, so both are written.

The directives begin with a percent (%) character followed by another character from the following list. If the % character is followed by a character that is not from this list, the directive is discarded, but the other character is printed. The directives are as follows:

Dir	Meaning
%%	A literal percent sign.
%a	File's last access time in the format returned by the C ctime(3C) function.
%Ak	File's last access time in the format specified by k, which is either an ampersand (@) or a directive for the C strftime(3C) function. The directives specify either the time or date. The possible values for k

```
 follow.  Some of them might not be available
 on all systems, due to differences in the
 strftime(3C) function between systems.

o  An ampersand (@).  The ampersand signifies
   seconds elapsed since Jan. 1, 1970, 00:00
   GMT.

o  A time field.  The time fields are as
   follows:

   k    Meaning

   H    The hour in 00, ..., 23 format.

   I    The hour in 01, ..., 12 format.

   k    The hour in 0, ..., 23 format.

   l    The hour in 1, ..., 12 format.

   M    The minute in 00, ..., 59 format.

   p    Specifies whether the locale's time
        is AM or PM.

   r    The time in a 12-hour format.  This
        results in a hh:mm:ss [A | P]M
        format.

   S    The second in a 00, ..., 61 format.

   T    The time in a 24-hour format.  This
        results in a hh:mm:ss format.

   X    The locale's time representation in
        H:M:S.

   Z    The time zone (for example, EDT) or
        nothing (if no time zone is
        determinable).

o  A date field.  The date fields are as
   follows:

   k    Meaning

   a    The locale's abbreviated weekday name
        in Sun, ..., Sat format.

   A    The locale's full weekday name, in
        Sunday, ..., Saturday format.  This
        is of variable length.

   b, h The locale's abbreviated month name
        in Jan, ..., Dec format.

   B    The locale's full-month name in
        January, ..., December format.  This
```

is of variable length.

c	The locale’s date and time in the following example format: Sat Nov 04 12:02:33 EST 1989.
d	The day of month in 01, ..., 31 format.
D	The date in mm/dd/yy format.
j	The day of year in 001, ..., 366 format.
m	The month in 01, ..., 12 format.
U	The number of the week in the year, with Sunday considered to be the first day of week, in 00, ..., 53 format.
w	The day of week in 0, ..., 6 format.
W	The number of the week in the year, with Monday considered to be the first day of week, in 00, ..., 53 format.
x	The locale’s date representation in mm/dd/yy format.
y	The last two digits of year in 00, ..., 99 format.
Y	The year in the following example format: 2002.

Dir	Meaning
%b	File’s size in 512-byte blocks (rounded up).
%B	File’s start time for the WORM retention period in the format returned by the C ctime(3C) function. A dash (-) is written if the item is not a WORM, or does not reside in a QFS or SAM-QFS file system.
%c	File’s last status change time in the format returned by the C ctime(3C) function.
%Ck	File’s last status change time in the format specified by k, which is the same as for the %Ak directive. For more information, see the %Ak directive previously in this list.
%d	File’s depth in the directory tree. A zero

```
              (0) means that the file is a command line
              argument.

        %e    File's creation time in the format returned
              by the C ctime(3C) function.

              A dash (-) is written if the item does not
              reside in a QFS or SAM-QFS file system.

        %Ek   File's creation time in the format specified
              by k, which is the same as for %Ak.  For
              more information, see the %Ak directive
              previously in this list.

              A dash (-) is written if the item does not
              reside in a QFS or SAM-QFS file system.

        %f    File's name with any leading directories
              removed.

        %F    Type of file system the file is on.  This
              value can be used for the -fstype test.

        %g    File's group name.  This is the 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.

        %j    File's last attribute change time in the
              format returned by the C ctime(3C) function.

              A dash (-) is written if the item does not
              reside in a QFS or SAM-QFS file system.

        %Jk   File's last attribute change time in the
              format specified by k, which is the same as
              for %Ak.  For more information, see the %Ak
              directive previously in this list.

              A dash (-) is written if the item does not
              reside in a QFS or SAM-QFS file system.

        %k    File's size in 1K blocks rounded up.

        %K    Segment number of the data segment or the

              index inode.

              Index inodes always have segment number 0.
              Data segments are numbered sequentially
              starting with 1.
              A dash (-) is written if the item is not an
              index inode and not a data segment.

        %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.

        %Q    Number of data segments that comprise the
              segmented file.

              A dash (-) is written if the item is not a
              segmented file.

        %r    File's stripe group number as it was set
              using the setfa -g command.  A dash (-) is
              written if the file's stripe group number
              was not set using setfa -g or if the file
              does not reside in a Sun QFS file system.
              For more information on the setfa -g
              command, see the setfa(1) man page.

        %R    The WORM retention period for a WORM capable
              directory or WORM file in YYYYy, DDd, HHh,
              MMm format. If the retention period is 0,
              "permanent" is written.

              A dash (-) is written if the item is not a
              WORM, or does not reside in a QFS or SAM-QFS
              file system.

        %s    File's size in bytes.

        %t    File's last modification time in the format
              returned by the C ctime(3C) function.

        %Tk   File's last modification time in the format
              specified by k, which is the same as for

              %Ak.  For more information, see the %Ak
              directive previously in this list.

        %u    File's user name, or numeric user ID if the
              user has no name.

        %U    File's numeric user ID.

        %w    File's stripe width as it was set using the
              setfa -s command.  A dash (-) is written if
              the file's stripe width was not set using
              setfa -s or if the file does not reside in a
```

```
                    Sun QFS or SAM-QFS file system.  For more
                    information on the setfa -s command, see the
                    setfa(1) man page.

              %W    The retention state of the item. If the WORM
                    is capable for a directory, worm-capable is
                    written, or active or over is written for a
                    file.

                    A dash (-) is written if the item is not a
                    WORM, or does not reside in a QFS or SAM-QFS
                    file system.

              %X    File's date the WORM retention period will
                    expire in the format of %c of the C
                    strftime(3C) function. If the retention
                    period is 0 (never expire), "*" is written.

                    A dash (-) is written if the item is not a
                    WORM, or does not reside in a QFS or SAM-QFS
                    file system.

              %y    File's residence time in the format returned
                    by the C ctime(3C) function.

                    A dash (-) is written if the item does not
                    reside in a QFS or SAM-QFS file system.

              %Yk   File's residence time in the format
                    specified by k, which is the same as for
                    %Ak.  For more information, see the %Ak
                    directive previously in this list.

                    A dash (-) is written if the item does not
                    reside in a QFS or SAM-QFS file system.

              %Z    Segment length setting in megabytes.  A dash
                    (-) is written if the item does not have the
                    segment attribute set.

     -prune
          Always yields true.  Does not examine any directories
          or files in the directory structure below the pattern
          just matched.  If -depth is  specified, -prune has no
          effect.

     -ls  True.  Writes information on the current file to
          standard output.  The information written is in ls(1)
          command format with -dils options.  For more
          information on the ls(1) command, see the ls(1) man
          page.

          By default, the block counts in the output are in 1K
          blocks.  If the POSIXLY_CORRECT environment variable is
          set, block counts are in 512-byte blocks.

  OPERATORS
     An expression can contain one or more operators.  The
     following operators are listed in order of decreasing
```

precedence:

operators	Action
(expr)	Forces 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

Example 1. The following command finds all files in the /sam4 directory that are not archived:

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

Example 2. The following command finds all regular files in the current directory that are archived, online, and are nonzero in length:

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

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

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

Alternatively, the following command could be used:

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

Example 4. The following command prints the modification time of all files in /sam6:

```
sfind /sam6 -printf "file %f mod time %Aa %Ab %Ad %AY %AT\n"
file file7 mod time Fri Nov 12 1999 18:44:27
```

Example 5. The following command finds all files on /sam6 that have at least one archive copy that has sections on more than one VSN, i.e. all files on /sam6 that have at

```
     least one archive copy that overflows VSNs.

     sfind /sam6 -ovfl -print

     Example 6.  The following command finds all files on /sam6
     that have at least one archive copy that has sections on
     more than one VSN, but fewer than five VSNs.

     sfind /sam6 -sections +1 -sections -5 -print

     Example 7.  The following command finds all files in /sam6
     whose stripe group was set to a value greater than 3 but
     less than 8 and whose stripe width was set to a value
     greater than 1, but less than 5.  It prints the file's path,
     stripe group number and stripe width value:

     sfind /sam6 -type f -setfa_g +3 -setfa_g -8 -setfa_s +1 -setfa_s -5 \
                                     -printf "Path: %p, g%r, s%w\n"
     Path: /sam6/seismic_scan/030610/1200/scn.dat, g4, s2

     Example 8.  The following command finds all files in /sam6
     which have disk archive copies on disk volume "diskv1" in
     file "d1/d4/d201/f107".

     sfind /sam6 -vsn diskv1 -archpos d1/d4/d201/f107

EXIT STATUS
     The sfind(1) command exits with status of 0 if all files are
     processed successfully.  It exits with a status greater than
     0 if errors occur.

SEE ALSO
     archive(1), find(1), release(1), setfa(1), ssum(1),
     stage(1).

     rearch(1M).

     printf(3C).
```

sls(1)

```
NAME
     sls - Lists directory content

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

AVAILABILITY
     SUNWqfs

     SUNWsamfs

DESCRIPTION
     This man(1) page describes the Oracle Corporation extensions
```

to the GNU version of the ls(1) command. Oracle Corporation modified the ls(1)command and added the following features to support Sun QFS and SAM-QFS software:

o -D, which lists a detailed description of each file.

o -2, which lists two lines of output for each file.

o -K, which lists all segments of a segmented file.

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

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

OPTIONS

The sls(1) command accepts the following options:

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

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

-c   Sorts directory contents according to the file status change times instead of the modification times. If the long listing format is being used, it generates the status change time instead of the modification time.

Equivalent to specifying --time=ctime and --time=status.

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

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

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

-g   Ignored. For UNIX compatibility.

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

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

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

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

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

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

-p   Suffixes each file name with a character that indicates
     the file type.  For directories, the suffix is a slash
     (/).  For symbolic links, the suffix is an at sign (@).

     For FIFOs, the suffix is a pipe symbol (|).  For
     sockets, the suffix is an equal sign (=).  There is no
     suffix for regular files.

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

-r   Sorts directory contents in reverse order.  Equivalent
     to specifying --reverse.

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

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

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

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

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

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

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

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

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

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

     Example:

```
server# sls -D mickey.gif
mickey.gif:
  mode: -rw-r--r--  links:   1  owner: root      group: other
  length:    319279  admin id:      7  inode: 1407.5
  project: system(0)
  offline;  archdone;  stage -n;
  copy 1: ---- May 21 10:29     1e4b1.1    lt DLT001
  access:      May 21 09:25     modification: May 21 09:25
  changed:     May 21 09:26     attributes:   May 21 10:44
  creation:    May 21 09:25     residence:    May 21 10:44
```

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

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

     The third line indicates the file’s project name and
     project ID (see schproj(1)).

     The fourth line shows the file states and attributes.

Possible file states, which are set by the system, are as follows:

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

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

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

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

segments n , offline o , archdone a , damaged d

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

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

output is as follows:

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

The first field indicates the archive copy number.

The second field consists of four dashes, as follows:

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

  Content Meaning

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

  U       The copy has been unarchived.

  -       The archive copy is active and valid.

o Dash 2 indicates the archive status, as follows:

  Content Meaning

  r       The archiver will rearchive this copy.

  -       This archive copy will not be rearchived.

o Dash 3 is unused.

o Dash 4 indicates a damaged, undamaged, or verified
  status, as follows:

  Content Meaning

  D       The archive copy is damaged.  This archive
          copy will not be staged.

  V       The archive copy has been verified. The file
          is flagged for data verification and this
          copy has been verified.

  -       The archive copy is not damaged, and if the
          file is flagged for data verification, this
          copy has not yet been verified. It is a
```

candidate for staging.

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

The fourth field contains two hex numbers separated by a period (.). The first hex number, 3498f, is the

position of the beginning of the archive file on the media. For disk archive copies the first number is an index to the file path (see below). The second hex number is the file byte offset divided by 512 of this copy on the archive file. In this example, 1 means that this is the first file on the archive file because it is offset by 512 bytes, which is the length of the tar(1) header.

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

For media type dk (disk archiving) the volume serial name is the disk volume as defined in diskvols.conf(4), and there is an additional field which is the path to the archived tar file. This path is relative to the pathname for the disk volume as specified in the diskvols.conf file.

For media type cb (Sun StorageTek 5800 Storage System disk archiving) the volume serial name is the disk volume as defined in diskvols.conf(4), and there is an additional field which is the metadata string for the archived tar file.

Various times are displayed for the file as follows:

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

The WORM feature changes the meaning of some of the timing attributes for a file. In addition, information regarding retention duration, state, and period (the latter in YYYYy DDd HHh MMm format) is available. The changes to original time attributes and the retention attributes are as follows:

Time Type	Meaning
modification	Start time for the retention period.
changed	Time the retention period was last changed.
attributes	The date the retention period will expire.
retention	The retention state of the file, active or over.
retention-period	The time supplied when the retention period was set on the file.

Directories are handled differently as retention periods are the default period for files and subdirectories contained in that directory. Unlike files, retention periods on directories can be shortened. Setting the WORM flag on a directory should be a reasonably rare occurance as the WORM feature is inherited from the parent. When the WORM flag is set on a directory only the state is changed to "worm-capable" indicating the directory can contain retained files.

The checksum attributes are displayed on the line as follows.

```
checksum: -g -u -a 1 0xec02591b41dca8aa 0x2cdc5977fdd5bbc4
```

The previous line is displayed for a file with any of the possible checksum attributes set. If -g is set, the file is marked for generating a checksum. If -u is set, the file is marked for verifying the checksum. The -a precedes the numeric algorithm indicator which specifies which algorithm is used when generating the checksum value. If two hex numbers appear, there is a valid checksum and the checksum value is the 2 hex numbers.

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

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

The first line shows the I/O type (always blockio), the media type, number of volumes, and blocksize. The

second and following lines show the section length, position and offset, and VSN for each volume. There will only be one section line except in the case of volume overflow. The blocksize will be zero until the first time the volume is loaded, at which time it will

```
         be filled in with the correct value.

         The -D option is equivalent to specifying
         --format=detailed.

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

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

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

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

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

     -N  Does not quote file names.  Equivalent to specifying
         --literal.

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

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

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

     -T cols
         Assumes that each tab stop is cols columns wide.  The

         default is 8.  Equivalent to specifying --tabsize cols.

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

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

```
-1   Lists one line per file.  Equivalent to specifying
     --format=single-column.

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

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

     o Position 1 - Offline/damaged status

          O   The file is offline.

          P   The file is offline with partial online.

          E   The file is damaged.

          -   The file is online.

     o Position 2-4 - Archiver attributes

          n   Never archive the file.

          a   Archive the file immediately after creation or
              modification (see archive(1) to set).  Ignore
              archive set age times.  This attribute remains
              set until a different archive command is
              issued for the file (see archive(1)).

          r   The file is scheduled to be re-archived on a
              different volume.  This attribute is set by
              the recycler.

          -   The attribute is not set.

     o Position 5-7 - Releaser attributes

          n   Never release the file (only the superuser can
              set this).

          a   Release as soon as 1 copy is archived.

          p   Partially release the file.  The first portion
              is left on disk after release.

          -   The attribute is not set.

     o Position 8-9 - Stage attributes

          n   Direct access to removable media (never stage
              on read).

          a   Associatively stage this file.
          -   The attribute is not set.

     o Position 10 - Not used.  Always a dash (-).

     o Position 11 - Blank space.

     o Position 12-14 - Checksum attributes.  Set by the
     ssum(1) command.

          g   Generate a checksum value when archiving.

          u   Checksum the file when staging.

          v   A valid checksum exists.

          -   The attribute is not set.

     o Position 15-16 - Not used.  Always a dash (-).

     o Position 17 - Blank space.

     o Position 18 - Segment attributes.

          s   The segment attribute is set.

          -   The attribute is not set.

     o Position 19 - Index and segment attributes.
          These attributes do not appear if the segment
          attribute (position 17) is not set.

          S   This is a data segment.

          I   This is an index for a file segment.  Four
              additional numbers contained within braces
              ({}) are written, as follows:  {n, o, a, d}.
              The numbers within the braces indicate the
              following:

              n   The number of data segments in the
                  segmented file.

              o   The number of data segments which are
                  offline.

              a   The number of data segments which are
                  archdone.

              d   The number of data segments which are
                  damaged.

          -   The attribute is not set.

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

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

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

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

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

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

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

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

     /sam1/testdir0/filea:
       mode: -rw-r-----  links:   1  owner: root      group: other

       length:    306581  admin id:      0  inode:    11748.11
       project: system(0)
       copy 1: ---- Oct 31 13:52        15.0    dk disk01
       access:      Oct 31 13:50  modification: Oct 31 13:50
       changed:     Oct 31 13:50  attributes:   Oct 31 13:50
       creation:    Oct 31 13:50  residence:    Oct 31 13:50

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

SEE ALSO
     archive(1), ls(1), release(1), samchaid(1M), schproj(1),
     ssum(1), stage(1), tar(1).
```

squota(1)

```
NAME
     squota - Reports quota information

SYNOPSIS
     squota [-a] [-g] [-h] [-k] [-u] [-O] [file]

AVAILABILITY
     SUNWsamfs
     SUNWqfs

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

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

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

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

OPTIONS
     This command accepts the following options:

     -a        Returns admin set quota statistics.

     -g        Returns group quota statistics.

     -h        Prints a brief usage summary and exits.

     -k        Display all storage units (block quantities) in
               units of 1024-byte blocks.  When specified, all
               block counts are returned in units of 1024-byte
               blocks.

     -u        Returns user quota statistics.

     -O        Returns online statistics only.  The default is to
               return total statistics as well as online
               statistics.

     file      Return the quota information pertaining to file.
               If file is writeable by the user issuing the
               command, information about the applicable user,
               group, and admin set IDs is returned.  If file is
               not writeable by the user issuing the command,
               information about the quotas for the user's GID
               and UID on the filesystem that file resides on is
               returned.

EXAMPLES
```

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

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

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

```
server% squota /qfs1/george
                                    Limits
        Type    ID    In Use     Soft     Hard
/qfs1/george
Files  admin    12         4        0        0
Blocks admin    12      6824        0        0
Grace period                    0s
---> Infinite quotas in effect.
/qfs1/george
Files  group   101         1     1000     1200
Blocks group   101         8    20000    30000
Grace period                    3d
No user quota entry.
```

Example 3. The following example is from a SAM-QFS file system:

```
server% squota /sam1/adams
                                 Online Limits                Total Limits
        Type    ID    In Use     Soft     Hard    In Use     Soft     Hard
/sam1/adams

Files  admin    12         4        0        0         4        0        0
Blocks admin    12      6824        0        0      3950        0        0
Grace period                    0s
---> Infinite quotas in effect.
/sam1/adams
Files  group   101         1     1000     1200         1        0        0
Blocks group   101         8    20000    30000         8   100000   120000
Grace period                    3d
/sam6
Files   user   130        11       15     2000        11       15     6000
Blocks  user   130       320      400   200000       560*     500   700000
Grace period                    0s                           0s
---> Total soft limits under enforcement (since 18h10m1s ago)
```

EXIT STATUS

This command returns the following:

o 0 on successful completion.

o 1 on a usage or argument error.

o 10 on an execution error.

```
FILES
     filesytem/.quota_a  Admin set quota information

     filesystem/.quota_g Group quota information

     filesystem/.quota_u User quota information

SEE ALSO
     samquota(1M)

     samfsck(1M)

     passwd(4) - User ID information

     group(4) - Group ID information

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

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

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

ssum(1)

```
NAME
     ssum - Set file checksum attributes

SYNOPSIS
     ssum [-d] [-e] [-f] [-g] [-u] filename...

     ssum [-d] [-e] [-f] [-g] [-u] -r dirname...[filename...]

AVAILABILITY
     SUNWsamfs

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

     The generate attribute must be set  on  a  file  before  any
     archive  copy  has  been made.  Likewise, the selected algo-
     rithm 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 the checksum use attri-
     bute set. Also, it is not valid to specify that a file never
```

be archived (archive -n) as well as specify that a checksum be used. Therefore, when a direct access, partial release, or archive never attribute is set on a file, attempting to set the checksum generate or use attribute on the file will result in an error and the attributes will be unchanged. Similarly, when either the checksum generate or use attribute is set on a file, attempting to set a direct access, partial release, or archive never attribute on the file will result in an error and the attributes will be unchanged.

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

OPTIONS

-d Return the file's checksum attributes to the default, which turns off checksumming. Using the -d option will not reset the 'checksum valid' flag if a valid checksum has been generated for a file. The -d option deletes

the checksum attributes, if no valid checksum has been generated.

-e Set data verification for the file or directory specified. This forces the generation and use of checksums for archiving and staging, and prevents the release of the file until all archive copies have been created and their checksums verified. A file with only one archive copy will never be released. Only a superuser can set this attribute on a file.

Files created in directories with the -e flag set inherit it.

-f Do not report errors.

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

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

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

SEE ALSO

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

stage(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
     SUNWqfs

     SUNWsamfs

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

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

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

OPTIONS
     -a   Set the associative staging attribute on  the  file  or
          directory.   Associative  staging  is  activated when a
          regular file that has the associative staging attribute
          set  is  staged.   All files in the same directory that
          have the associative staging attribute set are  staged.
          If  a  symbolic link has the associative staging attri-
          bute 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.  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 per-
          formance 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 the checksum use attri-
          bute (ssum -u).  The stage -n  attribute  is  not  sup-
          ported  on a Sun SAM-QFS shared file system client; the
          entire file is staged when accessed  on  a  client.  If
          stage -n is issued while the file is being staged, EIN-
          VAL is returned.

     -p   Specifies  that  the  offline  regular  file’s  partial
          blocks be staged.

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

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

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

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

SEE ALSO
     release(1), archive(1), ssum(1),  mount_samfs(1M),  mmap(2),
     write(2)
```

CHAPTER 2

Maintenance Commands (Man Pages Section 1M)

This chapter provides the section 1M man pages for Sun QFS and Sun Storage Archive Manager.

archive_audit(1M)

```
NAME
     archive_audit - Generate an archive audit

SYNOPSIS
     /opt/SUNWsamfs/sbin/archive_audit [ -f audit_file ] [ -V ] [
     -d ] [ -c archive_copy_number ]... root_path

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     archive_audit generates an audit of all archived  files  and
     removable  media files (excluding archiver and stager remov-
     able media files, and  removable  media  files  created  for
     disaster recovery which have not yet been referenced) in the
     SAM-QFS directory root_path by media type and VSN. The audit
     results are written to the VSN audit file.  An optional sum-
     mary of all archive VSNs is written to standard output.

     Note that archive_audit will  not  be  able  to  distinguish
     removable media files used by the stager daemon in file sys-
     tems which  have  been  created  in  systems  prior  to  Sun
     StorEdge  SAM-FS  4.0  and  upgraded, so these sizes will be
     counted in the totals.  Also, removable media files  created
     by a user for disaster recovery purposes may duplicate space
     on a volume assigned to an archive copy, in which  case  the
     space will be accounted for twice.

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_numbers
               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 out-
               put 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   seg_num   disk_path

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

     seg_num is the segment number of the archived segment of the
     file.   seg_num is 0 if it is a segmented file’s index inode
     or if the entry is a  directory  or  a  non-segmented  file.
     Data  segments of a segmented file are numbered sequentially
     beginning with 1.
     disk_path is the path to the  tar  archive  containing  this
     file  on  the  disk  archive volume.  If the volume is not a
     disk archive, this field is blank.

     The following is an example of the archive_audit line:

     lt DLT000 ---- 1 0 4ffd.9fa5e 169643 /sam5/QT/rainbow.sgi 6

     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  can-
                   didate 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  sec-
     tion)  for  an archive file or the file size for a removable
     media file.

     The eighth field is the name of the archive file  or  remov-
     able media file.

     The ninth field shows the number of the archived file's seg-
     ment.   This  field  is 0 if the archive copy is of the seg-
     mented file's index inode or if the  archived  file  is  not
     segmented.

     The last field is blank since this is a tape archive.  For a
     disk archive it would have a path such as "d3/f198".
```

EXIT STATUS

The following exit values are returned:

```
     0         Audit completed successfully.

     6         Nonfatal: An  issue  encountered  with  rootpath's
               filename or the path.

     7         Nonfatal: Closing  of  a  subdirectory  under  the
               rootpath failed.

     10        Nonfatal: sam_segment_vsn_stat for a file failed.

     11        Nonfatal: sam_vsn_stat for a file failed.

     12        Nonfatal: sam_readrminfo for a file failed.

     13        Nonfatal: idstat for a file failed.

     14        Nonfatal: getdent for a directory failed.

     15        Nonfatal: Invalid segment size for a file  encoun-
               tered.

     30        Fatal: Command line argument errors.

     31        Fatal: Audit file issues were encountered.

     32        Fatal: An issue with the root path or a  subdirec-
               tory was encountered.

     35        Fatal: Malloc errors terminated archive_audit.

SEE ALSO
     sam-archiverd(1M), mcf(4)
```

archive_mark(1M)

```
NAME
     archive_mark - Mark file as archived

SYNOPSIS
     /opt/SUNWsamfs/sbin/archive_mark -cn [ -offset n ]  [  -v  ]
     rm_file file

AVAILABILITY
     SUNWsamtp

DESCRIPTION
     archive_mark marks file as archived. This is done by setting
     the  file's  archive information based on information stored
     in the removable media  file  rm_file  used  to  create  the
     archive image of the file.

OPTIONS
     -cn  Assign archive copy n (where n is a number 1 to  4)  to
          this archive.

     -offset n
```

```
               Sets the file offset to the beginning of the data  por-
               tion  of  this file in the archive image.  The offset n
               is in bytes and the default is 0.

          -v   Selects verbose mode in which a message is  printed  at
               completion identifying the media.

          rm_file
               The name of the removable media file used to create the
               archive image of the file.

          file The name of the file to be marked as archived.

     NOTE
          Files archived to optical disk must be written to  a  remov-
          able  media  file with the archive recorded file name and be
          owned by the archive owner.  The default recorded file  name
          is SAM_ARCHIVE and the owner is sam_archive.

     SEE ALSO
          request(1),
```

archiver(1M)

```
NAME
     archiver - SAM-QFS file archiver command file processor

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

     /opt/SUNWsamfs/sbin/archiver [-A] [-a] [-b] [ -c archive_cmd
     ] [-f] [-l] [ -n file_system ] [-v]

AVAILABILITY
     SUNWsamfs

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

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

Sample default output:
      Reading archiver command file "example1.cmd"

      Notify file: /etc/opt/SUNWsamfs/scripts/archiver.sh

      Archive media:
      media:sg bufsize: 4 archmax:  512.0M Volume overflow not selected
      media:mo bufsize: 4 archmax:    4.8M Volume overflow not selected

      Archive libraries:
      Device:mo20 drives_available:0 archive_drives:1

      Device:tp30 drives_available:0 archive_drives:3

      Archive file selections:
      Filesystem samfs1  interval: 300
        Logfile: /var/opt/SUNWsamfs/archiver.log

      samfs1  Metadata
          copy:1  arch_age:240
      big  path:. minsize: 500.0k
          copy:1  arch_age:30
          copy:2  arch_age:7200
      all  path:.
          copy:1  arch_age:30

      Archive sets:
      allsets
          reserve: set//

      allsets.1
         .reserve: set//

      allsets.2
          archmax: 5G
         .reserve: set//

      allsets.3
         .reserve: set//

      allsets.4
         .reserve: set//

      all.1
         .reserve: set//
        media: mo
       Total space available:    2.1G

      big.1
         .reserve: set//
        media: sg
       Total space available:   77.5G

      big.2
         .archmax: 5G
         .reserve: set//
        media: sg
       Total space available:   77.5G

      samfs1.1
         .reserve: set//
        media: mo
       Total space available:    2.1G

     Archive Set parameters set by the archiver command file  are
     listed  for all Archive Sets.  Parameters defined by allsets
     and allsets.n are preceeded by the '.' character.

OPTIONS
     -A        Turn on all list options except -a and -b.

     -a        List archive detail for files.

               The -a option produces a line of output  for  each
               file  found  in  an  inodes scan of a file system.
               The line lists present and future archive activity
               for  the file.  The line is in a fixed format con-
               sisting of space (' ') separated  fields  as  fol-
               lows:

               1  A single character that identifies the file type:
                   'l' Symbolic link
                   'R' Removable media file
                   'I' Segment index
                   'd' Directory
                   'f' Regular file
                   'b' Block special
                   '?' Other

               2  The name of the file quoted using '"'.
                  The '"' and '\' characters in the file name are represented
                  by '\"' and '\\'.

               3  inode.gen  Inode and generation number

               4  Archive Set name.  If the file is not to be archived, '-'.

               5 - 8  Archive information for the four possible copies.
                  If no archive copy required '-'
                  If archived, 'media.VSN'
                  If not archived, the time at which archiving will begin
                      'yyyy-mm-ddThh:mm:ss' (ISO 8601)
                  If the copy is to be unarchived, the time for unarchiving
                      '/yyyy-mm-ddThh:mm:ss'

               The '-a' option  will  clear  any  previously  set
               option,  except  a  file  system name set by '-n'.
               This allows a user to generate  only  the  archive
               activity  information to standard out.  This could
               be used as input to sort, a spreadsheet  or  data-
               base.

     -b        Print the total size in bytes in base 10 units for
               -v, and the capacity and space for -l. By default,
               base 2 units is used.
     -c archive_cmd
               The name  of  the  archiver  command  file  to  be
               evaluated.                Default               is

               /etc/opt/SUNWsamfs/archiver.cmd.

     -f        List file system content.  Sample output:

                Filesystems:
                qfs1 mount: /qfs1
                Examine: noscan Interval: 2h
                Logfile:/var/opt/SUNWsamfs/archiver/log

                Producing statistics
                File type           Count  Percent   Bytes  Percent                Bytes

                All               411,958  100.00%    8.3G  100.00%           8935481659
                    offline            26     0.1%  264.1M    3.10%            276878242
                    archdone       19,962    4.85%    1.9G   22.58%           2018002292
                    copy1             658    0.16%    1.8G   21.74%           1942851010
                    copy2               0
                    copy3               0
                    copy4               0

                Regular           411,479   99.88%    8.3G   99.84%           8921596219
                    offline            26    0.01%  264.1M    3.10%            276878242
                    archdone       19,492    4.73%    1.9G   22.50%           2010445172
                    copy1             189    0.05%    1.8G   21.66%           1935297986
                    copy2               0
                    copy3               0
                    copy4               0

                Segmented               0    0.00%    0       0.00%                    0
                    offline             0
                    archdone            0
                    copy1               0
                    copy2               0
                    copy3               0
                    copy4               0

                Directories           473    0.11%   13.2M    0.16%             13881344
                    offline             0
                    archdone          469    0.11%    7.2M    0.08%              7553024
                    copy1             469    0.11%    7.2M    0.08%              7553024
                    copy2               0
                    copy3               0
                    copy4               0

                Symbolic links          5    0.00%    0       0.00%                    0
                    offline             0
                    archdone            0
                    copy1               0
                    copy2               0
                    copy3               0
                    copy4               0


                Removable media         1    0.00%    4.0k    0.00%                 4096
                    offline             0
```

```
              archdone           1    0.00%    4.0k    0.00%                  4096
              copy1              0
              copy2              0
              copy3              0
              copy4              0
```

Column 2 is the number of files. Column 3 is the percent of the total number of files. Column 4 is the total size in bytes. Column 5 is the percent of the total size. Column 6 is the exact total size in bytes.

-l   List input lines. Sample output:

```
           1: logfile = /var/opt/SUNWsamfs/archiver.log
           2: interval = 5m
           3: big . -minsize 500k
           4:      1 30s
           5:   2 2h
           6: all .
           7:      1 30s
           8: params
           9: allsets -reserve set
           10: allsets.2 -archmax 5G
           11: endparams
           12: vsns
           13: samfs1.1 mo .*
           14: all.1    mo .*
           15: big.1    sg .*
           16: big.2    sg .*
```

-n file_system
          List file system content (same as -f) for a single file system.

-v   List VSNs. Only lists VSNs with space available. Sample output:

```
           Archive libraries:
           Device:mo20 drives_available:0 archive_drives:1
             Catalog:
             mo.mo0001               capacity:    1.2G space:    1.1G  -il-o-------
             mo.mo0002               capacity:    1.2G space:    1.0G  -il-o-------

           Device:tp30 drives_available:0 archive_drives:3
             Catalog:
             sg.004977               capacity:   20.0G space:   18.0G  -il-o-b-----
             sg.004978               capacity:   20.0G space:    0     -il-o-b-----

             sg.004979               capacity:   20.0G space:   10.4G  -il-o-b-----
             sg.004975               capacity:   20.0G space:   18.0G  -il-o-b-----
             sg.004970               capacity:   20.0G space:   18.0G  -il-o-b-----
             sg.004971               capacity:   20.0G space:   13.1G  -il-o-b-----

           .
           .
           .

           Archive sets:
           allsets
               reserve: set//

           allsets.1
              .reserve: set//

           allsets.2
               archmax: 5G
              .reserve: set//

           allsets.3
              .reserve: set//

           allsets.4
              .reserve: set//

           all.1
              .reserve: set//
             media: mo
            Volumes:
              mo0001
              mo0002
            Total space available:    2.1G

           big.1
              .reserve: set//
             media: sg
            Volumes:
              004977
              004979
              004975
              004970
              004971
            Total space available:   77.5G

           big.2
              .archmax: 5G
              .reserve: set//
             media: sg
            Volumes:
              004977

              004979
              004975
              004970
              004971
            Total space available:   77.5G

           samfs1.1
              .reserve: set//
             media: mo
            Volumes:
              mo0001
              mo0002
            Total space available:    2.1G
```

SEE ALSO

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

archiver.sh(1M)

```
NAME
     archiver.sh - Sun QFS or SAM-QFS archiver exception
     notification script

SYNOPSIS
     /etc/opt/SUNWsamfs/scripts/archiver.sh prg_name pid severity
     msg_no msg

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The archiver executes the
     /etc/opt/SUNWsamfs/scripts/archiver.sh script when it
     encounters abnormal or exceptional events.  You can
     substitute a site-specific version of this script by using
     the archiver's notify directive in the archiver.cmd(4) file.

     For all events, the /etc/opt/SUNWsamfs/scripts/archiver.sh
     script logs events to syslog using the /usr/bin/logger
     command.  In addition, the emerg, alert, crit, and err
     keywords generate email to the root account, echoing the
     message string.

OPTIONS
     The archiver executes /etc/opt/SUNWsamfs/scripts/archiver.sh
     and any scripts defined by the user through the notify
     directive with the following arguments:

     prg_name  The name of the program that is calling this
               script.

     pid       The process ID of the program that is calling this
               script.

     severity  A keyword that identifies the severity and the
               syslog level of the event.  The keywords are as
               follows:  emerg, alert, crit, err, warning,
               notice, info, and debug.

     msg_no    The message number as found in the message
               catalog.

     msg       The text of the translated message string.

SEE ALSO
     archiver(1M), archiver.cmd(4)
```

arcopy(1M)

```
NAME
     sam-arcopy -  SAM-QFS archive copy daemon

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-arcopy

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-arcopy process is responsible  for  copying  SAM-QFS
     files   to   removable   media.   It  is  executed  by  sam-
     archiverd(1M).  All required information is  transmitted  to
     the sam-arcopy in memory mapped files.

SEE ALSO
     sam-archiverd(1M)
```

arfind(1M)

```
NAME
     sam-arfind - SAM-QFS archive find daemon

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-arfind file_system

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     sam-arfind is responsible for finding  SAM-QFS  file  system
     files  to be archived.  It is executed by sam-archiverd(1M).
     The only argument is the name of the file system.  All other
     required  information is transmitted to sam-arfind in memory
     mapped files.

SEE ALSO
     sam-archiverd(1M)
```

auditslot(1M)

```
NAME
     auditslot - Audit slots in a robot

SYNOPSIS
     /opt/SUNWsamfs/sbin/auditslot [ -e ]  eq:slot[:partition]  [
     eq:slot[:partition]...]

AVAILABILITY
     SUNWsamfs
```

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

OPTIONS
     -e   If slot is tape, skip to EOD and  update  space  avail-
          able.  Caution:  Skip  to  EOD is not interruptible and
          under certain conditions can take hours to complete.

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

SEE ALSO
     export(1M), import(1M), move(1M), mcf(4), sam-robotsd(1M)
```

backto(1M)

```
NAME
     backto - Restores configuration files to an existing
     release’s condition

SYNOPSIS
     backto level

AVAILABILITY
     SUNWsamfsr

     SUNWqfsr

DESCRIPTION
     The Sun QFS and SAM-QFS upgrade process moves certain files,
     for example license files, to new locations.  If you revert
     to a previous release, use the backto script to restore
     these files to their previous locations and formats.  Run
     this script before you remove the current release package.

     This command accepts the following options:

     level     Meaning

     4.2       Use this argument to revert to the 4.2 releases.

     4.3       Use this argument to revert to the 4.3 releases.

     5.2       Use this argument to revert to the 5.2 releases.

     Because some files have paths or arguments added that do not
     work on earlier systems, these files are not moved directly.
     For such files, either go back to the previous version of
     the file or edit the most current release’s version of the
     file to remove path changes and new features.
```

build_cat(1M)

```
NAME
     build_cat - Build a media changer catalog file

SYNOPSIS
     /opt/SUNWsamfs/sbin/build_cat [ -t media ] file catalog

     /opt/SUNWsamfs/sbin/build_cat [ -t media ] - catalog

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     build_cat will build a catalog file from  file.  If  ’-’  is
     substituted  for file, standard input will be used.  If nei-
     ther file or ’-’ is given, the usage message is emitted  and
     build_cat exits.

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

     The fields, in order, on each line are:

     Index     The index of this entry within the  catalog.   The
               index  must be an incrementing integer starting at
               zero.

     vsn       The volume serial name of the media.  If there  is
               no  volume  serial  name  then  the  character "?"
               should be used.

     bar code  The bar code or volser for the media.  If there is
               no  bar code then the string NO_BAR_CODE should be
               used.

     media type
               The media type for this media (see mcf(4)).

     ptoc-fwa  The next position to be used to write data to  the
               media.

     access count
               The number of times the media has been mounted.

     capacity  The capacity of the device in 1024-byte units.

     space avail
               The amount of space left in 1024-byte units.

     flags     The flags field from the catalog entry, in numeric

               form.

     sector size
```

```
               The tape block size or optical disk sector size.

     label time
               The time that the medium was labeled.

     slot      The  slot  containing  the   volume   within   the
               automated library.

     partition The partition or side of  a  magneto-optical  car-
               tridge.   The value of partition is 0 for tapes, 1
               or 2 for m-o cartridges.

     modification time
               The time the medium was last modified.

     mount time
               The time the medium was last mounted.

     reserve time
               The time the volume was reserved.  A  value  of  0
               means no reservation.

     reservation
               The volume  reservation  -  archive-set/owner/file
               system.

     information field
               Information about  this  volume  supplied  by  the
               user.

     lvtime    The last verified time for a tape.

     lvpos     The last verified position on tape.

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

FOREIGN MEDIA
     build_cat can be used to generate a catalog that contains  a
     combination  of  usual  SAM-QFS  media and so-called foreign
     media. Foreign media are those that use a  different  format
     from  SAM-QFS.  The  migration  toolkit (SAMmigkit) provides
     hooks for the site to use to enable SAM-QFS file systems  to

     stage (and optionally re-archive) data from foreign media.

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

SEE ALSO
     dump_cat(1M),   export(1M),   import(1M),    mcf(4),    sam-
     robotsd(1M)
```

chmed(1M)

NAME

chmed - Set or clear library catalog flags and values

SYNOPSIS

```
/opt/SUNWsamfs/sbin/chmed [-b] +flags specifier

/opt/SUNWsamfs/sbin/chmed [-b] -flags specifier

/opt/SUNWsamfs/sbin/chmed [-b] -capacity capacity specifier

/opt/SUNWsamfs/sbin/chmed [-b] -space space specifier

/opt/SUNWsamfs/sbin/chmed [-b] -time time specifier

/opt/SUNWsamfs/sbin/chmed [-b] -count count specifier

/opt/SUNWsamfs/sbin/chmed [-b] -vsn vsn specifier

/opt/SUNWsamfs/sbin/chmed [-b] -mtype media specifier

/opt/SUNWsamfs/sbin/chmed [-b] -I information specifier
```

AVAILABILITY

SUNWsamfs

WARNING

chmed sets or clears flags and values in a library catalog entry. These values are critical to the operation of the SAM-QFS environment 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.

OPTIONS

This command accepts the following argument:

-b Displays size in base 10 units. This command displays a modified catalog entry upon successful completion, size is displayed in base 2 units by default.

ARGUMENTS

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

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

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

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

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

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

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

A	needs audit
C	slot contains cleaning cartridge
E	volume is bad or expired cleaning media
N	volume is not in SAM-QFS format
R	volume is read-only (software flag)
U	volume is unavailable
W	volume is physically write-protected
X	slot is an export slot
b	volume has a bar code
c	volume is scheduled for recycling
f	volume found full or foul by archiver
d	volume has a duplicate vsn
l	volume is labeled
o	slot is occupied
p	high priority volume

NOTE: The f flag can mean that the volume is 100% full or that there is a problem with the tape. This can happen when a new tape is imported into the library with a partial label, or with a tape that does not have an EOD.

I is an information field to hold information on a volume. A maximum of 128 characters is allowed and these characters must be enclosed in quotation marks. An example is:

"Warehouse A, room 310, shelf 3"

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

space is the total number of bytes remaining to be written on the volume. The space may be specified with ’k’, ’M’,

’G’, ’T’, ’P’, and ’E’ multipliers. e.g. 200.5M or 0.2005G.

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

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

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

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

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

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

vsn gives the VSN of the volume to be affected.

DESCRIPTION

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

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

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

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

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

FOREIGN MEDIA

chmed can be used to modify existing catalog entries so that they denote so-called foreign media. Foreign media are those that are not in SAM-QFS format. The migration toolkit (SAMmigkit) provides hooks for the site to use to enable SAM-QFS file systems to stage (and optionally re-archive) data from the foreign media.

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

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

If you have many foreign cartridges, you can use build_cat to bulk load a catalog.

EXAMPLES

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

SEE ALSO
     build_cat(1M), mcf(5), sam-recycler(1M), samu(1M)

cleandrive(1M)

NAME
     cleandrive - Clean drive in media changer

SYNOPSIS
```
     /opt/SUNWsamfs/sbin/cleandrive eq
```

AVAILABILITY
     SUNWsamfs

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

     The SAM-QFS environment 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 samu (1M) VSN catalog display under the count field. The SAM-QFS environment tracks the number of cleaning cycles used for each cleaning tape. If the media changer supports the export operation, SAM-QFS file systems will export the tape when the number of remaining cycles equals zero. A DLT cleaning tape has 20 cycles and an Exabyte cleaning tape has 10 cycles. Each time a cleaning tape is imported, the cleaning cycle is reset to the highest number of cycles for that type of tape.

FILES

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

SEE ALSO
     mcf(4), sam-robotsd(1M), samu(1M)

clri(1M)

```
NAME
     clri - clear inode

SYNOPSIS
     clri [ -F samfs ] [ -V ] mount-point i-number

AVAILABILITY
     SUNWsamtp

DESCRIPTION
     clri writes zeroes on the inode numbered i-number on the Sun
     QFS or SAM-QFS file system currently mounted on mount-point.
     i-number can be expressed as either a  decimal  integer,  an
     octal integer prefixed with a zero, or a hexidecimal integer
     prefixed with 0x.

     clri must be run as  root.   Once  you've  cleared  all  the
     inodes you wish for a filesystem, you'll need to unmount and
     remount the filesystem to flush the  inode  cache  to  disk.
     Finally,  if  there are any directory entries which point at
     the newly-cleared inodes, those directory  entries  will  be
     cleared  automatically  by the filesytem the first time they
     are referenced.

EXAMPLE
     Here's an example of using clri:

          Mount the filesystem

     bilbo# mount /sam1

          Find out the inode number which must be cleared.  Here, let's say
          we would like to clear "file0."

     bilbo# cd /sam1/test
     bilbo# sls -i
        169 file0     166 file3     339 file5      60 file7     160 file9
        342 file2      63 file4     163 file6     336 file8

           Ok, now we have its inode: 169.  Let's clear it!

     bilbo# /opt/SUNWsamfs/tools/clri /sam1 169

           But, look!  It's still there!  Sure looks weird, though...

     bilbo# sls -l file0
     ----------   0 root     root             0 Dec 31  1969 file0

           Even "sync" doesn't help...

     bilbo# sync
     bilbo# sls -l file0

     ----------   0 root     root             0 Dec 31  1969 file0

           ...until we unmount and remount the filesystem.
```

```
bilbo# cd /
bilbo# umount /sam1
bilbo# mount /sam1
bilbo# cd /sam1/test

      This is actually what clears the directory entry:

bilbo# ls -l file0
file0: No such file or directory

      And, now it’s gone!

bilbo# ls -l
bilbo 64
-rw-rw----   1 root     other        218 Aug 19 16:41 file2
-rw-rw----   1 root     other        206 Aug 19 16:41 file3
-rw-rw----   1 root     other        257 Aug 19 16:41 file4
-rw-rw----   1 root     other        179 Aug 19 16:41 file5
-rw-rw----   1 root     other        230 Aug 19 16:41 file6
-rw-rw----   1 root     other        192 Aug 19 16:41 file7
-rw-rw----   1 root     other        212 Aug 19 16:41 file8
-rw-rw----   1 root     other        240 Aug 19 16:41 file9
```

damage(1M)

```
NAME
     damage - Marks archive entries as damaged

SYNOPSIS
     /opt/SUNWsamfs/sbin/damage [-a] -c copy_no [-f]
     [-m media_type [-v vsn]] [-M] [-o] filename ...

     /opt/SUNWsamfs/sbin/damage [-a] [-c copy_no] [-f]
     -m media_type [-v vsn] [-M] [-o] filename ...

     /opt/SUNWsamfs/sbin/damage [-a] -c copy_no [-f]
     [-m media_type [-v vsn]] [-M] [-o] -r dirname ...  filename
     ...

     /opt/SUNWsamfs/sbin/damage [-a] [-c copy_no] [-f]
     -m media_type [-v vsn] [-M] [-o] -r dirname ...  filename
     ...

AVAILABILITY
     SUNWsamtp

DESCRIPTION
     The damage command marks archive copies as damaged.  The
     command marks copies of one or more files or directories as
     damaged based on the archive copy number and/or the media
     type and VSN specified.  There are several ways to mark one
     or more archive copies as damaged.  These ways are as
     follows:

     o  By copy number
     o  By copy number, media type, and VSN

     o  By copy number and media type

     o  By media type

     o  By media type and VSN

     If a fatal error is detected when staging an archive copy,
     that archive copy is marked as damaged.  An archive copy
     that is damaged is not selected for staging.

OPTIONS
     This command accepts the following options:

     -a        Rearchives the damaged copy.

     -c copy_no
               Marks the specified archive copy number as
               damaged.  If one or more -c options are are
               specified, only those archive copies (copies 1, 2,

               3, or 4) are marked as damaged.  Specify 1, 2, 3,
               or 4 for copy_no.  Either a -c or a -m option must
               be specified.

     -f        Suppresses errors.

     -m media_type
               Marks all copies from the specified media_type as
               damaged.  For the list of possible media_type
               specifications, see the mcf(4) man page.  Either a
               -c or a -m option must be specified.  If you
               specify a -m option, you can also specify a -v
               option.

     -M        Marks only metadata as damaged.  This includes
               directories, the segment index, and
               removable-media files.  Regular files are not
               marked as damaged.  If you are marking a directory
               as damaged, you must specify the -M option.

     -o        Specifies that the file must be online before it
               is marked as damaged.  If the file is offline, the
               damage command stages the file to disk before
               deleting any entries.

     -r dirname ...
               Recursively marks one or more specified dirnames
               and subdirectories as damaged.  The archive
               entries of files in the directories and
               subdirectories are marked as damaged.

     -v vsn    Marks the archive copies on vsn as damaged.  For
               vsn, specify a volume serial name (VSN).  If you
               specify a -v option, you must also specify a -m
               option.
     filename ...
              Marks the archive copies for one or more specified
              filename arguments as damaged.

SEE ALSO
     mcf(4).
```

dev_down.sh(1M)

```
NAME
     dev_down.sh - SAM-QFS device down notification script

SYNOPSIS
     /etc/opt/SUNWsamfs/scripts/dev_down.sh prg_name pid
     log_level msg_no eq

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The /etc/opt/SUNWsamfs/scripts/dev_down.sh script can be
     executed by the sam-robotsd(1M) daemon when a device is
     marked down or off.

     To enable this feature, copy
     /opt/SUNWsamfs/examples/dev_down.sh to
     /etc/opt/SUNWsamfs/scripts/dev_down.sh and modify it to take
     the desired action for your installation.

     As released, the /opt/SUNWsamfs/examples/dev_down.sh script
     sends email to root with the relevant information.

OPTIONS
     This script accepts the following arguments:

     prg_name  The name of the program that is calling this
               script.

     pid       The process ID of the program that is calling this
               script.

     log_level Log priority level.  An integer number such that
               0 < log_level < 7.  0 is highest priority, and 7
               is lowest priority.

     msg_no    The message number as found in the message
               catalog.

     eq        The Equipment Number of the device.

EXAMPLE
     The following is an example
     /etc/opt/SUNWsamfs/scripts/dev_down.sh file:

      #!/bin/sh
      #
```

```
#   /etc/opt/SUNWsamfs/scripts/dev_down.sh - Take action in the
#       event a device is marked down by the SAM-QFS software.
#
#   arguments:  $1: caller

#                   $2: caller’s pid
#                   $3: logging level
#                   $4: message catalog number
#                   $5: device identifier
#
# Change the email address on the following line to send
# email to the appropriate recipient.
/usr/ucb/mail -s "SAM-QFS Device downed" root <<EOF
‘date‘
SAM-QFS has marked the device $5,
as down or off.  Check device log.
EOF
```

The example sends email to root to report that a device has been marked down or off.

SEE ALSO

sam-robotsd(1M).

dmpshm(1M)

NAME

dmpshm - Dumps Sun QFS and SAM-QFS shared memory segments

SYNOPSIS

dmpshm

AVAILABILITY

SUNWsamfs

DESCRIPTION

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

dump_cat(1M)

NAME

dump_cat - Dumps the media changer catalog file in text format

SYNOPSIS

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

AVAILABILITY

SUNWsamfs

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

OPTIONS
     -n        Outputs the count of entries in the catalog.

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

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

EXAMPLES
     The following is a sample dump_cat listing:

     # audit_time Wed Dec 31 18:00:00 1969
     # version 410  count 32 mediatype
     # Index  VSN     Barcode  Type   PTOC Access Capacity  Space Status    Sector Label time  Slot Part \
Modification time Mount time Reserved Time Archive-Set/Owner/File System Volume Location LVTime LVPos
     #   ---status---  ---label time----  --last mod time--  ----mount time---
     #
       0 004974 004974 sg       0    3 19915760 19915760 0x6a000200 131072 0x3f
     8aec0f     0   0          0 0x3fdf5a79          0 // NO_INFORMATION 0 0
     #   -il-o-b-----  10/13/03 13:16:47  12/31/69 18:00:00  12/16/03 13:18:17
       1 000120 000120 sg     0x4    4 19915760 19915760 0x6a000200 262144 0x3f
     d0ae8e     1   0 0x3fdf51c0 0x3fdf51be          0 // "1440 Northland Drive, Shelf 15" 0 0
     #   -il-o-b-----  12/05/03 10:13:02  12/16/03 12:41:04  12/16/03 12:41:02
      14 000139 000139 sg    0x20    2 19915760 19915760 0x7a000200 131072 0x3f
     8aec20    14   0 0x3fdf51ea 0x3fdf51e7          0 // "vsn 000139 arset.0 from /samfs1" 0 0
     #   -ilEo-b-----  10/13/03 13:17:04  12/16/03 12:41:46  12/16/03 12:41:43
      15 700178 700178 sg   0x804    3 19915760 19382920 0x7a000a00 262144 0x3f
     d4f746    15   0 0x3fdf5317 0x3fdf5213          0 // NO_INFORMATION 0 0

SEE ALSO
     build_cat(1M), sam-robotsd(1M).
```

dump_log(1M)

```
NAME
     dump_log - Dumps the contents of  the  fifo  and  ioctl  log
     buffers

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

AVAILABILITY
     SUNWsamfs

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

```
OPTIONS
     -f        This causes dump_log  to  run  continuously;   the
               default  is  to  dump the circular buffer once and
               then terminate.

     fifo_log  The log buffer of fifo commands sent  by  the  Sun
               QFS or SAM-QFS file system to the sam-amld daemon.
               If  none  is  specified  the  default  is  to  use
               /var/adm/log/fs_fifo_log.

     ioctl_log The log buffer of the ioctl daemon  commands  sent
               to the Sun QFS or SAM-QFS file system.  If none is
               specified    the     default     is     to     use
               /var/adm/log/fs_ioctl_log.

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

          debug   logging

     option   in   the   /etc/opt/SUNWsamfs/defaults.conf   file.
     dump_log is not intended for general use, and is provided as
     a utility to supply Oracle Corporation  analysts  with  troub-
     leshooting information when necessary.

FILES
     /var/opt/SUNWsamfs/amld/fs_fifo_log
                         fifo buffer used by Sun QFS and  SAM-QFS
                         file systems

     /var/opt/SUNWsamfs/amld/fs_ioctl_log
                         ioctl buffer used by Sun QFS and SAM-QFS
                         file systems
```

exarchive(1M)

```
NAME
     exarchive - Exchanges archive copies

SYNOPSIS
     exarchive -c copy_m -c copy_n [-f] [-M] [filename] . . .

     exarchive -c copy_m -c copy_n [-f] [-M] -r dirname
     [filename] . . .

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The exarchive command exchanges archive copies for one or
     more files or directories.  You must specify two -c options
     (see OPTIONS).

OPTIONS
     This command accepts the following options:
```

```
     -c copy_m

     -c copy_n Specifies the copies to be exchanged.  The copy_m
               is exchanged with copy_n.  Exactly two -c options
               must be specified.  The first copy (copy_m) must
               have a valid archive entry.

     -f        Suppresses errors.

     -M        Exarchives meta data only.  This includes
               directories, the segment index, and removable
               media files.  Regular files are not exarchived.
               If you are exchanging a directory, you must
               specify the -M option.

     -r dirname
               Recursively exchanges the archive entries of the
               specified dirname and its subdirectories.  The
               archive entries of files in the directories and
               subdirectories are exchanged.

     filename  Exchanges the archive copies for the specified
               filename.

SEE ALSO
     unarchive(1M).
```

export(1M)

```
NAME
     export, samexport - Export a cartridge from a robot

SYNOPSIS
     /opt/SUNWsamfs/sbin/export [-f] eq:slot
     /opt/SUNWsamfs/sbin/export [-f] mediatype.vsn
     /opt/SUNWsamfs/sbin/samexport [-f] eq:slot
     /opt/SUNWsamfs/sbin/samexport [-f] mediatype.vsn

AVAILABILITY
     SUNWsamfs

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

OPTIONS
     -f   The -f option is used for  network-attached  StorageTek
          automated  libraries only. The -f option will cause the
          volume specified to be exported to the  CAP  (Cartridge
          Access  Port)  and  the SAM-QFS catalog updated accord-
          ingly. The CAPID must be defined in the stk  parameters
          file.  See  the stk(7) man page for details on defining
```

the CAPID.

For the network-controlled libraries such as the GRAU using the GRAU ACI interface, IBM 3494, or STK libraries using ACSLS and not specifying the -f option, this utility only removes the catalog entry for the cartridge from the catalog. Physical removal and addition of cartridges within these libraries is performed by utilities supplied by GRAU, IBM, and STK.

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

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

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

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

FILES

mcf	The configuration file for SAM-QFS environments

SEE ALSO

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

fsmadm(1M)

NAME

fsmadm - Starts or stops the fsmgmtd daemon

SYNOPSIS

fsmadm action

AVAILABILITY

SUNWqfs

SUNWsamfs

DESCRIPTION

The fsmadm command starts up or shuts down the fsmgmtd daemon. You can also use this command to get the status of the daemon and to add clients that can securely access this Sun QFS server.

```
OPTIONS
     add host1[.domain1] [host2[.domain2]] ...
          Adds the hosts (host1, host2, ...) to a configuration
          file such that these hosts can remotely manage the
          local Sun QFS server.  If the host is on a different
          domain, specify the domain (host-name.domain-name).
          Requests from other hosts will be refused by the SAM-
          QFS Manager daemon.

     list This will list the hosts that can currently manage the
          local Sun QFS server.  Hosts not listed will not be
          able to manage the local Sun QFS server via the SAM-QFS
          Manager.

     remove host1 [host2] ...
          Removes the hosts (host1, host2, ...) from a
          configuration file such that these hosts can no longer
          manage the local Sun QFS server via the SAM-QFS
          Manager.

     start
          Starts the SAM-QFS Manager daemon.  Does not configure
          the daemon to automatically restart.  Does not modify
          the /etc/inittab file.

     stop Stops the SAM-QFS Manager daemon.  Does not modify
          /etc/inittab file.

     restart
          Restarts the SAM-QFS Manager daemon.  Does not modify
          /etc/inittab file.

     config arg
          Configures the automatic restart feature for the
          SAM-QFS Manager daemon.

          Specify one of the following for arg:

               arg       Action

               -n        Stops the SAM-QFS Manager daemon.  The
                         automatic restart feature is controlled
                         using the service management facility
                         (smf(5)). See smf(5) for more
                         information.

               -a        Starts the SAM-QFS Manager daemon.
                         Configures init to restart the daemon
                         every time it dies.  The automatic
                         restart feature is controlled using the
                         service management facility (smf(5)).
                         See smf(5) for more information.

     status
          Displays version and configuration information for the
          SAM-QFS Manager daemon if the SAM-QFS Manager daemon is
          running and if automatic restart is enabled.

FILES
```

```
     This command resides in the following location:

     Software and Package          Location

     Sun QFS (SUNWqfs)             /opt/SUNWqfs/sbin/fsmadm

     Sun Storage Archive Manager (SAM-QFS) (SUNWsamfs)
                                   /opt/SUNWsamfs/sbin/fsmadm

SEE ALSO
     init(1M), fsmgmtd(1M).
```

fsmdb(1M)

```
NAME
     fsmdb - SAM-QFS program to index recovery points and  gather
     file system metrics

SYNOPSIS
     /opt/SUNWsamfs/sbin/fsmdb

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     fsmdb program is invoked to index recovery points and gather
     file system metrics.

SEE ALSO
     fsmadm(1M) fsmgmtd(1M)
```

fsmgmtd(1M)

```
NAME
     fsmgmtd - SAM-QFS Manager RPC API server process

SYNOPSIS
     /opt/SUNWsamfs/sbin/fsmgmtd

AVAILABILITY
     SUNWfsmgr

DESCRIPTION
     fsmgmtd is the RPC API (Application Programmer Interface)
     server process.  After the SUNWsamfs or SUNWqfs package is
     installed, this daemon must be manually started by using the
     fsmadm(1M) utility, as follows:

     fsmadm config -a

     The preceding command starts /opt/SUNWsamfs/sbin/fsmgmtd and
     adds the fsmgmt.xml file to
     /var/svc/manifest/application/management.
```

To stop the fsmgmtd daemon and disable it in smf, enter the following command:

```
fsmadm config -n
```

After the first manual start, fsmgmtd is started by smf.

fsmgmtd(1M) performs the following other actions:

- o Registers the server program with rpcbind(1M).
- o Initializes the fsmgmt API library to keep track of changes made to the various configuration files by other processes.
- o Maintains a timestamp, which allows multiple clients to co-exist.
- o Keeps a check to note that multiple clients are modifying the configuration files and sends messages to indicate the same.
- o Initializes tracing for the daemon.

Tracing can be enabled by including an entry in /etc/opt/SUNWsamfs/defaults.conf. The following example lines enable tracing:

```
trace
all = on                       # trace all daemons
fsmgmt = on                  # enable tracing for fsmgmt

fsmgmt.options = all oprmsg  # trace all and oprmsg events
fsmgmt.size = 10M            # limit the trace file size to 10M
endtrace
```

The trace file is written to /var/opt/SUNWsamfs/trace/fsmgmt. For more information on tracing, see the Trace File Controls section in the defaults.conf(1M) man page.

The messages in the trace file convey information about the state and progress of the work performed by this daemon. The messages are primarily used by Sun engineers and support personnel to improve performance and diagnose problems. As such, the message content and format are subject to change with bugfixes and feature releases.

The tracing mechanism is similar to the mechanism used by other Sun QFS and SAM-QFS daemons. To prevent the trace files from growing indefinitely, you can implement trace file rotation. For information on this, see trace_rotate(1M). You can specify that rotation be performed when the trace files reach a certain age and size in the defaults.conf file. For more information, see the defaults.conf(4) man page.

You can start the fsmgmtd daemon at package installation

```
     time or manually later after the SUNWsamfs or SUNWqfs
     package is installed, by entering the following at the
     system prompt:

     fsmadm config -a

SEE ALSO
     init(1M), rpcbind(1M), fsmadm(1M), trace_rotate(1M).

     defaults.conf(4).
```

fsmgr(1M)

```
NAME
     fsmgr - Configures tracing for the SAM-QFS Manager, changes
     the session timeout value for the Java Web Console, and
     enables/disables external connections to the Java Web
     Console.

SYNOPSIS
     fsmgr action

AVAILABILITY
     SUNWfsmgr

DESCRIPTION
     The fsmgr command changes the trace level of SAM-QFS
     Manager, changes the session timeout value for the Java Web
     Console, and enables/disables external connections to the
     Java Web Console.

OPTIONS
     trace [ level ]
          Traces the SAM-QFS Manager execution.  If level is not
          specified, it displays the current trace level.

          You can specify zero or more of the following for
          level:

               1    Traces important messages only.

               2    Traces moderately important messages,
                    including messages in trace level 1.

               3    Traces all messages.

               off  Turns off tracing.

     session [ timeout_value_in_minutes ]
          Sets the Java Web Console Session Timeout Value.
          timeout_value_in_minutes must be an integer of 10 or
          greater.

     connection [ enable|disable ]
          Enables or disables external connections to the Java
          Web Console.
```

FILES
This command resides in the following location:

Software and Package	Location
SAM-QFS Manager (SUNWfsmgr)	/opt/SUNWfsmgr/bin/fsmgr

SEE ALSO
init(1M), fsmgmtd(1M).

fsmgr_setup(1M)

NAME
fsmgr_setup - Installs, removes, and upgrades the SAM-QFS Manager software

SYNOPSIS
```
fsmgr_setup [-h] [-u]
```

DESCRIPTION
The fsmgr_setup script installs, removes, and upgrades the SAM-QFS Manager software distributed with the Sun QFS and SAM-QFS software packages. The SAM-QFS Manager is a graphical user interface tool that allows you to configure a Sun QFS file system or a SAM-QFS file system with storage and archive management capabilities.

AVAILABILITY
SUNWfsmgr

OPTIONS
This command supports the following options:

Option	Action
-h	Displays a help message.
-u	Removes the SAM-QFS Manager software and the other supporting applications. The other supporting applications are as follows: o TomCat o JRE o Sun ONE Application Framework o Java Web Console

EXIT STATUS

Code	Message
0	Install successfully performed
1	Syntax error

2	Script perform by non-superuser
3	Abort by user
4	Failed to create log file
5	Failed to create /tmp directory
6	Failed to create package admin file
7	Unsupported platform
20	Command not issued in the designated CDROM location
21	Deprecated (Do not use)
22	Deprecated (Do not use)
23	Unsupported OS detected
24	Insufficient space in / (root) directory
25	Insufficient space in /tmp directory
26	Failed to unzip File_System_Manager_2.x.zip
27	Failed to untar Java Web Console JAR file
28	Failed to install SUNWfsmgrr
29	Failed to install SUNWfsmgru
30	Newer version of SAM-QFS Manager detected
31	Same version of SAM-QFS Manager detected
50	Failed to remove SUNWfsmgrr
51	Failed to remove SUNWfsmgru
100	Failed to install Java Web Console

FILES

The fsmgr_setup script generates a log file during the installation process. If the installation is unsuccessful, you can use this file to debug your problem or you can send the file to your authorized service provider for analysis.

The script writes the log file to the following location:

```
/var/tmp/fsmgr.setup.log.MM.dd.YYYY.HH:mm
```

SEE ALSO

Sun QFS Installation Guide

fsmupd(1M)

```
NAME
     fsmupd - Helps to update SAM-QFS Manager components

SYNOPSIS
     /opt/SUNWsamfs/sbin/fsmupd

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     fsmupd Facilitates the  updating  of  SAM-QFS  Manager  com-
     ponents from one version to the next. Also performs required
     post-install and pre-removal activities as required.

SEE ALSO
     fsmadm(1M) fsmgmtd(1M)
```

generic(1M)

```
NAME
     sam-robotsd, sam-genericd, sam-stkd, sam-ibm3494d, sam-sonyd
     - SAM-QFS media changer daemons

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-robotsd mshmid pshmid

     /opt/SUNWsamfs/sbin/sam-genericd mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-stkd mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-ibm3494d mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-sonyd mshmid pshmid equip

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-robotsd daemon starts and monitors the execution of
     the media changer library control daemons for SAM-QFS.  The
     sam-robotsd daemon is started automatically by the sam-amld
     daemon if there are any libraries defined in the mcf file.
     The sam-robotsd daemon starts and monitors the correct
     daemon for all defined libraries.  For more information on
     the mcf file, see the mcf(4) man page.

     Each library daemon is responsible for monitoring the
     preview table for the VSNs that are controlled by that
     daemon.  If a request is found for one of its VSNs, the
     daemon finds an available drive under its control and moves
     the cartridge into that drive.  When the device is ready,
     the daemon notifies the SAM-QFS library daemon, and the
     device is assigned to the waiting process.
     The identifiers are as follows:

     mshmid    The identifier of the master shared memory segment
               created by the sam-amld daemon.

     pshmid    The identifier of the preview shared memory
               segment created by the sam-amld daemon.

     equip     The equipment number of the device.

     The sam-genericd daemon controls libraries that conform to
     the SCSI II standard for media changers, and it is the
     daemon that controls the ADIC/Grau ABBA library through the
     grauaci interface.  For more information on this interface,
     see the grauaci(7) man page.

     The sam-stkd daemon controls StorageTek libraries through
     the ACSAPI interface and is included in the SAM-QFS software
     package.  For more information on this interface, see the
     stk(7) man page.

     The sam-ibm3494d daemon controls IBM 3494 tape libraries
     through the lmcpd interface and is included in the SAM-QFS
     software package.  For more information on this interface,
     see the ibm3494(7) man page.

     The sam-sonyd daemon controls Sony libraries through the
     Sony DZC-800S PetaSite Application Interface Library and is
     included in the SAM-QFS software package.  For more
     information on this interface, see the sony(7) man page.

FILES
     mcf       The master configuration file for SAM-QFS
               environments.

SEE ALSO
     sam-amld(1M).

     mcf(4).

     acl2640(7), acl452(7), grauaci(7), ibm3494(7), ibm3584(7),
     sam-remote(7), sony(7), stk(7).
```

gnutar(1M)

```
NAME
     gnutar - GNU version of tar

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

HAStoragePlus_samfs(1M)

```
NAME
     HAStoragePlus_samfs - Obtains the constituent components of
     a family set or mount point.

SYNOPSIS
     HAStoragePlus_samfs family_set|mount_point

AVAILABILITY
     SUNWqfs

     SUNWsamfs

DESCRIPTION
     The HAStoragePlus_samfs is a script to obtain the
     constituent components of a single SAM-QFS family set or
     mount point.  It outputs the constituent components as a
     comma separated list.

     The HAStoragePlus_samfs command must be run as root.

OPTIONS
     None.

EXAMPLE
     # HAStoragePlus_samfs /samfs1
```

import(1M)

```
NAME
     import - Imports cartridges into a library or the historian

SYNOPSIS
     /opt/SUNWsamfs/sbin/import [[-v volser] | [-c num -s pool]]

     [-e] [-l] [-n] eq

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

AVAILABILITY
     SUNWsamfs

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

     import 27

     The second form of the import command can be used only when
     eq is the Equipment Identifier of the default historian(7)
     and the cartridge is neither two-sided nor partitioned.
     This form adds an entry to the historian's catalog for the
     given type and the given barcode or volser.  At least one of
     the -b barcode or -v volser identifiers must be present.
     For example:

     import -b 007001 -m lt 27

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

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

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

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

               must be specified together.

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

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

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

     -l        (Network-attached StorageTek automated libraries
               only.)

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

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

     -n        Specifies that the media is unlabeled foreign tape
               (not SAM-QFS media). It is write protected and can
               be only used for read access.
     -v volser (Network-attached ADIC/GRAU, StorageTek, and IBM
               3494 automated libraries only.  For the IBM 3494
               library, this option is accepted only when running
               in shared mode; for more information, see the
               ibm3494(7) man page.)

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

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

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

               If the second form of the import command is used,

               eq must be the equipment number of the default
               historian.

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

FILES
     mcf       The configuration file for SAM-QFS environments.

SEE ALSO
     export(1M), sam-robotsd(1M).

     mcf(4).

     historian(7), ibm3494(7).
```

itemize(1M)

```
NAME
     itemize - Catalog optical disk or jukebox

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

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     itemize generates a list of files on a specific optical disk
```

or generates a catalog listing of disks or tapes for a robot, as specified by device. device is the device name or equipment number defined in the mcf file.

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

```
  file ID   version   length   uid   gid
```

These fields contain the following information:

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

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

```
  EA   access_time   count   use   ty   vsn
       lvtime        lvpos
```

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

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

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

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

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

MEDIA ERROR

A read or write or positioning error was detected.

When itemizing a robot with a two line display then lvtime (last verified time) and lvpos (last verified position) are shown.

OPTIONS

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

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.

EXAMPLES

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

```
server# itemize 50
Robot VSN catalog: eq: 50   count: 476
EA      access_time  count  use  ty vsn
   0    Jan 22 15:57   117  76%  mo OPT000  SLOT VACANT
   1    Jan 22 17:17    86  76%  mo OPT001  SLOT VACANT
   2    Jan 22 15:57    55  76%  mo OPT002
   3    Jan 22 16:13    72  76%  mo OPT003
   4    Jan 22 16:29   807  76%  mo OPT004

   5    Jan 22 16:45    27  76%  mo OPT005
   6    Jan 22 17:01    44  76%  mo OPT006
   7    Jan 22 15:14    36   0%  mo OPT007
   8    Jan 22 15:14    43   0%  mo OPT008
   9    Jan 22 15:15    30   0%  mo OPT009
  10    Jan 22 13:32    45  99%  mo OPT010
  11    Jan 22 15:47    35  99%  mo OPT011
  12    Jan 22 15:49    43  99%  mo OPT012
  13    Jan 22 15:53    31  84%  mo OPT013
  14    Jan 22 09:44    30   0%  mo OPT014
  15    Jan 22 09:45    52   0%  mo OPT015
  16    Jan 22 15:57  2163  99%  mo OPT016
  17    Jan 22 12:29  1618   0%  mo OPT017
```

This example shows an itemize listing from an optical disk:

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

.

This example shows an itemize two line listing for a Robot VSN catalog with DIV media. A lvtime (last verified time) of none indicates the media has never had a completed tpverify(1M) run. A non-zero lvpos (last verified position) indicates the last tpverify was canceled and the starting position of the next verify.

```
server# itemize -2 20
Robot VSN catalog: eq: 20       count: 3
slot    access_time  count  use  ty vsn
        lvtime       lvpos
   0    Apr  5 16:34     6   0% ti 000219
        Apr  4 09:23 0
   1    Apr  5 16:17    28  29% ti 000210
        Apr  5 15:12 0x9bb9
   2    Apr  5 16:35     4   0% ti 000211
        none         0
```

load(1M)

NAME
samload, load - Loads media into a device

SYNOPSIS
```
/opt/SUNWsamfs/sbin/samload [ -w ] eq:slot[:partition] [ deq
]

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

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

/opt/SUNWsamfs/sbin/load [ -w ] mediatype.vsn [ deq ]
```

AVAILABILITY
SUNWsamfs

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

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

The load and samload commands are identical; samload is pro-

```
     vided  as an alternative to avoid conflict with the Tcl com-
     mand of the same name.

OPTIONS
     -w        load will  wait  for  the  operation  to  complete
               before terminating.

FILES
     mcf                 The  configuration  file   for   SAM-QFS
                         environments

SEE ALSO
     unload(1M), set_state(1M), mcf(4), sam-robotsd(1M)
```

load_notify.sh(1M)

```
NAME
     load_notify.sh - Sends email when a volume needs to be
     imported or loaded

SYNOPSIS
     /opt/SUNWsamfs/examples/load_notify.sh prg_name pid
     log_level msg_no vsn

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The /etc/opt/SUNWsamfs/scripts/load_notify.sh script is
     called by the appropriate media-changer daemon when a
     requested volume is not in a library, is not marked
     unavailable, and the attended state is set to yes.
     Appropriate media-changer daemons include sam-genericd,
     sam-stkd, and so on.  For more information on the
     media-changer daemons, see the sam-robotsd(1M) man page.

     By default, this script sends email to root with the
     following message:

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

     To enable this feature, copy
     /opt/SUNWsamfs/examples/load_notify.sh to
     /etc/opt/SUNWsamfs/scripts/load_notify.sh and modify it to
     take the desired action for your installation.

OPTIONS
     This script accepts the following arguments:

     prg_name  The name of the program that is calling this
               script.

     pid       The process ID of the program that is calling this
               script.
     log_level Log priority level.  An integer number such that
               0 < log_level < 7.  0 is highest priority, and 7
               is lowest priority.

     msg_no    The message number as found in the message
               catalog.

     vsn       The volume serial name (VSN) identifier of the
               volume that needs to be imported or loaded.

FILES
     /opt/SUNWsamfs/examples/load_notify.sh

SEE ALSO
     sam-fsd(1M), sam-robotsd(1M), samset(1M).
```

log_rotate.sh(1M)

```
NAME
     log_rotate.sh - Rotates log files

SYNOPSIS
     /opt/SUNWsamfs/examples/log_rotate.sh file [ minsize ]

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The log_rotate.sh script rotates log files generated by Sun
     Storage Archive Manager (SAM-QFS) environments and other
     programs.

     The process of rotating log files assumes that you want to
     keep no more than seven generations of a file in your
     directories at one time.  If the size of file is minsize or
     greater, the files are rotated.  When the files are rotated,
     the newest file is renamed file.1, the next-newest file is
     renamed file.2, and so on.  The oldest file in the directory
     is deleted as new ones are added, so the oldest file in the
     directory at any time is always called file.7.  This process
     provides the following benefits:

     o  A given file never becomes so large that it is unwieldy
        to copy or view.

     o  Entries are expired after a period of time.  This
        prevents file systems from filling up due to the volume
        of log entries.

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

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

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

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

OPTIONS

This script accepts the following arguments:

file	The log file to be rotated. For example, sam-log.
minsize	Specify an integer number, in bytes, that represents the minimum size of the log file to be rotated. Log files smaller than this minimum are not rotated. The default minsize is 100000.

To enable this script, copy it from /opt/SUNWsamfs/examples/log_rotate.sh to /opt/SUNWsamfs/scripts/log_rotate.sh, modify it to take the desired action for your installation, and set up a crontab(1) entry to run the log_rotate.sh script.

EXAMPLES

The examples that follow assume that you have copied the script from its location in /opt/SUNWsamfs/examples/log_rotate.sh to /opt/SUNWsamfs/scripts/log_rotate.sh.

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

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

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

```
     Example 2.  To rotate file releaser-log every week, the
     entry would appear as follows:

     40 2 * * 0  /etc/opt/SUNWsamfs/scripts/log_rotate.sh /var/adm/releaser-log

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

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

     /opt/SUNWsamfs/examples/log_rotate.sh

SEE ALSO
     crontab(1), syslogd(1M).
```

mccfg(1M)

```
NAME
     mccfg - Media Changer Configuration

SYNOPSIS
     /opt/SUNWsamfs/tools/mccfg

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The mccfg command is a two-part Sun Storage Archive Manager
     (SAM-QFS) media changer and tape drive installation
     configuration script.

     The first part configures /kernel/drv/samst.conf lun 0 fc-
     fabric media changers.

     The second part configures /etc/opt/SUNWsamfs/mcf for both
     parallel and fc-fabric SCSI media changers, media changer
     installed tape drives, and standalone tape drives.  Media
     changer installed tape drives are put in the required
     /etc/opt/SUNWsamfs/mcf ascending order for proper
      SAM-QFS operation. A historian is also be added to the
     /etc/opt/SUNWsamfs/mcf.  Missing device information in
     either /etc/opt/SUNWsamfs/inquiry.conf or
     /kernel/drv/st.conf produces a warning.

     Multipath devices are supported where the first path seen is
     used in the /etc/opt/SUNWsamfs/mcf configuration.

LIMITATIONS
     Network-attached media changer installed tape drives are
     configured as standalone tape drives.  The user must create
     the /etc/opt/SUNWsamfs/mcf virtual library entry and change
     the standalone tape drive’s family set to that of the
     virtual library’s.
     Parallel SCSI-2 media changer and installed tape drives need
     to be on the same SCSI bus for automatic configuration.
     However, a single bus is not a requirement of SAM-QFS, and
     better performance is achieved with multiple buses.

     A media changer which is lun 1 fc-fabric WWN must be
     manually /kernel/drv/samst.conf configured because of cfgadm
     and luxadm limitations.

     For automatic configuration to work, all removable media
     equipment must be able to be opened and respond to a USCSI
     commands.

FILES
     mcf       The configuration file for SAM-QFS environments.

NOTES
     Run samd config for mccfg changes to take effect.

SEE ALSO
     inquiry.conf(4), mcf(4).

     historian(7), samst(7).

     st(7D).
```

mount_samfs(1M)

```
NAME
     mount_samfs - Mounts a Sun QFS or SAM-QFS file system

SYNOPSIS
     mount -F samfs [generic_options]
     [-o FSType_specific_options] special |  mount_point

     mount -F samfs [generic_options]
     [-o FSType_specific_options] special mount_point

AVAILABILITY
     SUNWsamfs

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

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

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

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

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

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

               o  Miscellaneous Tuning Options

               o  I/O Options

               o  Storage and Archive Management Options

               o  Shared File System Options

               o  Multireader File System Options

               o  Sun QFS and SAM-QFS Options

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

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

               NOTE: If running the Sun QFS software on a Linux
               client, the available mount options are very
               limited. The following mount options are the ONLY
               ones available on a Linux client system: rw, ro,
               retry, shared, rdlease, wrlease, aplease,
               minallocsz, maxallocsz, min_pool, meta_timeo,
               noauto, and auto.

               The noauto and auto options are only recognized
               within the /etc/fstab file and min_pool only in
               the samfs.cmd file.  The maximum value for
               meta_timeo is 60.

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

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

MISCELLANEOUS OPTIONS
     The following options can be used when mounting a Sun QFS or
     SAM-QFS file system.  These options can affect file system
     features and system performance.

     nosam |  sam
               The nosam option mounts a SAM-QFS file system, but
               only the file system functionality is enabled.
               The archiving, releasing, and staging
               functionality is disabled.  When a file system is

               mounted with this option, the file system returns
               ENOSPC when it reaches 100% capacity.

               Note that mounting a file system with the nosam
               option offers no data protection for newly created
               files or for previously archived files that have
               been modified.  The default is sam.

     noarscan |  arscan
               The noarscan option disables file system scans,
               typically performed by the sam-arfind daemon, for
               finding archive candidates on a mounted file
               system.  This mount option can be useful for file
               systems in which new files are no longer being
               created yet staging and releasing are still
               desired.  The default is arscan.

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

     nogfsid |  gfsid
               The nogfsid option disables the setting of a
               global file system id, and uses the historical
               setting of the root slice device type paired with
               the file system type.  The gfsid option enables
               the setting of a global file system id, and uses
               the file system id that is stored in the
               superblock, which consists of the file system
               creation time paired with the hostid.  The default
               is gfsid.

     nocdevid |  cdevid
               The nocdevid option disables the setting of a
               global file system device id, and uses the
               historical setting of the root slice device type.
               The cdevid option enables the setting of a global
               file system device id that consists of the samioc
               module major number paired with the file system
               equipment number as specified in the mcf(4) file.
```

```
                    The default is cdevid.

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

          noquota |  quota
                    The noquota option disables file system quotas.
                    The quota option enables file system quotas,
                    provided that at least one file system quota file
                    is present.  The default is quota.  For more

                    information on quotas, see the Sun QFS File System
                    Configuration and Administration
                     Guide.

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

                    o  If sync_meta=0, metadata is held in a buffer
                       before being written to disk.  This delayed
                       write delivers higher performance.  This is the
                       default for Sun QFS and SAM-QFS file systems
                       that are not mounted as multireader file
                       systems or as Sun QFS shared file systems.

                    o  If sync_meta=1, metadata is written  to disk
                       every time it changes.  This slows performance,
                       but it ensures data consistency.  This is the
                       default for Sun QFS file systems that are
                       mounted as multireader file systems or as Sun
                       QFS shared file systems.  In a Sun QFS shared
                       file system, this is the setting that must be
                       in effect if failover capability is required.

          worm_capable
                    The worm_capable option allows Write Once Read
                    Many (WORM) files to be stored in SAM-QFS
                    filesystems.  Enabling this feature allows the
                    WORM flag to be set on files and directories. Once
                    the WORM flag is set, a file’s data and path are
                    immutable and the file can not be deleted until
                    its retention period expires.  In addition, the
                    volume on which the WORM file resides can not be
                    deleted using sammkfs.

          worm_lite The worm_lite option is similar to the
                    worm_capable mount option but eases the
                    restrictions regarding actions that can be taken
                    on WORM-enabled volumes and retained files.  WORM
                    lite enabled volumes can be deleted using sammkfs.
                    Retained files can be removed before their
                    retention period expires and their retention
                    period can be shortened (must have root
                    privileges).  File data and path remain immutable.

          worm_emul The worm_emul option is similar to the
```

```
                    worm_capable mount option and enables WORM
                    "Emulation mode".  The difference with this option
                    is the trigger used to retain files is the
                    transition from a writable to read-only file.
                    File data and path are immutable after appying the
                    WORM trigger.  A file retained in this mode can

                    not be deleted until it’s retention period
                    expires.  Volumes containing WORM emulation mode
                    files can not be deleted using sammkfs.

          emul_lite The emul_lite option is similar to the
                    worm_capable mount option and enables WORM
                    "Emulation Lite mode".  The trigger to retain
                    files is the transition from a writable to read-
                    only file.  Retained files can be removed before
                    their retention period expires and their retention
                    period can be shortened (must have root
                    privileges).  Data and path changes to a file are
                    immutable after applying the trigger.  Emulation
                    lite enabled volumes can be deleted using sammkfs.

          def_retention=n
                    The def_retention option sets the default
                    retention period.  This option requires a WORM
                    mount option enabled.  This option sets the
                    default retention period for files which have the
                    WORM feature enabled with no supplied retention
                    period.  The retention period can take three
                    forms.  A value of permanent (or 0)specifies
                    permanent retention.  A value of the form MyNdOhPm
                    where M, N, O, P are arbitrary non-negative
                    integers; y, d, h, m specify the number of years,
                    days, hours, and minute(s) respectively.  Note
                    that combinations of this form are allowed, and
                    specifiers may be omitted, e.g., 5y, 3d1h, 4m.
                    The final form is a simple integer value in
                    minutes for n, an integer 1 < n < 2147483647 (231
                    - 1).  If this option is not supplied, a 30 day
                    (43,200 minute) default retention period is used.

          rd_ino_buf_size=n
                    rd_ino_buf_size sets the size of buffer to n. This
                    is the buffer which is used to read the .inodes
                    file into buffer cache.  For n, specify an integer
                    such that 1024 < n < 16384.  n is in units of
                    bytes and rounded down to the nearest power of 2.
                    The default is 16384 bytes.

          wr_ino_buf_size=n
                    wr_ino_buf_size sets the size of the buffer to n.
                    This is the buffer which is used to synchronously
                    write an inode through to the disk.  For n,
                    specify an integer such that 512 < n <
                    rd_ino_buf_size.  n is in units of bytes and
                    rounded down to the nearest power of 2.  The
                    default is 512 bytes.

     BLOCK FILE SYSTEM GENERIC OPTIONS
          The following options are available for Sun QFS and SAM-QFS
          file systems.  Also see the mcf(4) man page.

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

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

                    o  128 kilobytes/DAU for DAUs < 128 kilobytes

                    o  1 for DAUs > 128 kilobytes

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

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

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

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

          dio_rd_form_min=n

                    Sets the read well-aligned lower limit to n 1024-
                    byte blocks.  By default, n=256, 1024-byte blocks.
                    If n=0, automatic I/O type switching for well-
                    aligned reads is disabled.

          dio_rd_ill_min=n
                    Sets the read misaligned lower limit to n 1024-
```

```
                    byte blocks.  By default, n=0, which disables
                    automatic I/O type switching for misaligned reads.

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

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

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

          atime= -1 |  0 | 1
                    The file system is mounted by default with cached
                    access time recording (atime = 0). This means
                    access time updates to disk are deferred for up to
                    1 minute after the file is last accesssed.  Note,
                    the file access time is immediately updated on
                    disk if SAM is enabled and the space used is above
                    the low water mark or when the access time
                    coincides with updates to the ctime or mtime.  See
                    stat(2). The access time is also updated when the
                    the file system is unmounted.  If atime = 1, the
                    file system will always update access time on
                    disk.  If atime = -1, the file system will not
                    update access time except when it coincides with
                    updates to the ctime or mtime.  See stat(2).  The
                    atime = -1 option reduces disk activity on file
                    systems where access times are unimportant (for
                    example, a Usenet news spool).  Note, atime = -1,
                    should not be set when SAM is enabled.

                    The POSIX standard requires that access times be
                    marked on files.  Note, for atime = 0 (the
                    default), the current access time may not be
                    updated on disk in case of an interruption.

          noatime   The noatime is added to be compatible with other
                    file systems.  If noatime is specified, atime = -1
                    will be set. This means the file system will not
                    update access time except when it coincides with
                    updates to the ctime or mtime.  See stat(2).
                    Note, noatime, should not be set when SAM is
                    enabled.

          forcedirectio
```

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

          nodio_szero |  dio_szero
                    The dio_szero option causes uninitialized areas of
                    sparse files written with direct I/O to be zeroed
                    when the area is accessed.  This makes the sparse
                    file behavior the same as that for paged I/O.  By
                    default, sparse files written by direct I/O do not
                    have the uninitialized areas zeroed for
                    performance reasons.  The default is nodio_szero.

          force_nfs_async
                    Causes the file system to cache nfs data written
                    to the server even if nfs has requested that the
                    data be written synchronously through to disk.
                    The force_nfs_async option is only useful if the
                    file system is mounted as a nfs server and the
                    clients have set the nfs mount option noac.  The
                    default nfs noac behavior without force_nfs_async
                    causes data to be synchronously written through to
                    disk. Caution, the force_nfs_async option violates
                    the nfs protocol and should be used with care.
                    Data may be lost in the event of a server
                    interruption.  Also, data is cached on the server
                    and will not be immediately seen by all the
                    clients if there are multiple nfs servers.
                    Multiple nfs servers can be enabled with Shared
                    QFS.

          sw_raid   Causes the file system to align the writebehind

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

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

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

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

          wr_throttle=n
                    Sets the maximum number of outstanding write bytes
                    for one filesystem to n kilobytes.  If n = 0,
                    there is no limit.

                    The default is 5% of system memory. Using the 5%
                    formula, and given the memory size on the left,
                    the wr_throttle setting is on the right:
                           1 GB     51 MB
                           4 GB    205 MB
                          16 GB    819 MB
                          64 GB    3.2 GB

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

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

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

          noabr |  abr
                    For Oracle RAC with SAM-QFS AIO only.  Disable
                    (enable) Application Based Recovery of software
                    mirrors.  Applies only to SAM-QFS filesystems
                    built on Solaris Volume Manager mirrored volumes
                    that likewise support Application Based Recovery.
                    Default is enabled.

          nodmr |  dmr
                    For Oracle RAC with SAM-QFS AIO only.  Disable
                    (enable) Directed Mirror Reads of software
                    mirrors.  Applies only to SAM-QFS filesystems
                    built on Solaris Volume Manager mirrored volumes
                    that likewise support directed mirror reads.
                    Default is enabled.
```

```
STORAGE AND ARCHIVE MANAGEMENT OPTIONS
     The following options can be used when mounting a SAM-QFS
     file system.  These options pertain to the storage and
     archive management facilities of these file systems.

     nosam_db |  sam_db
               The nosam_db option indicates there is no
               associated database with this file system.  The
               sam_db option indicates there is an associated
               database with this file system and file system
               activity logging is enabled. The sam-fsd daemon
               starts sam-fsalogd who logs file system activity.
               For more information, see the fsalogd.cmd(4) man
               page.  The default is nosam_db.

     high=n    Sets the high-water mark for disk cache
               utilization to n percent.  When the amount of
               space used on the disk cache reaches n percent,
               the SAM-QFS file systems start the releaser
               process.  For more information, see the sam-
               releaser(1M) man page. If n is set to 100,
               releaser is not started and ENOSPC is returned.
               The default is 80.

     low=n     Sets the low-water mark for disk cache utilization
               to n percent.  When the amount of space used on

               the disk cache reaches n percent, the SAM-QFS file
               system starts the releaser process, which stops
               releasing disk space.  The default is 70.

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

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

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

     partial_stage=n
               Sets the partial stage size for the file system to
               n kilobytes.  For a partial release file, this
               value specifies the offset in the file past which
               access results in the entire file being staged to
               disk.  For n, specify a integer from 0 to whatever
```

```
                   has been set for the maxpartial option.  The
                   default is equal to whatever has been set for the
                   partial option.

          stage_n_window=n
                   Sets the stage -n buffer size for the file system
                   to n kilobytes.  This option applies to files that
                   are read directly from the archive media.  This
                   attribute is set by using the stage(1) command's
                   -n option.  For a file with this attribute, this
                   is the size that is staged in to the application's
                   buffer at any one time.  For n, specify an integer
                   such that 64 < n < 2097152.  The default is 8192.
                   If the total number of outstanding stage_n buffers
                   is less than physical memory, the access is not
                   NFS, and the stage_n_window is less than 1%
                   physical memory, then the buffer is allocated in
                   pageable memory. Otherwise, blocks are allocated
                   for the buffer from the file system.  Note, the
                   SAM-QFS shared file system does not support stage

                   -n from a client.

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

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

     SHARED FILE SYSTEM OPTIONS
          The following options are supported for Sun QFS and SAM-QFS
          shared file systems.

          Both file system equipment types ms and ma are supported.
          For a description of the ma and ms file systems, see the
          mcf(4) man page.  For a description of the Sun QFS shared
          file system, see the Sun QFS Configuration and
          Administration Guide.

          The stripe width is set by default to round robin (using the
          stripe=0 mount option).

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

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

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

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

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

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

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

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

mh_write  Enables simultaneous reads and writes to the same
          file from multiple hosts.  If mh_write is used,
          the Sun QFS shared file system switches all hosts
          into directio. The application must use page
```

```
                    aligned memory buffers and well formed sector I/O
                    (512 bytes).  Caution, if the application does not
                    adhere to these alignment rules, data correctness
                    is not guaranteed.

                    This option is effective only on the metadata
                    server host.  If this option is specified when
                    mounting the file system on a client host, it is

                    ignored.  If the client host becomes the metadata
                    server in the future, however, this option becomes
                    effective.  For this reason, it is recommended to
                    use this mount option on the metadata host and all
                    potential metadata server hosts.  If the mh_write
                    option is not specified on the metadata server,
                    only one host can write at any one time to a
                    single file.

          min_pool=n
                    Sets the minimum number of shared file system
                    threads to keep around.  The number of threads
                    grows and shrinks dynamically based on load.  This
                    parameter tells the system to keep at least that
                    many threads in the active pool.  For n, specify
                    an integer such that 8 < n < 2048.  The default
                    n=64.  For Linux the default n=8.  NOTE:  The
                    min_pool parameter must be set in samfs.cmd file.
                    It is ignored if set in the /etc/vfstab file or on
                    the mount(1M) command.

          nstreams=n
                    * No longer used. *

          meta_timeo=n
                    Allow attributes and directory data to be cached
                    by a host system for up to n seconds before
                    checking for consistency with the metadata server.
                    The default n=3.

                    Example 1.  With the default setting of
                    meta_timeo=3, the file system verifies attribute
                    and directory consistency with the metadata server
                    at least every 3 seconds.  For instance, a new
                    file created on one host may not be seen by an
                    ls(1) command on another host for up to 3 seconds.

                    Example 2.  If meta_timeo=0, the file system
                    verifies attribute and directory consistency with
                    the metadata server before each use.  The cattr
                    mount option can be used with meta_timeo=0 to
                    ensure that changes made by other hosts currently
                    modifying a file are also immediately visible.

                    Example 3.  If meta_timeo=3, with the nocattr
                    mount option (default), the file system verifies
                    attribute consistency if it has not been checked
                    in the past 3 seconds; however, attribute changes
                    made by a client host which is currently modifying
                    a file may not be detected until the client lease
                    time has expired.

                    Example 4.  If meta_timeo=3, with the cattr mount
                    option, the file system verifies attribute
                    consistency if it has not been checked in the past
                    3 seconds, and also ensures that attribute changes
                    made by other hosts are detected within that time
                    interval.

          cattr |  nocattr
                    Enable (disable) attribute consistency checking.
                    If cattr is set, the file system ensures that
                    attribute changes made by a host which is
                    modifying a file are visible to other hosts within
                    the meta_timeo interval.  (Directories are not
                    affected by cattr; directory modifications are
                    always visible within the time interval set by
                    meta_timeo.)

                    With the default setting of nocattr, attribute
                    changes made by a host (in particular, file size
                    and modification time) may not be visible to other
                    hosts until the write or append lease time has
                    expired.

                    Note that enabling cattr may adversely affect
                    performance, as additional network traffic is
                    required.

          lease_timeo=n
                    The read, write, and/or append lease for a single
                    file is relinquished if it is not being used after
                    n seconds.  lease_timeo varies from -1 to 15
                    seconds.  If lease_timeo is >=0, the lease is
                    relinquished if it is not being used after n
                    seconds.  If lease_timeo is set to -1, the lease
                    is not relinquished and the lease expires based on
                    the lease time.  Note, the read and write lease is
                    not relinquished if mh_write is set because
                    multiple reader/writer hosts are enabled.  The
                    default n is 0.

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

          These options can be specified only on Sun QFS file systems.
          The writer option cannot be used if you are mounting the

          file system as a Sun QFS shared file system, however, the
          reader option is supported. Note, sync_meta should be set to
          1 if the reader option is used in a Sun QFS shared file
          system.

          A major difference between the multireader file system and
```

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

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

writer    Sets the file system to type writer.  There can be
          only one host system that has the file system
          mounted with the writer option at any one time.
          If writer is specified, files are flushed to disk
          at close and directories are always written
          through to disk.  The option atime = 1 is set for
          writer.

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

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

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

invalid=n When specified in conjunction with the reader
          option, holds cached attributes for the
          multireader file system at least n seconds after
          file modification.  Caution, it is possible to
          read stale data if invalid is set to a nonzero
          value.  For n, specify an integer such that
          0 < n < 60.  By default, n=0.

          Example 1.  If invalid=0, which is the default,
          the file system always checks to see if the inode

          is stale.  That is, it checks to see if the inode
          has been changed by the writer host.

          Example 2.  If invalid=30, the file system checks
          the inode 30 seconds after the last check. This
          means that if you issue an ls(1) command, you
          might not see a new file for 30 seconds after it
          has been created on the writer host.  This also
          means that if you open an existing file, for
          example with the cat(1) command, you might not see
          any changes made to the file on the writer host in
          the past 30 seconds.
```

```
     refresh_at_eof
               When specified in conjunction with the reader
               option, the current file size is refreshed when
               the read buffer exceeds the end of file.

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

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

FILES
     /etc/mnttab         Table of mounted file systems.

     /etc/vfstab         List of default parameters for each file
                         system.

     /etc/opt/SUNWsamfs/samfs.cmd
                         List of default and global parameters
                         for SAM-QFS file systems.  For more
                         information, see the samfs.cmd(4) man
                         page.

SEE ALSO
     release(1), setfa(1), ssum(1).

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

     mount(2).

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

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

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

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

move(1M)

```
NAME
     move - Move a cartridge in a library

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

AVAILABILITY
     SUNWsamfs

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

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

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

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

FILES
     mcf       The configuration file for SAM-QFS environments

SEE ALSO
     export(1M), import(1M), mcf(4), sam-robotsd(1M)
```

nrecycler.sh(1M)

```
NAME
     nrecycler.sh - Sun Storage Archive Manager (SAM-QFS)
     nrecycler post-processing script

SYNOPSIS
     /etc/opt/SUNWsamfs/scripts/nrecycler.sh gen_media vsn slot
     eq specific_media fs_name [ vsn_modifier ]

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-nrecycler(1M) process executes the
     /etc/opt/SUNWsamfs/scripts/nrecycler.sh script after it has
     finished draining a cartridge of all known active archive
```

images and recycling is complete.

As released, /etc/opt/SUNWsamfs/scripts/nrecycler.sh sends email to root with the relevant information.

OPTIONS

This script accepts the following arguments:

gen_media	Generic media type. Specify od for magneto-optical media. Specify tp for tape media. This argument is used to construct the name of the appropriate media labeling command, either odlabel(1M) or tplabel(1M).
vsn	The volume serial name (VSN) of the cartridge being processed.
slot	The slot location of the media in the library.
eq	The Equipment Number of the library in which the media cartridge is located.
specific_media	The specific media type. For information on specific media types, see the mcf man page. This information is supplied to the chmed(1M) command if needed.
fs_name	Either hy, which represents the historian, or the family set name of the library.
vsn_modifier	The VSN modifier. Used only for magneto-optical.

EXAMPLE

The following is an example /etc/opt/SUNWsamfs/scripts/nrecycler.sh file:

```
#!/bin/csh -f
#
#   /etc/opt/SUNWsamfs/scripts/nrecycler.sh - post-process a VSN after nrecycler h
as
#   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: odlabel or tplabel
#
#      $2 - VSN of cartridge being post-processed
#
#      $3 - Slot in the library where the VSN is located
#
#      $4 - equipment number of the library where the VSN is located
#
#      $5 - actual media type ("mo", "lt", etc.) - used to chmed
#           the media if required
#
#      $6 - family set name of the physical library, or the string
#           "hy" for the historian library.    This can be used to
#           handle recycling of off-site media, as shown below.
#
#          $7 - VSN modifier, used for optical and D2 media
#
#
#

#   It is a good idea to log the calls to this script
#echo ‘date‘ $* >>  /var/opt/SUNWsamfs/nrecycler.sh.log

#   As an example, if uncommented, the following lines will relabel the VSN,
#   if it exists in a physical library.  If the VSN is in the historian
#   catalog (e.g., it’s been exported from a physical library and moved
#   to off-site storage), then email is sent to "root" informing that the
#   medium is ready to be returned to the site and reused.
#
#set stat=0
#if ( $6 != hy ) then
#    /opt/SUNWsamfs/sbin/chmed -R $5.$2
#    /opt/SUNWsamfs/sbin/chmed -W $5.$2
#    if ( $5 != "d2" ) then
#        if ( $1 != "od" ) then
#            /opt/SUNWsamfs/sbin/${1}label -w -vsn $2 -old $2 $4:$3
#                       if ( $status != 0 ) then
#                           set stat = 1
#                       endif
#        else
#            /opt/SUNWsamfs/sbin/${1}label -w -vsn $2 -old $2 $4:$3:$7
#                       if ( $status != 0 ) then
#                           set stat = 1
#                       endif

#               endif
#    else
#        /opt/SUNWsamfs/sbin/${1}label -w -vsn $2 -old $2 $4:$3:$7
#               if ( $status != 0 ) then
#                       set stat = 1
#               endif
#    endif
#else
#    mail root <</eof
#VSN $2 of type $5 is devoid of active archive
#images.  It is currently in the historian catalog, which indicates that
#it has been exported from the on-line libraries.
#
#You should import it to the appropriate library, and relabel it using
#${1}label.
#
#This message will continue to be sent to you each time the nrecycler
#runs, until you relabel the VSN, or you use
#the SAM-QFS samu or SAM-QFS Manager programs to export this medium
#from the historian catalog to suppress this message.
#/eof
#endif
#echo ‘date‘ $* done >>  /var/opt/SUNWsamfs/nrecycler.sh.log
#if ( $stat != 0 ) then
#       exit 1
#else
#       exit 0
```

```
#endif
#
#
#   These lines would inform "root" that the VSN should be removed from the
#   robotic library:
#
#mail root <</eof
#VSN $2 in library $4 is ready to be shelved off-site.
#/eof
#echo ‘date‘ $* done >>  /var/opt/SUNWsamfs/nrecycler.sh.log
#exit 0

#  The default action is to mail a message reminding you to set up this
#  file.  You should comment out these lines (through and including the /eof
#  below) after you’ve set up this file.
#
mailx -s "Robot $6 at hostname ‘hostname‘ recycle." root <</eof
The /etc/opt/SUNWsamfs/scripts/nrecycler.sh script was called by the
SAM-QFS recycler
with the following arguments:

      Media type: $5($1)  VSN: $2  Slot: $3  Eq: $4
      Library: $6

/etc/opt/SUNWsamfs/scripts/nrecycler.sh is a script which is called when the recy
cler
determines that a VSN has been drained of all known active archive
copies.  You should determine your site requirements for disposition of
recycled media - some sites wish to relabel and reuse the media, some
sites wish to take the media out of the library for possible later use
to access historical files.  Consult the sam-nrecycler(1m) man page for more
information.
/eof
#echo ‘date‘ $* done >>  /var/opt/SUNWsamfs/nrecycler.sh.log
exit 0
```

The example first checks to see if the VSN is in a physical library. If it is, the example script first clears the read-only and write-protect catalog bits. It then issues a tplabel(1M) or odlabel(1M) command to relabel the cartridge with its existing label. Relabeling has the effect of clearing all the expired archive images from the cartridges, thus enabling the archiver to re-use the cartridge. Labeling also clears the recycle bit in the VSN’s catalog entry.

If the VSN is in the historian catalog, the script sends an email message to root. Note that a cartridge in a manually mounted drive is shown in the historian catalog as well, so you may want to see if the VSN is currently in a drive and relabel it if necessary.

SEE ALSO

odlabel(1M), sam-nrecycler(1M), tplabel(1M).

odlabel(1M)

```
NAME
     odlabel - Label optical media

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

AVAILABILITY
     SUNWsamfs

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

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

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

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

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

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

     -V        Verbose, lists label information written.

     -erase    Erases the media  completely  before  a  label  is
               written.   This is a security feature that is nor-
               mally not necessary.  Complete media erasure  will
               take  a long time to perform since all data in the
               media is erased.

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

qfsdump(1M)

```
NAME
     qfsdump, qfsrestore - Dump or restore file system data

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

     qfsrestore [ -dilrRstTv2 ] [-B size ]  [-b  bl_factor  ]  -f
     dump_file [file... ]

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

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

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

     If dump file contains ACLs, they could be  either  of  POSIX
     ACLs  or  NFSv4  ACLs.  Each  type  of ACL would normally be
     restored to the filesystem supporting that type of  ACL.  If
     the  dump  file  contains POSIX ACLs and the filesystem sup-
     ports NFSv4 ACLs, the POSIX ACLs will automatically be  con-
     verted  to  NFSv4 ACLs. If the dump file contains NFSv4 ACLs
     and the filesystem supports POSIX ACLs, no  conversion  will
     be  performed,  a  warning will be issued, and files will be
     restored with empty ACLs.

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

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

     -H        (qfsdump only) 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        (qfsrestore only)  Prints  inode  numbers  of  the
               files  when listing the contents of the dump.  See
               also the -l, -t, and -2 options.

     -I include_file
               (qfsdump only) Takes the list  of  files  to  dump
               from  include_file.  This file has one relative or
               absolute path to be dumped per line.   After  pro-
               cessing  include_file,  any  [file] arguments from
               the command line are processed.

     -l        (qfsrestore only) 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        (qfsdump only) Suppresses printing of warning mes-
               sages  during  the dump for those files which will
               be  damaged  should  the  dump  be  restored.   By
               default, such warning messages are displayed.

     -r        (qfsrestore only)  Replaces  existing  files  when
               restoring control structures if the existing files
               have an older modification time  than  the  dumped
               files.

     -R        (qfsrestore only)  Replaces  existing  files  when
               restoring control structures.

     -s        (qfsrestore only) Causes  leading  slashes  to  be
               stripped  from  filenames prior to restoring them.
               This is useful if the dump was made with an  abso-
               lute  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        (qfsrestore only) Instead of restoring  the  dump,
               qfsrestore  will  list  the  contents  of the dump
               file.  See also the -i, -l, and -2 options.

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

```
qfsdump statistics:
                      Files:              52020
                      Directories:        36031
                      Symbolic links:     0
                      Resource files:     8
                      File  archives:     0
                      Damaged files:      0
                      Files with data:    24102
                      File  warnings:     0
                      Errors:             0
                      Unprocessed dirs:   0
                      File data bytes:    0
```

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

"File archives" refers to the number of archive images associated with the above Files, Directories, Symbolic links and Resource files. "Damaged files" refers to the number of Files, Directories, Symbolic links, and Resource files which are either already marked damaged (for a qfsdump), or were damaged during a restore because of having no archive image (for a qfsrestore).

"Files with data" refers to the number of Files that have online (full or partial) data dumped or restored.

"File warnings" refers to the number of Files, Directories, Symbolic links and Resource files which would be damaged should the dump be restored (because they had no archive images at the time of the dump).

"Errors" refers to the number of error messages which were printed during the dump or restore. These errors are indications of a problem, but the problem is not severe enough to cause an early

exit from qfsdump or qfsrestore. Examples of errors during restore are failing to create a symbolic link, failing to change the owner or group of a file. Errors which might occur during a dump include pathname too long, failing to open a directory for reading, failing to read a symbolic link or resource file, or finding a file with an invalid mode.

"Unprocessed dirs" refers to the number of directories which were not processed due to an error (such as being unable to create the directory).

"File data bytes" is the amount of file data dumped or restored.

-v Prints file names as each file is processed. This

```
                    option is superseded by options -l or -2.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

EXAMPLES

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

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

To restore a file system dump to /qfs1:

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

SEE ALSO

sls(1), cat(1)

qfsrestore(1M)

NAME

qfsdump, qfsrestore - Dump or restore file system data

SYNOPSIS

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

qfsrestore [ -dilrRstTv2 ] [-B size ]  [-b  bl_factor  ]  -f
dump_file [file... ]
```

DESCRIPTION

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

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

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

If dump file contains ACLs, they could be either of POSIX ACLs or NFSv4 ACLs. Each type of ACL would normally be restored to the filesystem supporting that type of ACL. If the dump file contains POSIX ACLs and the filesystem supports NFSv4 ACLs, the POSIX ACLs will automatically be converted to NFSv4 ACLs. If the dump file contains NFSv4 ACLs and the filesystem supports POSIX ACLs, no conversion will be performed, a warning will be issued, and files will be restored with empty ACLs.

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

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

     -H        (qfsdump only) 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        (qfsrestore only)  Prints  inode  numbers  of  the
               files  when listing the contents of the dump.  See
               also the -l, -t, and -2 options.

     -I include_file
               (qfsdump only) Takes the list  of  files  to  dump
               from  include_file.  This file has one relative or
               absolute path to be dumped per line.   After  pro-
               cessing  include_file,  any  [file] arguments from
               the command line are processed.

     -l        (qfsrestore only) 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        (qfsdump only) Suppresses printing of warning mes-
               sages  during  the dump for those files which will
               be  damaged  should  the  dump  be  restored.   By
               default, such warning messages are displayed.

     -r        (qfsrestore only)  Replaces  existing  files  when
               restoring control structures if the existing files
               have an older modification time  than  the  dumped
               files.

     -R        (qfsrestore only)  Replaces  existing  files  when
               restoring control structures.

     -s        (qfsrestore only) Causes  leading  slashes  to  be
               stripped  from  filenames prior to restoring them.
               This is useful if the dump was made with an  abso-
               lute  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        (qfsrestore only) Instead of restoring  the  dump,
               qfsrestore  will  list  the  contents  of the dump
               file.  See also the -i, -l, and -2 options.

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

```
qfsdump statistics:
                      Files:              52020
                      Directories:        36031
                      Symbolic links:     0
                      Resource files:     8
                      File  archives:     0
                      Damaged files:      0
                      Files with data:    24102
                      File  warnings:     0
                      Errors:             0
                      Unprocessed dirs:   0
                      File data bytes:    0
```

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

"File archives" refers to the number of archive images associated with the above Files, Directories, Symbolic links and Resource files. "Damaged files" refers to the number of Files, Directories, Symbolic links, and Resource files which are either already marked damaged (for a qfsdump), or were damaged during a restore because of having no archive image (for a qfsrestore).

"Files with data" refers to the number of Files that have online (full or partial) data dumped or restored.

"File warnings" refers to the number of Files, Directories, Symbolic links and Resource files which would be damaged should the dump be restored (because they had no archive images at the time of the dump).

"Errors" refers to the number of error messages which were printed during the dump or restore. These errors are indications of a problem, but the problem is not severe enough to cause an early

exit from qfsdump or qfsrestore. Examples of errors during restore are failing to create a symbolic link, failing to change the owner or group of a file. Errors which might occur during a dump include pathname too long, failing to open a directory for reading, failing to read a symbolic link or resource file, or finding a file with an invalid mode.

"Unprocessed dirs" refers to the number of directories which were not processed due to an error (such as being unable to create the directory).

"File data bytes" is the amount of file data dumped or restored.

-v Prints file names as each file is processed. This

```
                    option is superseded by options -l or -2.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

EXAMPLES

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

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

To restore a file system dump to /qfs1:

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

SEE ALSO

sls(1), cat(1)

rearch(1M)

```
NAME
     rearch - Marks archive entries to be rearchived

SYNOPSIS
     rearch [-f] [-M] [-o] -m media -v vsn filename ...

     rearch [-f] [-M] [-o] -c n filename ...

     rearch [-f] [-M] [-o] -m media -v vsn -r dirname [filename
     ...]

     rearch [-f] [-M] [-o] -c n -r dirname [filename ...]

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The rearch command marks archive entries for one or more
     files or directories to be rearchived.  You must specify
     either a copy number or both a media type and a VSN number.
     In addition, you must specify either a file name or both a
     directory name and a file name.

OPTIONS
     This command accepts the following options:

     -c n      Specifies the archive copy number.  If one or more
               -c n options are specified, only those archive
               copies (1 to 4) are marked.  The default is all
               copies.

     -f        Suppresses errors.

     -M        Rearchives metadata only. This includes
               directories, the segment index, and removable
               media files. Regular files and symbolic links are
               not rearchived.

     -m media  Specifies the media type.  If specified, archive
               copies on the specified media are marked.  For
               more information on media types, see the mcf(4)
               man page.

     -o        Requires the file to be online before its archive
               entry is deleted.  If the file is offline, the
               command stages the file onto disk before deleting
               any entries.

     -v vsn    Marks archive copies on VSN vsn for rearchiving.
               This option must be specified in conjunction with
               the -m media option.

     -r dirname
               Recursively rearchives the archive entries of the
               specified dirname and its subdirectories.  The
               rearch flag for archive entries of files in the
```

```
                    directories and subdirectories is set.  If no -r
                    dirname option is specified, at least one filename
                    must be specified.

          filename ...
                    Specifies one or more files for rearchiving.  If
                    you are using the first form of the command,
                    either a filename or an asterisk (*) is required.
                    If you are using the third or fourth forms of the
                    command, and you do not specify a filename, you
                    must use the -r option and specify a dirname.

     SEE ALSO
          mcf(4).
```

recover.sh(1M)

```
     NAME
          recover.sh - Recovers files archived after last
          samfsdump(1M) was taken

     SYNOPSIS
          /opt/SUNWsamfs/examples/recover.sh /mount_point

     AVAILABILITY
          SUNWsamfs

     DESCRIPTION
          The recover.sh script recovers files using the information
          in the archiver log.  This script can be useful in a
          disaster recovery situation when a file system has been lost
          and is recovered from a saved samfsdump(1M) file.  If files
          were archived for the first time after the dump was taken,
          there is no record of them in the dump.  This script can be
          used to reload those files from the archive copy by using
          the star(1M) program.

     USAGE
          Step 1.   Edit the archiver log file and extract the
                    relevant portion.

                    In this editing session, you should eliminate
                    entries for second, third, or fourth archive
                    copies from this file because otherwise the files
                    are recovered multiple times, which wastes time.
                    You should also eliminate directory entries.
                    Directory entries are noted by a d in field 12 of
                    the archiver log.

                    After the file is edited, save the edited file to
                    a temporary file.  For example, save this file to
                    /tmp/arlog.in.

          Step 2.   Copy the script from its default location to a
                    temporary location.
```

Use a command such as the following to copy the script to a temporary location:

```
server# cp /opt/SUNWsamfs/examples/recover.sh /tmp/recover.sh
```

Step 3. Edit a working copy of the script and modify it for your site.

Edit the copy and change the value of BLK_SIZE from 128 to the block size in kilobytes for the VSNs in question.

Step 4. Run the recover.sh script.

This creates a new script to actually do the work of recovering the files. In the following example, the SAM-QFS mount point is /sam1.

```
server# /tmp/recover.sh /sam1 < /tmp/arlog.in > /tmp/recover.out
```

If you have multiple drives and want to recover from more than one VSN at a time, you can split this script into pieces first. The following line appears at the end of the work for each VSN:

```
"#  ----------- end of files for vsn " XXX " ----
-----"
```

The XXX is replaced with the VSN’s bar code label.

Step 5. Create a temporary directory to which the recovered files can be written.

Create this directory in a SAM-QFS file system. Although this could be your mount point, it is probably better to recover to a temporary directory in the SAM-QFS file system first, and then move the files to their final location after recovery is complete and everything looks as expected. For example:

```
server# mkdir /sam1/recover
```

Step 6. Change to the temporary directory to receive the recovered files.

Use the cd(1) command to change to the directory in which you want the files recovered.

```
server# cd /sam1/recover
server# sh -x /tmp/recover.out
```

Step 7. Run the recover.out script.

The /tmp/recover.out shell script is created in the previous step. It can be used to recover all the files listed in the /tmp/arlog.in file.

Run the recover.out script. If you have split the scripts, you may have to run it multiple times.

WARNINGS

Improper use of this script can damage user or system data. Please refer to the Disaster Planning and Recovery Guide or contact technical support before using this script.

NOTES

If used with the SAM-Remote clients or server, the recovery must be performed on the server to which the tape library is attached.

Do not run multiple recovery scripts at the same time.

FILES

This script resides in the following location:

/opt/SUNWsamfs/examples/recover.sh

SEE ALSO

archiver(1M), request(1M), star(1M).

recycler(1M)

NAME

sam-recycler - Recycles SAM-QFS volumes

SYNOPSIS

```
/opt/SUNWsamfs/sbin/sam-recycler [-b] [-c] [-C] [-d] [-E]
[-n] [-s] [-t] [-v] [-V] [-x] [-X]
[family_set | archive_set]
```

AVAILABILITY

SUNWsamfs

DESCRIPTION

The sam-recycler command invokes the recycler. The recycler removes expired archive copies and frees up archive volumes. Often, the recycler is invoked through root's crontab(1) file at an off-peak time. However, the recycler can be invoked at any time.

You can specify that only a specific library or archive set be recycled. You can recycle by library only when archiving to tape or magneto optical cartridges in a library. Note that you cannot recycle by library if you are using disk archiving.

If you want to recycle by archive set, you must name the archive sets to be recycled in the /etc/opt/SUNWsamfs/archiver.cmd file.

You can provide directives to the recycler through lines entered in the /etc/opt/SUNWsamfs/recycler.cmd file and in the /etc/opt/SUNWsamfs/archiver.cmd file. If no directives are present and no family_set or archive_set is specified on

```
     the command line, recycling does not occur.  The following
     are the default recycler settings:

     o  The maximum data quantity to recycle (-dataquantity) is 1
        gigabyte (1G).

     o  The high water mark (-hwm) is 95.

     o  The VSN gain (-mingain) is 60 for volumes <200GB and 90
        for volumes >=200GB.

     o  The number of volumes (-vsncount) to recycle is 1.

     o  Automatic email is not sent.

     NOTE: Extreme care must be taken when configuring the
     recycler if you are using disk archiving in an environment
     with multiple SAM-QFS servers. The diskvols.conf file for
     each SAM-QFS server must point to a unique set of disk
     volume resource specifications (disk archiving target

     directories). If any of these are shared between different
     SAM-QFS servers, then running the recycler from one SAM-QFS
     server will destroy the disk archive data that is being
     managed by the other SAM-QFS server.

OPTIONS
     The following options determine the volumes to be recycled
     and the content of the recycler log file.

     -b   Displays the capacity and remaining space for each
          volume in base 10 units in the recycler log file. By
          default, space is displayed in base 2 units.

     -c   Displays the extrapolated capacity of each volume.
          This is the volume's capacity assuming the compression
          observed on the volume so far continues for the rest of
          the volume.  This option produces an additional line
          for each volume with the heading Alpha:.

     -C   Suppresses listing of initial catalog(s).

     -d   Displays messages during the volume selection phase of
          processing.  These messages indicate why each volume
          was, or was not, selected for recycling.

     -E   Specifies that the volume section of the recycler's log
          file list only volumes that are not 100% free.

     -n   Prevents any actions from being taken.  This option
          causes /opt/SUNWsamfs/sbin/sam-recycler to behave as if
          -recycle_ignore were specified in the
          /etc/opt/SUNWsamfs/archiver.cmd file for all archive
          sets.

     -s   Suppresses the listing of individual volumes in the
          initial catalog section.

     -t   Recycle tape volumes only.
     -v   Displays information about which files are resident on
          the volume that is marked for recycling.  If no path
          name can be calculated for the inode, it lists the
          inode.  These files are on volumes that are being
          drained.  Using this option can consume a lot of CPU
          cycles.

     -V   Suppresses the volume section in the listing.

     -x   Displays messages for expired archive copies.  These
          are copies that are older than the time the volume upon
          which the copies reside was labeled.  Such copies
          generate an error message when staged.  The data for

          those copies is irrecoverable.  These archive copies
          must be unarchived.  If any such copies are discovered,
          the recycler stops.  This is the default behavior.
          Also see the -X option.

     -X   Inhibits the messages that indicate the existance of
          expired archive copies.  Typically, if the recycler
          detects expired archive copies, it stops.  Use this
          options if you want the recycler to continue in the
          presence of expired archive copies.  Also see the -x
          option.

     family_set | archive_set
          Recycles only the named family_set or archive_set.
          This is an optional argument.  If a family_set is
          specified, the library associated with the family set
          is recycled.  The family set is the fourth field in a
          server's mcf file.  If an archive_set is specified,
          that archive set is recycled.  The archive_set
          specified must include the copy number, as stated in
          the /etc/opt/SUNWsamfs/archiver.cmd file.  For example,
          arset.1.

          If no family_set or archive_set name is specified, the
          recycler recycles according to specifications in the
          /etc/opt/SUNWsamfs/archiver.cmd and the
          /etc/opt/SUNWsamfs/recycler.cmd files.  It examines
          each library and archive set specified.

          Regardless of a specification, only archive sets and
          family sets that have a current usage that is less than
          the high-water mark are recycled.

OPERATION
     The recycler splits its work into two phases:  volume
     selection and volume recycling.

     Phase 1 - Volume Selection
               The recycler selects volumes for recycling based
               on the amount of space used by expired archive
               copies as a percentage of total space on a volume.
               For each library or archive set being recycled,
               the volumes with the highest percentages of
               expired copies are selected to bring the media
```

utilization in the library or archive set below the configured high-water-mark. This assumes that each volume selected would contribute at least VSN-minimum-percent-gain percent of its total space if it were recycled. If no such volumes exist, the library or archive set cannot be recycled. Ties in expired space are resolved by selecting the volumes with the least amount of

unexpired space. For more information on setting a high water mark, see the recycler.cmd(4) man page.

A few conditions can prevent a volume from being selected. A volume cannot be recycled if it contains data associated with a removable media file created by the request(1) command. In addition, it cannot be recycled if it is listed in the /etc/opt/SUNWsamfs/recycler.cmd file's no_recycle section.

After volumes have been selected, they are recycled.

Phase 2 - Volume Recycling

Volume recycling differs depending upon whether the archive media is a disk volume or whether it is a removable cartridge in a library. Archiving to disk volumes is described first.

When a disk volume is selected for recycling, the volume is not marked for recycling. Additional archive copies can be written to it. Expired archive copies on the disk volume are identified and removed. Valid archive copies are left alone.

When a tape or magneto optical volume is selected for recycling, the system prevents additional archive copies from being written to it. If you are recycling to cartridges in a library, all files with active archive copies in volumes on the cartridges are marked to be re-archived. The archiver moves these copies to other volumes. In subsequent runs, the recycler checks these volumes and post-processes them when all valid archive copies have been relocated.

The recycler checks to see if there are volumes that were selected for recycling that have not yet been post-processed. If such volumes exist, and they are now devoid of active archive copies, the sam-recycler command invokes the /etc/opt/SUNWsamfs/scripts/recycler.sh(1M), which post-processes these volumes with arguments including the generic media type (tp or od), the VSN, the element address in the library, and the equipment number of the library in which the volume resides. The script can relabel the cartridge using either the original VSN or a new

VSN; or it can export the cartridge from the library; or it can perform another user-defined

action.

The /etc/opt/SUNWsamfs/scripts/recycler.sh(1M) script clears the recycling flag to indicate that recycling has completed on the volume. The odlabel(1M) and tplabel(1M) commands clear this flag after the cartridge has been relabeled.

RECYCLER OUTPUT

The recycler log is divided into several sections.

The first section describes each library catalog and archive set. The header contains the family set name or archive set name and the vendor, product, and catalog path name. Then, the capacity and remaining space for each volume appears, in bytes, with suffixes k, M, G, and T representing kilobytes, megabytes, gigabytes, and terabytes, respectively. In this log file, a kilobyte=1024 bytes, a megabyte=1024*1024 bytes, and so on by default. If -b option is specified, the capacity and remaining space for each volume appears, in base 10 units. Then, a summary, containing the total capacity and total space remaining is shown in bytes and as a percentage of space used. The recycling parameters set in the recycler and archiver command files are also shown.

The second section is a series of tables, one for each library and archive set that has associated volumes. The name of the library or archive set is shown just to the right of the ----Percent---- label. A tape volume can be associated with only one physical library. But same as disk volumes it can belong to multiple archive sets. Attempts to assign a volume to multiple archive sets are marked with a in multiple sets label. The following fields are displayed:

Field Name	Meaning
Status	A phrase giving the volume’s recycle status, as follows:

empty VSN	The volume is empty of both expired and current archive images
full VSN	The volume has no free space, but it does have current archive images.
in multiple sets	The volume matches multiple archive sets in the /etc/opt/SUNWsamfs/archiver.cmd file.
new candidate	The volume was chosen for recycling during this recycler run.

```
                    no-data VSN    The volume contains only
                                   expired archive images and
                                   free space.

                    no_recycle VSN The volume is listed in the
                                   no_recycle section of the
                                   /etc/opt/SUNWsamfs/recycler.cmd
                                   file.

                    archive -n files
                                   The volume contains archive
                                   images for files now marked as
                                   archive -n.

                    old candidate  The volume was already marked
                                   for recycling before this
                                   recycler run.

                    request files  The volume contains archive
                                   images for removeable media
                                   files.

                    partially full The volume contains both
                                   current archive images and
                                   free space.

                    shelved VSN    The volume is not currently
                                   located in any library.

     Archives Count The number of archive copies that are
                    contained on this volume.

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

     Percent Use    The percentage of space in use on this volume
                    by current archive copies.  It is estimated
                    by summing up the sizes of the archive copies
                    on the medium.  Because of compression, this
                    value can overstate the amount of space
                    actually used by these images.  This is the
                    amount of data that would need to be moved if
                    the volume were selected for recycling.

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

                    recycling this cartridge.

                    The Percent Obsolete value is calculated as
                    follows:

                    100% - In Use - Free

                    Because In Use can overstate the actual space
                    used (because of compression), the sum of In
```

```
               use + Free can exceed 100%, which renders
               Percent Obsolete to be a negative value.
               Although aesthetically unpleasing, this does
               not cause any problems in the operation of
               the recycler.

Percent Free   The percentage of free space remaining on
               this volume.  This value comes directly from
               the library catalog.  It gives the percent of
               the volume's total capacity that is available
               to hold new archive images.
```

For media that supports data compression, a best-guess value of the average compression is calculated from the ratio of the number of physical tape blocks consumed on the volume (that is, the difference of capacity - space) to the logical number of tape blocks written to the volume. The latter value is kept in the catalog. This ratio is then used to adjust the In Use value before it is written to the log file.

The first volume to appear in the log file, for each library or archive set, is the one most in need of recycling.

Here is an example recycler log file:

```
========== Recycler begins at Thu Feb  5 13:40:20 1998 ===========
3 catalogs:

0  Family: hy                    Path: /tmp/y
   Vendor: SAM-FS                Product: Historian
   EA                     ty    capacity         space vsn
      (no VSNs in this media changer)
   Total Capacity:  0    bytes, Total Space Available: 0    bytes
   Media utilization 0%, high 0% VSN_min 0%

1  Family: ad40                  Path: /var/opt/SUNWsamfs/catalog/ad40
   Vendor: ADIC                  Product: Scalar DLT 448
   EA                     ty    capacity         space vsn
      0                   lt        19.2G         0    DLT3

      1                   lt        17.7G        17.6G DLT4N
      5                   lt        17.7G        17.6G DLT6
   Total Capacity:  54.6G bytes, Total Space Available: 35.2G bytes
   Media utilization 35%, high 75% VSN_min 50%

2  Family: arset0.1              Path: /etc/opt/SUNWsamfs/archiver.cmd
   Vendor: SAM-FS                Product: Archive set
   EA                     ty    capacity         space vsn
      0                   lt         0            0    DLT5
      1                   lt        19.2G         0    DLT3
      2                   lt         0            0    DLT2
      3                   lt        17.7G        17.6G DLT4N
      4                   lt        17.7G        17.6G DLT6
   Total Capacity:  54.6G bytes, Total Space Available: 35.2G bytes
   Media utilization 35%, high 80% VSN_min 50%
   Send mail to root when this archive set needs recycling.

6 VSNs:
```

```
                    ---Archives---   -----Percent-----
-----Status-----    Count    Bytes   Use Obsolete Free   Library:Type:VSN
shelved VSN          677    648.9M                       <none>:lt:DLT0

                    ---Archives---   -----Percent-----   arset0.1
-----Status-----    Count    Bytes   Use Obsolete Free   Library:Type:VSN
no-data VSN            0      0        0   100      0    ad40:lt:DLT3
empty VSN              0      0        0     0      0  (NULL):lt:DLT2
empty VSN              0      0        0     0    100    ad40:lt:DLT6
full VSN               4     32.1k     0     0      0  (NULL):lt:DLT5
partially full         4     40.8k     0     0    100    ad40:lt:DLT4N

Recycler finished.

========== Recycler ends at Thu Feb  5 13:40:41 1998 ===========
```

Here is the corresponding archiver.cmd file:

```
interval = 2m
no_archive .
fs = samfs1
arset0 testdir0
     1 1s
     2 1s
     3 1s
     4 1s

no_archive .
fs = samfs2
no_archive .
vsns
arset0.1 lt DLT3 DLT4N DLT6 DLT1
arset0.2 lt DLT3 DLT4N DLT6 DLT1
arset0.3 lt DLT3 DLT4N DLT6 DLT1
arset0.4 lt DLT3 DLT4N DLT6 DLT1
samfs1.1 lt DLT3
samfs2.1 lt DLT4N
endvsns
params
arset0.1 -drives 4 -recycle_hwm 80 -recycle_mingain 50
endparams
```

Here is the corresponding /etc/opt/SUNWsamfs/recycler.cmd file:

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

RECYCLING HISTORIAN CARTRIDGES

The recycler recycles volumes listed in the historian's catalog. The volumes listed in the historian catalog have been exported from a library or have been or are currently in a manually-mounted device.

The /etc/opt/SUNWsamfs/scripts/recycler.sh(1M) script is passed the name hy, signifying volumes that reside in the historian catalog so that it can cope with the possibility of the volumes being recycled residing in an off-site storage facility. Typically, the /etc/opt/SUNWsamfs/scripts/recycler.sh(1M) script sends email to the administrator when this occurs to remind the administrator to bring the off-site volume back on site so that it can be reused. Volumes do not need to be on site to be drained of archive copies unless such a volume contains the only available archive copy of an off-line file.

RECYCLING BY ARCHIVE SET

When the recycler recycles by archive set, it treats each archive set as a small library that holds just the volumes assigned to the archive set in the /etc/opt/SUNWsamfs/archiver.cmd file. The volumes that are identified as belonging to a recycling archive set are removed from the recycler's version of the catalog for the library that physically contains the volume. Thus, only the volumes that are not part of an archive set remain in the library catalog.

To enable recycling for a given archive set, it must have one of the recycling options specified in the /etc/opt/SUNWsamfs/archiver.cmd file. For more information, see the archiver.cmd(4) man page.

MESSAGES

Consider the following message:

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

The preceding message means that the recycler could not set the rearchive flag for a file. When this happens, the recycler typically emits a message containing the path name, as follows:

```
Jan 22 10:17:17 jupiter sam-recycler[3400]: Cannot ioctl(F_IDSCF)
      /samfs1/testfile
```

However, in the first message, you see text beginning with Cannot find pathname.... This means that the recycler failed in its attempt to convert the inode number (in the preceding example message, it is inode number 406) and generation number (here, 25) into a path name in the /samfs1 file system.

The most likely reason for this to occur is that the file was deleted between the time that the recycler determined it needed to be rearchived and the time the recycler actually issued the system call to set the rearchive flag.

SEE ALSO

chmed(1M), odlabel(1M), recycler.sh(1M). sam-archiverd(1M), tplabel(1M).

archiver.cmd(4), mcf(4), recycler.cmd(4).

recycler.sh(1M)

```
NAME
     recycler.sh - Sun Storage Archive Manager (SAM-QFS) recycler
     post-processing script

SYNOPSIS
     /etc/opt/SUNWsamfs/scripts/recycler.sh gen_media vsn slot eq
     specific_media fs_name [ vsn_modifier ]

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-recycler(1M) process executes the
     /etc/opt/SUNWsamfs/scripts/recycler.sh script after it has
     finished draining a cartridge of all known active archive
     images and recycling is complete.

     As released, /etc/opt/SUNWsamfs/scripts/recycler.sh sends
     email to root with the relevant information.

OPTIONS
     This script accepts the following arguments:

     gen_media Generic media type.  Specify od for
               magneto-optical media.  Specify tp for tape media.
               This argument is used to construct the name of the
               appropriate media labeling command, either
               odlabel(1M) or tplabel(1M).

     vsn       The volume serial name (VSN) of the cartridge
               being processed.

     slot      The slot location of the media in the library.

     eq        The Equipment Number of the library in which the
               media cartridge is located.

     specific_media
               The specific media type.  For information on
               specific media types, see the mcf man page.  This
               information is supplied to the chmed(1M) command
               if needed.

     fs_name   Either hy, which represents the historian, or the
               family set name of the library.

     vsn_modifier
               The VSN modifier.  Used only for magneto-optical.

EXAMPLE
     The following is an example
     /etc/opt/SUNWsamfs/scripts/recycler.sh file:

     #!/bin/csh -f
     #
     #   /etc/opt/SUNWsamfs/scripts/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: odlabel or tplabel
     #
     #      $2 - VSN of cartridge being post-processed
     #
     #      $3 - Slot in the library where the VSN is located
     #
     #      $4 - equipment number of the library where the VSN is located
     #
     #      $5 - actual media type ("mo", "lt", etc.) - used to chmed
     #           the media if required
     #
     #      $6 - family set name of the physical library, or the string
     #           "hy" for the historian library.    This can be used to
     #           handle recycling of off-site media, as shown below.
     #
     #          $7 - VSN modifier, used for optical and D2 media
     #
     #
     #

     #   It is a good idea to log the calls to this script
     #echo ‘date‘ $* >>  /var/opt/SUNWsamfs/recycler.sh.log

     #   As an example, if uncommented, the following lines will relabel the VSN,
     #   if it exists in a physical library.  If the VSN is in the historian
     #   catalog (e.g., it’s been exported from a physical library and moved
     #   to off-site storage), then email is sent to "root" informing that the
     #   medium is ready to be returned to the site and reused.
     #
     #set stat=0
     #if ( $6 != hy ) then
     #    /opt/SUNWsamfs/sbin/chmed -R $5.$2
     #    /opt/SUNWsamfs/sbin/chmed -W $5.$2
     #    if ( $5 != "d2" ) then
     #        if ( $1 != "od" ) then
     #            /opt/SUNWsamfs/sbin/${1}label -w -vsn $2 -old $2 $4:$3
     #                       if ( $status != 0 ) then
     #                           set stat = 1
     #                       endif
     #        else
     #            /opt/SUNWsamfs/sbin/${1}label -w -vsn $2 -old $2 $4:$3:$7
     #                       if ( $status != 0 ) then
     #                           set stat = 1
     #                       endif

     #               endif
     #    else
     #        /opt/SUNWsamfs/sbin/${1}label -w -vsn $2 -old $2 $4:$3:$7
     #               if ( $status != 0 ) then
     #                       set stat = 1
     #               endif
     #    endif
     #else
     #    mail root <</eof
     #VSN $2 of type $5 is devoid of active archive
```

```
#images.  It is currently in the historian catalog, which indicates that
#it has been exported from the on-line libraries.
#
#You should import it to the appropriate library, and relabel it using
#${1}label.
#
#This message will continue to be sent to you each time the recycler
#runs, until you relabel the VSN, or you use the Sun QFS samu or
#SAM-QFS Manager programs to export this medium from the historian catalog to
#suppress this message.
#/eof
#endif
#echo ‘date‘ $* done >>  /var/opt/SUNWsamfs/recycler.sh.log
#if ( $stat != 0 ) then
#       exit 1
#else
#       exit 0
#endif
#
#
#   These lines would inform "root" that the VSN should be removed from the
#   robotic library:
#
#mail root <</eof
#VSN $2 in library $4 is ready to be shelved off-site.
#/eof
#echo ‘date‘ $* done >>  /var/opt/SUNWsamfs/recycler.sh.log
#exit 0

#  The default action is to mail a message reminding you to set up this
#  file.  You should comment out these lines (through and including the /eof
#  below) after you’ve set up this file.
#
mailx -s "Robot $6 at hostname ‘hostname‘ recycle." root <</eof
The /etc/opt/SUNWsamfs/scripts/recycler.sh script was called by
the SAM-QFS recycler
with the following arguments:

      Media type: $5($1)  VSN: $2  Slot: $3  Eq: $4
      Library: $6

/etc/opt/SUNWsamfs/scripts/recycler.sh is a script which is called when the recy
cler determines that a VSN has been drained of all known active archive
copies. You should determine your site requirements for disposition of
recycled media - some sites wish to relabel and reuse the media, some
sites wish to take the media out of the library for possible later use
to access historical files.  Consult the recycler(1m) man page for more
information.
/eof
#echo ‘date‘ $* done >>  /var/opt/SUNWsamfs/recycler.sh.log
exit 0
```

The example first checks to see if the VSN is in a physical library. If it is, the example script first clears the read-only and write-protect catalog bits. It then issues a tplabel(1M) or odlabel(1M) command to relabel the cartridge with its existing label. Relabeling has the effect of clearing all the expired archive images from the cartridges, thus enabling the archiver to re-use the cartridge.

```
     Labeling also clears the recycle bit in the VSN's catalog
     entry.

     If the VSN is in the historian catalog, the script sends an
     email message to root.  Note that a cartridge in a manually
     mounted drive is shown in the historian catalog as well, so
     you may want to see if the VSN is currently in a drive and
     relabel it if necessary.

SEE ALSO
     odlabel(1M), sam-recycler(1M), tplabel(1M).
```

releaser(1M)

```
NAME
     sam-releaser - SAM-QFS disk space releaser process

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-releaser file_system  low_water_mark
     weight_size [weight_age]

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-releaser process  controls  the  activities  of  the
     SAM-QFS releaser. The releaser makes disk cache available by
     identifying archived files and releasing  their  disk  cache
     copy.   This  process  is  started automatically by the file
     system when disk cache utilization  reaches  the  high-water
     mark.

     If   the    releaser    command    file    is    present    in
     /etc/opt/SUNWsamfs/releaser.cmd,  the  sam-releaser  process
     reads that file.  Directives in the  releaser.cmd  file  are
     overridden  by  the  equivalent  command-line  arguments, if
     present.  For more information on the releaser command file,
     see the releaser.cmd(4) man page.

OPTIONS
     This command accepts the following arguments:

     file_system This is the file system whose disk space  is  to
                 be  released.   The  argument  may be either the
                 name of the file  system,  or  its  mount_point.
                 The  releaser attempts to release the disk space
                 of archived files on the file system mounted  on
                 the mount_point until low_water_mark is reached.

     low_water_mark
                 A percentage of the file system that is  allowed
                 to  be  completely  occupied  with  files at all
                 times.  Specify an integer  number  that  is  at
                 least  0  but  no  more  than 100.  The releaser
                 attempts to release disk space  until  the  file
                 system is at or below this threshold.
     weight_size A weighting factor that is  used  to  prioritize
                 release  candidates.   Specify  a floating-point
                 value that is at least 0.0 but no more than 1.0.
                 For  more  information  on  weight_size, see the
                 PRIORITY WEIGHTS section of this man page.

     weight_age  A weighting factor that is  used  to  prioritize
                 release  candidates.   Specify  a floating-point
                 value that is at least 0.0 but no more than 1.0.

                 For  more  information  on  weight_age,  see the
                 PRIORITY WEIGHTS section of this man page.

ALGORITHM
     The releaser reads the SAM-QFS .inodes file  and  builds  an
     ordered  list  of the files that can be released.  The posi-
     tion of each file on the list depends on a  priority  calcu-
     lated  for  each  inode  by  the  releaser (see the PRIORITY
     WEIGHTS section of this man page.)  Only the  top  list_size
     files  are kept on the list.  The default list_size is 30000
     for file systems with less than 1M inodes, and  100000  with
     more  than 1M inodes.  See releaser.cmd(4) for a description
     of list_size.

     Starting with the file with the numerically  largest  prior-
     ity,  the disk space used by each file is released until the
     low_water_mark has been reached.  If the list  is  exhausted
     before   the  low_water_mark  is  reached,  the  process  is
     repeated.   If, while repeating the process,  no  files  are
     found that can be released, the releaser stops.  If the file
     system is still above high-water mark, the file system  res-
     tarts the releaser.

PRIORITY WEIGHTS
     Each inode is assigned a priority based on its size and age.
     The  size  of  the  file  (expressed  in units of 4-kilobyte
     blocks) is multiplied by the  weight_size  parameter.   This
     result  is  added  to the priority calculated for the age of
     the file to form the file's final priority.

     The releaser can use one of the following  two  methods  for
     determining  the  contribution  of  the age of a file to the
     file's release priority:

     o The first method is to take the most recent of the  file's
       access,  modification, and residence-change age and multi-
       ply by weight_age.

     o The second method allows specification of weights for each
       of  the  access, modification, and residence-change times.
       These  are  specified  by   the   weight_age_access=float,
       weight_age_modify=float,   and  weight_age_residence=float
       directives, respectively, in the releaser.cmd  file.   The
       sum  of the product of the weight and corresponding age is
       the contribution of the age to the  file's  priority.   To
       specify  any  of  these priority weights, you must use the
       releaser.cmd file.  For information  on  the  releaser.cmd
       file, see the releaser.cmd(4) man page.
```

For both methods, the ages are expressed in minutes.

LOG

Within the releaser.cmd file, you can specify a log file for each SAM-QFS file system. If the releaser.cmd file does not exist, or if no logfile=filename directive exists in the file, no logging occurs. For more information on the logfile=filename directive, see the releaser.cmd(4) man page.

The releaser creates the log file (if it does not exist) and appends the following to it for each run:

```
Releaser begins at Tue Sep 29 15:31:15 1998
inode pathname          /sam1/.inodes
low-water mark          40%
list_size               10000
weight_size             1
weight_age              0.5
fs equipment number     1
family-set name         samfs1
started by sam-fsd?     no
release files?          no
release rearch files?   yes
display_all_candidates? no
---before scan---
blocks_now_free:     117312
lwm_blocks:          233750
---scanning---
64122.5 (R: Tue Sep 29 11:33:21 CDT 1998) 237 min, 64004 blks S0 /sam1/250m
5131.5 (R: Tue Sep 22 17:39:47 CDT 1998) 9951 min, 156 blks S0 /sam1/filecq
5095.5 (R: Tue Sep 22 17:39:49 CDT 1998) 9951 min, 120 blks S0 /sam1/filecu
5062 (R: Tue Sep 22 18:38:50 CDT 1998) 9892 min, 116 blks S0 /sam1/filebz
5039.5 (R: Tue Sep 22 17:40:01 CDT 1998) 9951 min, 64 blks S0 /sam1/filedi
5036.5 (R: Tue Sep 22 17:37:34 CDT 1998) 9953 min, 60 blks S0 /sam1/fileio
5035.5 (R: Tue Sep 22 17:40:13 CDT 1998) 9951 min, 60 blks S0 /sam1/filedw
5032.5 (R: Tue Sep 22 17:38:08 CDT 1998) 9953 min, 56 blks S0 /sam1/filejq
5031.5 (R: Tue Sep 22 17:39:56 CDT 1998) 9951 min, 56 blks S0 /sam1/fileda
5024.5 (R: Tue Sep 22 17:38:00 CDT 1998) 9953 min, 48 blks S0 /sam1/filejh
5024 (R: Tue Sep 22 17:38:22 CDT 1998) 9952 min, 48 blks S0 /sam1/fileka
5023.5 (R: Tue Sep 22 17:40:07 CDT 1998) 9951 min, 48 blks S0 /sam1/filedn
5019 (R: Tue Sep 22 17:40:44 CDT 1998) 9950 min, 44 blks S0 /sam1/filefk
5015 (R: Tue Sep 22 17:40:28 CDT 1998) 9950 min, 40 blks S0 /sam1/fileep
5011.5 (R: Tue Sep 22 17:40:14 CDT 1998) 9951 min, 36 blks S0 /sam1/filedx
5011.5 (R: Tue Sep 22 17:39:58 CDT 1998) 9951 min, 36 blks S0 /sam1/filede
5011 (R: Tue Sep 22 17:41:07 CDT 1998) 9950 min, 36 blks S0 /sam1/filegk
5007.5 (R: Tue Sep 22 17:39:51 CDT 1998) 9951 min, 32 blks S0 /sam1/filecw
5007 (R: Tue Sep 22 17:41:10 CDT 1998) 9950 min, 32 blks S0 /sam1/filegr
5007 (R: Tue Sep 22 17:40:42 CDT 1998) 9950 min, 32 blks S0 /sam1/filefg
5007 (R: Tue Sep 22 17:40:30 CDT 1998) 9950 min, 32 blks S0 /sam1/filees
5004.5 (R: Tue Sep 22 17:38:14 CDT 1998) 9953 min, 28 blks S0 /sam1/filejv
5004 (R: Tue Sep 22 17:38:57 CDT 1998) 9952 min, 28 blks S0 /sam1/filelm
5002 (R: Tue Sep 22 18:38:54 CDT 1998) 9892 min, 56 blks S0 /sam1/filecd
4996.5 (R: Tue Sep 22 17:38:06 CDT 1998) 9953 min, 20 blks S0 /sam1/filejp

4995.5 (R: Tue Sep 22 17:39:57 CDT 1998) 9951 min, 20 blks S0 /sam1/filedc
4992.5 (R: Tue Sep 22 17:37:24 CDT 1998) 9953 min, 16 blks S0 /sam1/fileig
4992 (R: Tue Sep 22 17:39:06 CDT 1998) 9952 min, 16 blks S0 /sam1/filelv
```

```
     4986 (R: Tue Sep 22 18:38:50 CDT 1998) 9892 min, 40 blks S0 /sam1/fileca
     4982 (R: Tue Sep 22 17:36:54 CDT 1998) 9954 min, 5 blks S0 /sam1/filehk
     4981 (R: Tue Sep 22 17:41:09 CDT 1998) 9950 min, 6 blks S0 /sam1/filegn
     4980.5 (R: Tue Sep 22 17:40:15 CDT 1998) 9951 min, 5 blks S0 /sam1/filedz
     ---after scan---
     blocks_now_free:       0
     blocks_freed:          65452
     lwm_blocks:            233750
     archnodrop: 0
     already_offline: 647
     damaged: 0
     extension_inode: 0
     negative_age: 0
     nodrop: 0
     not_regular: 7
     number_in_list: 32
     rearch: 1
     released_files: 32
     too_new_residence_time: 0
     too_small: 1
     total_candidates: 32
     total_inodes: 704
     wrong_inode_number: 14
     zero_arch_status: 3
     zero_inode_number: 0
     zero_mode: 0
     CPU time: 0 seconds.
     Elapsed time: 1 seconds.

     Releaser ends at Tue Sep 29 15:31:16 1998
```

The first block of lines shows the arguments with which the releaser was invoked, the name of the .inodes file, the low-water mark, the size and age weight parameters, the equipment number of the file system, the family set name of the file system, whether the releaser was started by sam-fsd or by the command line, whether files should be released, and whether each inode should be logged as encountered.

The second block of lines begins with the heading ---before scan---. It shows the number of blocks currently free in the cache and the number that would be free if the file system were exactly at the low-water mark. The goal of the releaser is to increase blocks_now_free so that it is equal to or larger than lwm_blocks.

The third block of lines begins with the heading ---scanning---. This block lists the files released by the

releaser and contains information for each file in separate fields. The fields are as follows:

Field Number	Content
1	This field contains the release priority.
2	This field contains the date and time in the following format: (tag: date_and_time). The tag is either A for access, M for modify,

	or R for residency, depending on if the date that follows represents the access, modify or residency time. The date_and_time is the most recent of the three dates listed.
3	This field contains the age and size of the file. The age of the file is expressed in minutes. The size of the file is expressed in blocks. These two figures are multiplied by their respective weights and the sum taken to yield the release priority.
4	This field contains an S followed by the segment number. This is the number of the segment that was released.
5	This field contains the full path name of the released file.

Note that if the weight_age_access=float, weight_age_modify=float or weight_age_residence=float directives are specified in the releaser.cmd file, these lines show only the priority, size, and pathname.

The fourth block of lines begins with the heading ---after scan---. This block shows the statistics accumulated by the releaser during the previous scan pass are shown. These statistics are as follows:

Statistic	Meaning
archnodrop	The number of inodes marked archnodrop. These files are never released because the archiver is trying to keep them in cache.
already_offline	The number of inodes that were offline.
damaged	The number of inodes marked as damaged.
extension_inode	The number of extension inodes found. Used by volume overflow.
negative_age	The number of inodes that had an age in the future. This is usually caused by personal computers with incorrect clock settings acting as NFS clients.
nodrop	The number of inodes marked with release -n. For more information on marking files as never release, see the release(1) man page.
not_regular	The number of inodes that were not regular files.
number_in_list	The number of inodes that were on the

```
                         releaser's   candidate   list  when  the
                         releaser was finished scanning.

     rearch              The number of files with a  copy  marked
                         for rearchiving.

     released_files      The number of files released.

     too_new_residence_time
                         The number of  inodes  whose  residence-
                         change time was within minimum residence
                         age of the current time as specified  on
                         the  min_residence_age=time directive in
                         the releaser.cmd file.

     too_small           The number of files that were too  small
                         to be released.

     total_candidates    The number of  inodes  found  that  were
                         viable candidates for releasing.

     total_inodes        The total number of inodes scanned.

     wrong_inode_number  The number of inodes whose inode  number
                         did  not match their offset in the inode
                         file.  This is usually  not  a  concern,
                         but you should run samfsck(1M) to rescue
                         any orphan inodes.  If you have  already
                         run  samfsck(1M)  and this field remains
                         nonzero, no further action is  required.
                         For  more information on the samfsck(1M)
                         command, see the samfsck(1M) man page.

     zero_arch_status    The number of inodes that had no archive
                         copies.

     zero_inode_number   The number of inodes that  had  zero  as
                         their inode number.

     zero_mode           The number of inodes that were unused.

     CPU time            The number of CPU seconds  used  in  the
                         current scan.

     Elapsed time        The number of wall-clock seconds used in
                         the current scan.
```

NOTES

When a file is created, the residency age is set to the creation time. The residency age of a file must be at least the value set by the min_residence_age=time directive before the file is considered for release. This is to prevent a file which was recently staged in from being released. The default time is 10 minutes.

If the releaser selects a file as a release candidate, and immediately thereafter the file is accessed, the file might still be released by the file system even though the file has been recently accessed. This can happen because the file system only prohibits release of a file that is currently in use. It does not check the access age of the file again when it is released.

SEE ALSO

release(1).

mount_samfs(1M), samfsck(1M).

releaser.cmd(4).

reserve(1M)

NAME

reserve - Reserve a volume for archiving.

SYNOPSIS

```
/opt/SUNWsamfs/sbin/reserve mediatype.vsn
asname/owner/fsname [time]
/opt/SUNWsamfs/sbin/reserve eq:slot[:partition]
asname/owner/fsname [time]
```

AVAILABILITY

SUNWsamfs

DESCRIPTION

reserve assigns the volume for archival of specific files.

Normally, the archiver performs reservation of volumes. This command is provided to pre-reserve a volume.

The volume is determined by the specifier mediatype.vsn , or eq:slot[:partition]

The reservation is specified by the fields asname, owner, and fsname These fields may be empty depending on the options in the archiver command file.

time is the time the volume is reserved. If not specified, the reserve time is set to the present time. Several formats are allowed for time. Examples are:

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

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

SEE ALSO

archiver(1M), archiver.cmd(1M), unreserve(1M)

restore.sh(1M)

```
NAME
     restore.sh - Restores files online

SYNOPSIS
     restore.sh log_file mount_point

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The restore.sh script restores files to their online or
     partially online status.  This script should be used after
     performing a file system restore using the samfsrestore(1M)
     command.

     This script accepts the following arguments:

     log_file  Specify the name of the log file that was created
               by the samfsrestore(1M) command.

     mount_point
               Specify the mount point of the file system being
               restored.

USAGE
     Step 1.  Recreate or restore the file system.
     You can do this by using the samfsrestore(1M) command with
     its -g option.  This creates a log file.

     Step 2.  Run the restore.sh script.
     The first argument is the log file created in the previous
     step, and the second argument is the file system mount
     point.  This script stages back the files that were
     previously online or partially online at the time the
     .inodes copy or samfsdump(1M) was created.

FILES
     The restore.sh script resides in the following location:

     /opt/SUNWsamfs/examples/restore.sh

SEE ALSO
     Sun QFS and Sun SAM-QFS Troubleshooting Guide.

     samfsdump(1M), samfsrestore(1M).
```

robots(1M)

```
NAME
     sam-robotsd, sam-genericd, sam-stkd, sam-ibm3494d, sam-sonyd
     - SAM-QFS media changer daemons

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-robotsd mshmid pshmid

     /opt/SUNWsamfs/sbin/sam-genericd mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-stkd mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-ibm3494d mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-sonyd mshmid pshmid equip

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-robotsd daemon starts and monitors the execution of
     the media changer library control daemons for SAM-QFS.  The
     sam-robotsd daemon is started automatically by the sam-amld
     daemon if there are any libraries defined in the mcf file.
     The sam-robotsd daemon starts and monitors the correct
     daemon for all defined libraries.  For more information on
     the mcf file, see the mcf(4) man page.

     Each library daemon is responsible for monitoring the
     preview table for the VSNs that are controlled by that
     daemon.  If a request is found for one of its VSNs, the
     daemon finds an available drive under its control and moves
     the cartridge into that drive.  When the device is ready,
     the daemon notifies the SAM-QFS library daemon, and the
     device is assigned to the waiting process.

     The identifiers are as follows:

     mshmid    The identifier of the master shared memory segment
               created by the sam-amld daemon.

     pshmid    The identifier of the preview shared memory
               segment created by the sam-amld daemon.

     equip     The equipment number of the device.

     The sam-genericd daemon controls libraries that conform to
     the SCSI II standard for media changers, and it is the
     daemon that controls the ADIC/Grau ABBA library through the
     grauaci interface.  For more information on this interface,
     see the grauaci(7) man page.

     The sam-stkd daemon controls StorageTek libraries through
     the ACSAPI interface and is included in the SAM-QFS software
     package.  For more information on this interface, see the
     stk(7) man page.
```

The sam-ibm3494d daemon controls IBM 3494 tape libraries through the lmcpd interface and is included in the SAM-QFS software package. For more information on this interface, see the ibm3494(7) man page.

The sam-sonyd daemon controls Sony libraries through the Sony DZC-800S PetaSite Application Interface Library and is included in the SAM-QFS software package. For more information on this interface, see the sony(7) man page.

FILES

mcf	The master configuration file for SAM-QFS environments.

SEE ALSO

sam-amld(1M).

mcf(4).

acl2640(7), acl452(7), grauaci(7), ibm3494(7), ibm3584(7), sam-remote(7), sony(7), stk(7).

rpc.sam(1M)

NAME

sam-rpcd - SAM-QFS RPC API server process

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-rpcd

AVAILABILITY

SUNWsamfs

DESCRIPTION

sam-rpcd is the RPC API (Application Programmer Interface) server process. It is initiated by sam-amld.

sam-rpcd uses the RPC program number that is paired with the RPC program name samfs. sam-rpcd must run on the same machine as the SAM-QFS file system. You need to make the following entry in /etc/services on the server:

```
samfs       5012/tcp       # SAM-QFS 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)

sam-amld(1M)

```
NAME
     sam-amld - Initialize the SAM-QFS automated library daemons

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-amld

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     sam-amld initializes the SAM-QFS automated  library  daemons
     system  daemons.  It is typically started when a file system
     is mounted, but can be started  without  mounting  the  file
     system.

FILES
     /opt/SUNWsamfs/sbin Location of SAM-QFS daemons
     /etc/opt/SUNWsamfs  Location of SAM-QFS daemon configuration
                         files
     /etc/opt/SUNWsamfs/mcf
                         The  configuration  file   for   SAM-QFS
                         environments.

SEE ALSO
     mcf(4),    mount(1M),     mount_samfs(1M),     archiver(1M),
     generic(1M), samd(1M), scanner(1M), robots(1M)

NOTES
     To start sam-amld, use  the command samd start

     To shutdown sam-amld, use  the command samd stop
```

sam-archiverd(1M)

```
NAME
     sam-archiverd -  SAM-QFS file archive daemon

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-archiverd

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The archiver daemon  automatically  archives  SAM-QFS  files
     when  a  SAM-QFS  file  system is mounted.  It is started by
     sam-fsd, and it cannot be  executed  from  a  command  line.
     Directives  for  controlling  the archiver are read from the
     archiver       commands       file,       which       is
     /etc/opt/SUNWsamfs/archiver.cmd.  This file does not have to
     be present for the  archiver  daemon  to  execute.   If  the
     archiver.cmd  file  is  present, however, it must be free of
     errors.   Errors  in  the  archiver.cmd  file  prevent   the
```

archiver daemon from executing. If the archiver.cmd file is not present, all files on the file system are archived to the available removable media according to archiver defaults.

sam-archiverd executes in the directory /var/opt/SUNWsamfs/archiver. This is the archiver's working directory. Each sam-arfind daemon executes in a subdirectory named for the file system being archived. Each sam-arcopy daemon executes in a subdirectory named for the archive file (rm0 - rmxx) being archived to.

ARCHIVING INTERNALS

Archive Sets are the mechanism that the archiver uses to direct files in a samfs file system to media during archiving.

All files in the file system are members of one and only one Archive Set. Characteristics of a file are used to determine Archive Set membership. All files in an Archive Set are copied to the media associated with the Archive Set. The Archive Set name is simply a synonym for a collection of media volumes.

Files are written to the media in an Archive File which is written in tar format. The combination of the Archive Set and the tar format results in an operation that is just like using the command find(1) to select files for the tar command.

In addition, the file system meta data, (directories, the index of segmented files, and the removable media information), are assigned to an Archive Set to be copied to media.

The Archive Set name is the name of the file system. (See mcf(4)). Symbolic links are considered data files for the purposes of archiving.

Each Archive Set may have up to four archive copies defined. The copies provide duplication of files on different media. Copies are selected by the Archive Age of a file.

Files in an Archive Set are candidates for archival action after a period of time, the Archive Age, has elapsed. The Archive Age of a file is computed using a selectable time reference for each file. The default time reference is the file's modification time.

For processing files in archive sets with an unarchive age specified, the unarchive age default time reference is the file's access time. But, in this case, two other conditions are recognized: If the modification time is later than the access time, the modification time is used. And, if an archive copy was unarchived, the file will be rearchived only after the file is staged from another copy, i.e the file was offline at the time a read access was made to the file.

Since users may change these time references to values far in the past or future, the time reference will be adjusted by the archiver to keep it in the range: creation_time <= time_ref <= time_now.

Scheduling archive copies.

Finding files to archive.

Each file system is examined by an individual sam-arfind. The examination is accomplished by one of three methods. The method is selected by the examine = method directive. (See archiver.cmd(4)). The examination methods are:

1. Continuous archiving. Scanning directories is performed as files and directories are created and changed.

2. The 'traditional' examination mode. The first time that sam-arfind executes, all directories are recursively scanned. This assures that each file gets examined. The file status "archdone" is set if the file does not need archiving. All other scans are performed by reading the .inodes file.

3. Scan only the directory tree. Recursively descend through the directory tree. If a directory has the "noarchive" attribute set, it will not be examined. This allows the system administrator to identify directories that

contain only files and sub directories that have all archive copies and no changes will be made to the files or sub directories. This can dramatically reduce the work required to examine a file system.

4. Read the .inodes file. If an inode does not have "archdone" set, determine the file name and examine the inode. If a large percentage of the files have status "archdone" set, this method is faster than the scandirs method.

Determining the Archive Set

In this step, the archiver determines the archive set to which the file belongs using the file properties descriptions. If the Archive Age of the file has been met or exceeded, add the file to the archive request (ArchReq) for the Archive Set. The ArchReq contains a 'batch' of files that can be archived together. For segmented files, the segment, not the entire file, is the archivable unit, so the properties (e.g. minimum file size) and priorities apply to the segment. The ArchReq-s are files in separate directories for each filesystem. I.e: /var/opt/SUNWsamfs/archiver/file_system/ArchReq and you can display them by using the showqueue(1M) command. An ArchReq is removed once the files it specifies have been archived.

The characteristics used for determining which Archive Set a file belongs in are:

directory path portion of the file's name complete file name using a regular expression

user name of the file's owner

group name of the file's owner

minimum file size

maximum file size

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

Each file is given a file archive priority. The archive priority is computed from properties of the file and property multipliers associated with the Archive Set. The computation is effectively:

```
    ArchivePriority = sum(Pn * Mn)

    where:  Pn = value of a file property
            Mn = property multiplier
```

Most property values are 1 or 0 as the property is TRUE or FALSE. For instance, the value of the property 'Copy 1' is 1 if archive copy 1 is being made. The values of 'Copy 2', 'Copy 3' and 'Copy 4' are therefore 0.

Others, such as 'Archive Age' and 'File size' may have values other than 0 or 1.

The archive priority and the Property multipliers are floating point numbers. The default value for all property multipliers is 0.

The file properties used in the priority calculation are:

Archive Age	seconds since the file's Archive Age time reference (time_now - time_ref)
Copy 1	archive copy 1 is being made
Copy 2	archive copy 2 is being made
Copy 3	archive copy 3 is being made
Copy 4	archive copy 4 is being made
Copies made	number of archive copies previously made
File size	size of the file in bytes
Archive immediate	immediate archival requested for file
Rearchive	archive copy is being rearchived
Required for release	archive copy is required before file may

```
                    be released

All the priorities that apply for a file are added together.
The  priority  of  the  ArchReq  is  set to the highest file
priority in the ArchReq.

When the filesystem scan is finished, send each  ArchReq  to
sam-archiverd.

Composing archive requests.

If the ArchReq  requires  automatic  'owner'  Archive  Sets,
separate the ArchReq by owner.

Sort the files according to the 'sort' method.  Sorting  the
files  will  tend  to keep the files together in the archive
files.  The default is no  sorting  so  the  files  will  be
archived  in  the  order  encountered during the file system
scan.

Separate the ArchReq into online and offline files.  All the
online  files  will  be  archived  together, and the offline
files will be together.

The priority of each ArchReq created during this process  is
set  to the highest file priority in the ArchReq.  Enter the
ArchReq into the scheduling queue in priority order.

Scheduling from the queue.

When an ArchReq is ready to be scheduled to  an  sam-arcopy,
the  volumes are assigned to the candidate ArchReq-s as fol-
lows:
     The volume that has most recently been used for the  Archive
     Set is used if there is enough space for the ArchReq.

     If an ArchReq is too big for one volume, files that will fit
     on the volume are selected for archival to that volume.  The
     remaining files will be archived later.

     An ArchReq with a single file that is too large  to  fit  on
     one  volume,  and  is  larger than 'ovflmin' will have addi-
     tional volumes assigned as required.  The additional volumes
     are selected in order of decreasing size.  This is to minim-
     ize the number of volumes required for the file.

For each candidate ArchReq, compute the a scheduling  prior-
ity  by adding the archive priority to the following proper-
ties and the associated multipliers:

Archive volume loaded
                    the first volume to be  archived  to  is
                    loaded in a drive

Files offline       the request contains offline files

Multiple archive volumes
                    the file being  archived  requires  more
                    than one volume
     Multiple stage volumes
                         the file being archived  is  offline  on

                         more than one volume

     Queue wait          seconds that the ArchReq has been queued

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

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

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

     Assigning an ArchReq to an sam-arcopy.

     Step through each ArchReq-s to mark the archive  file  boun-
     daries  so  that each archive file will be less than archmax
     in size.  If a file is larger than archmax, it will  be  the
     only file in an archive file.

Using priorities to control order of archiving.
     By default, all archiving priorities are set to  zero.   You
     may  change  the  priorities  by  specifying property multi-
     pliers.  This allows you to control the order in which files
     are archived.  Here are some examples (see archiver.cmd(4)):

     You may cause  the  files  within  an  archive  file  to  be
     archived in priority order by using -sort priority.

     You may reduce the media loads and unloads with:   -priority
     archive_loaded 1 and -priority stage_loaded 1.

     You may cause online files to  be  archived  before  offline
     files with:  -priority offline -500.

     You may cause the archive copies to  be  made  in  order  by
     using:   -priority  copy1  4000,  -priority  copy2  3000,  -
     priority copy3 2000, -priority copy4 1000.

OUTPUT FORMAT

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

     A 2000/06/02 15:23:41 mo OPT001 samfs1.1 143.1 samfs1 6.6 16384 lost+found d 0 51
     A 2000/06/02 15:23:41 mo OPT001 samfs1.1 143.22 samfs1 19.3 4096 seg d 0 51
     A 2000/06/02 15:23:41 mo OPT001 samfs1.1 143.2b samfs1 22.3 922337 rmfile R 0 51
     A 2000/06/02 15:23:41 mo OPT001 samfs1.1 143.34 samfs1 27.3 11 system l 0 51
     A 2000/06/02 15:23:41 mo OPT001 samfs1.1 143.35 samfs1 18.5 24 seg/aa I 0 51
     A 2000/06/02 15:23:43 ib E00000 all.1 110a.1 samfs1 20.5 14971 myfile f 0 23
     A 2000/06/02 15:23:44 ib E00000 all.1 110a.20 samfs1 26.3 10485760 seg/aa/1 S 0 23
     A 2000/06/02 15:23:45 ib E00000 all.1 110a.5021 samfs1 25.3 10485760 seg/aa/2 S 0 23
```

```
A 2000/06/02 15:23:45 ib E00000 all.1 110a.a022 samfs1 24.3 184 seg/aa/3 S 0 23
A 2003/10/23 13:30:24 dk DISK01/d8/d16/f216 arset4.1 810d8.1 qfs2 119571.301 1136048 t1/fileem f 0 0
A 2003/10/23 13:30:25 dk DISK01/d8/d16/f216 arset4.1 810d8.8ad qfs2 119573.295 1849474 t1/fileud f 0 0
A 2003/10/23 13:30:25 dk DISK01/d8/d16/f216 arset4.1 810d8.16cb qfs2 119576.301 644930 t1/fileen f 0 0
A 2003/10/23 13:30:25 dk DISK01/d8/d16/f216 arset4.1 810d8.1bb8 qfs2 119577.301 1322899 t1/fileeo f 0 0
```

Field	Description
1	A for archived. R for re-archived; U for unarchived.
2	Date of archive action.
3	Time of archive action.
4	Archive media.
5	VSN. For removable media cartridges, this is the volume serial name. For disk archives, this is the disk volume name and archive tar file path.
6	Archive set and copy number.
7	Physical position of start of archive file on media and file offset on the archive file / 512.
8	File system name.
9	Inode number and generation number. The generation number is an additional number used in addition to the inode number for uniqueness since inode numbers get re-used.
10	Length of file if written on only 1 volume. Length of section if file is written on multiple volumes.
11	Name of file.
12	Type of the file. File is of type c: d directory f regular file l symbolic link R removable media file I segment index S data segment
13	Section of an overflowed file/segment.
14	Equipment number from the mcf of the device on which the archive copy was made.

SEE ALSO

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

sam-arcopy(1M)

```
NAME
     sam-arcopy -  SAM-QFS archive copy daemon

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-arcopy

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-arcopy process is responsible  for  copying  SAM-QFS
     files   to   removable   media.   It  is  executed  by  sam-
     archiverd(1M).  All required information is  transmitted  to
     the sam-arcopy in memory mapped files.

SEE ALSO
     sam-archiverd(1M)
```

sam-arfind(1M)

```
NAME
     sam-arfind - SAM-QFS archive find daemon

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-arfind file_system

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     sam-arfind is responsible for finding  SAM-QFS  file  system
     files  to be archived.  It is executed by sam-archiverd(1M).
     The only argument is the name of the file system.  All other
     required  information is transmitted to sam-arfind in memory
     mapped files.

SEE ALSO
     sam-archiverd(1M)
```

sam-catserverd(1M)

NAME
     sam-catserverd - SAM-QFS media manager daemon

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-catserverd

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-catserverd daemon keeps track of media in SAM-QFS library catalogs. A library catalog is the central repository of all information needed by the SAM-QFS environments to find cartridges in an automated library. The library catalog file is a binary, UFS-resident file that contains information about each slot in a library or manual drive. The information in the catalog includes the Volume Serial Name (VSN), the capacity and space remaining, and the flags indicating the status of the VSN.

     When the sam-catserverd daemon starts, it checks for the presence of a catalog file for each automated library defined in the mcf file. If a file is not found, the sam-catserverd daemon creates a library catalog file in the default location, /var/opt/SUNWsamfs/catalog/family_set_name. The family set name is used for the catalog file name. Alternatively, a file can be specified by the user in the Additional Parameters field on the library definition line in the mcf file.

     If the automated library is SCSI attached, the library catalog is a one-to-one mapping between the library catalog entries and physical slots in the automated library. However, if the automated library is network-attached, the library catalog is not a direct mapping to the slots, but it is a list of VSNs known to be present in the automated library.

     The library catalog contains the following information about each VSN in the library:

     o Status bits

     o Media type

     o Volume serial number

     o Storage slot

     o Partition

     o Count of access

     o Capacity of volume

```
     o  Space left on volume

     o  Block size or sector size for optical media

     o  Label time

     o  Last modification time

     o  Last mount time

     o  Bar Code

     o  First word address of PTOC (for optical media) or last
        position found (for tape media).

     If reserved VSNs are used, the following fields are also
     present:

     o  Time reservation made

     o  Archive set

     o  Owner

     o  File system

SEE ALSO
     build_cat(1M), dump_cat(1M), export(1M), import(1M).

     mcf(4).
```

sam-clfsd(1M)

```
NAME
     sam-clfsd - SAM-QFS shared file system client daemon

SYNOPSIS
     sam-clfsd [ -d ] [ -f fsname ] [ -h ] [ -i fsname ] [ -l ] [
     -u fsname ] [ -w ] [ dev...  ]

AVAILABILITY
     SUNWclqfs

DESCRIPTION
     sam-clfsd loads  the  samfs  and  samioc  modules  into  the
     operating system if they are not already loaded, and updates
     or reports on configuration  information.   The  options  to
     sam-clfsd are:

     -h        Print out a short usage message and exit.

     -l        Report the names of all configured file systems.

     -f  fsname
               Compare the named file system with the listed  dev
               ... devices, and report any discrepancies, such as
               missing partitions, partitions that  don't  belong
               to the named file system, etc..

     -i  fsname
               Verify that the listed dev  ...  devices  comprise
               the  slices  of the named file system, and install
               and configure the file system.

     -u  fsname
               Uninstall the named file system from the  system's
               configured  file  systems.  This command will fail
               on a mounted file system.

     -d        Start up the sam-sharefsd daemon for the specified
               file  system.   Useful  only  with  the -i option.
               After installing the file system, sam-clfsd  forks
               off  a  child that starts up the sam-sharefsd dae-
               mon.  If the daemon exits with a non-fatal  error,
               the  restarting  it  as  necessary.   The  command
               itself returns.

     -w        Causes the sam-clfsd  program  to  await  a  fatal
               error  from  the  sam-sharefsd  daemon  instead of
               returning immediately.  Useful only  with  the  -i
               and -d options.

     sam-clfsd must be run as root.
```

EXAMPLE

```
     Here's an example using sam-clfsd:

          Configure the file system:

     juniper# sam-clfsd -di shsam1 /dev/dsk/c4t50020F23000055A8d0s1 \
/dev/dsk/c4t50020F23000078F1d0s0 /dev/dsk/c4t50020F23000078F1d0s1
     FS 'shsam1' installed
     juniper# mount shsam1

     juniper# umount shsam1
     juniper# ps
        PID TTY      TIME CMD
        481 console  0:00 csh
       3722 console  0:00 sam-shar
       3721 console  0:00 sam-clfsd
       3727 console  0:00 ps
     juniper# kill 3721 3722
     juniper# sam-clfsd -u shsam1
```

SEE ALSO

```
     mount(1M) samfsconfig(1M)
```

sam-clientd(1M)

```
NAME
     sam-remote, sam-clientd, sam-serverd - Describes the Sun
     SAM-Remote interface and daemons

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-serverd mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-clientd mshmid pshmid equip

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The Sun SAM-Remote client and server software allows
     automated libraries to be shared among the Solaris systems
     in a SAM-QFS environment.  Sun SAM-Remote allows you to
     configure multiple storage clients that archive and stage
     files from a centralized optical and/or tape library.  This
     environment also allows you to make multiple archive copies
     on various media housed in multiple libraries.

DAEMONS
     The Sun SAM-Remote daemons, sam-serverd and sam-clientd,
     control Sun SAM-Remote.  The sam-robotsd daemon starts the
     sam-serverd and sam-clientd daemons.  The identifiers
     associated with these daemons are as follows:

     mshmid    The identifier of the master shared memory segment
               created by sam-amld.

     pshmid    The identifier of the preview shared memory
               segment created by sam-amld.

     equip     The equipment number of the device.

     For more information on the sam-robotsd or sam-amld daemons,
     see the sam-robotsd(1M) or sam-amld(1M) man pages.

CONFIGURATION
     Configuring the Sun SAM-Remote client and server software
     involves adding lines to the mcf file on both the system to
     be used as the Sun SAM-Remote client and on the system to be
     used as the Sun SAM-Remote server.

     In addition, a client configuration file must be created on
     the Sun SAM-Remote client, and a server configuration file
     must be created on the Sun SAM-Remote server.

     Each entry in mcf file can configure up to ten clients per
     server.  Use more mcf entries to configure more than ten
     clients.

Device and Network Interfaces                     sam-remote(7)

     In the mcf file, the Equipment Type field contains sc to
     define a Sun SAM-Remote client or ss to define a Sun SAM-
```

Remote server.

The server configuration file defines the disk buffer characteristics and media to be used for each client. For a client named portland for example:

```
portland
          media
          100 at (000031|000032|000034|000035|000037|000038)
          endmedia
```

The media definitions must be indented with white space or tab characters. The regex data must be enclosed by parentheses.

For a complete description of the Sun SAM-Remote configuration process, see the SAM-QFS Configuration and Administration Guide.

FILES

mcf	The master configuration file for SAM-QFS, Sun QFS, the Sun SAM-Remote client, and the Sun SAM-Remote server.
/opt/SUNWsamfs/lib/librmtsam.so	The Sun SAM-Remote shared object library.

SEE ALSO

sam-amld(1M), sam-robotsd(1M).

mcf(4).

SAM-QFS Configuration and Administration Guide.

sam-dbupd(1M)

NAME

sam-dbupd - SAM-QFS Updates the MySQL data base from events in the sam-fsalogd event log

SYNOPSIS

/opt/SUNWsamfs/sbin/sam-dbupd

AVAILABILITY

SUNWsamfs

DESCRIPTION

sam-dbupd continuously updates the SAM-QFS MySQL sideband database by reading events from the sam-fsalogd log file. sam-dbupd is initiated by sam-fsd.

sam-dbupd reads the fsalogd.cmd and creates an inventory file of paths to the fsalog files. The inventory file of unprocessed fsalog files is maintained over a samd stop/samd start or an umount/mount or a system panic. However, the inventory file does not contain absolute path names of the fsalog files. Therefore, when the location of fsalog files is changed via the fsalogd.cmd, any unprocessed fsalog files must be moved to the new location in order to be processed by sam-dbupd.

sam-dbupd reads events from the fsalog files and updates the sideband database accordingly. Events are marked when fully processed and completely processed fsalog files are removed from the inventory file when all events in the fsalog file have been completed.

SEE ALSO
samdb(1M) sam-fsalogd(1M) fsalogd.cmd(4M) samdb.conf(4)

sam-fsalogd(1M)

NAME
sam-fsalogd - Logs SAM-QFS file system activity

SYNOPSIS
```
/opt/SUNWsamfs/sbin/sam-fsalogd
```

AVAILABILITY
SUNWsamfs

DESCRIPTION
The sam-fsalogd daemon is initiated by the sam-fsd daemon. The sam-fsd daemon starts a file system activity daemon for each configured SAM-QFS file system.

The sam-fsalogd daemon opens a door to the SAM-QFS file system. The sam-fsalogd daemon receives events associated with this file system and logs them. Events include:

TABLE 1. SAM-QFS file system activity events

Event	Description	Parameter(s)	Time
ev_none	-	-	-
ev_create	file created	nlinks,namehash	creation time
ev_change	uid/gid changed	-	change time
ev_close	modified & closed	filemode	modify time
ev_rename	renamed	0=new,namehash	parent modify time
ev_rename	renamed	1=old,namehash	old parent modify time
ev_remove	removed	nlinks,namehash	change time
ev_offline	marked offline	-	residence time
ev_online	marked online	-	residence time
ev_archive	copy archived	copy number	copy creation time
ev_modify	copies stale	-	modify time
ev_archange	copies changed	copy number	modify time
ev_restore	file restored	0=old	old modify time
ev_restore	file restored	1=new	new modify time
ev_umount	umount fs	-	now

FILES
Detailed trace information is written to the sam-fsalogd

trace file.

In the fsalogd.cmd file, you can specify the directory pathname where the logs are stored for each SAM-QFS file system.

SEE ALSO

mount_samfs(1M). sam-fsd(1M).

fsalogd.cmd(4M). defaults.conf(4M).

sam-fsd(1M)

NAME

sam-fsd - Initializes Sun QFS and SAM-QFS environments

SYNOPSIS

```
/usr/lib/fs/samfs/sam-fsd [ -C  ]  [  -N  ]  [  -D  ]  [  -c
defaults] [ -d diskvols] [ -f samfs] [ -m mcf] [ -v ]
```

AVAILABILITY

SUNWsamfs

DESCRIPTION

sam-fsd initializes Sun QFS and SAM-QFS environments and performs tasks for the file system kernel code. These tasks include sending messages to syslog, and starting the archiver, releaser, shared fs, and stager daemons. It is managed as a service by the Solaris Service Management Facility smf(5)

When started, sam-fsd reads the configuration files defaults.conf, diskvols.conf, mcf, and samfs.cmd located in the directory /etc/opt/SUNWsamfs. These files may be changed at any time while sam-fsd is running. The changes will take place when sam-fsd is restarted, or sent the signal SIGHUP.

The file systems are configured and necessary daemons are started. Configuration parameters are set, and table files are written for use by other components of the SAM-QFS environment.

If errors occur in any of the configuration files, sam-fsd refuses to run and writes a notification message to syslog. The problem must be corrected, and the signal SIGHUP sent to sam-fsd. sam-fsd then rereads the configuration files. The syslog message contains the command necessary to signal sam-fsd .
 'kill -HUP sam-fsd-pid'

Trace Files

Several Sun QFS and SAM-QFS daemons write messages to trace files. These messages contain information about the state and progress of the work performed by the daemons. The messages are primarily used by Sun engineers and support personnel to improve performance and diagnose problems. As such, the message content and format are subject to change with bugfixes and feature releases.

The daemons writing trace files are: sam-archiverd, sam-catserver, sam-fsd, sam-rftd, sam-recycler, sam-sharefsd, and sam-stagerd.

To prevent the trace files from growing indefinitely, sam-fsd monitors the size and age of the trace files and periodically executes the script /opt/SUNWsamfs/sbin/trace_rotate. This script moves the trace files to sequentially numbered copies. The script is executed when the trace file exceeds a specified size, or age. The size and age are specified in defaults.conf. If /opt/SUNWsamfs/sbin/trace_rotate does not exist, sam-fsd performs no action.

OPTIONS

sam-fsd may be started by direct execution to provide detailed messages about problems in configuration files. In this case, the following options are allowed:

-c	defaults Sets an alternate defaults.conf file to check. defaults is the path to the alternate defaults configuration file.
-d	diskvols Sets an alternate diskvols.conf file to check. diskvols is the path to the alternate diskvols configuration file.
-f	fs_name Sets a single file system. fs_name is the family set name from the mcf file.
-m	mcf Sets an alternate mcf file to check. mcf is the path to the alternate mcf file.
-v	Sets verbose mode.
-C	Configure SAM-QFS if not already configured. Must be the only option.
-N	Exits with a non-zero status if any SAM-QFS file systems are configured. Used by the Solaris SMF facility. Must be the only option.
-D	Used by the SMF facility to start sam-fsd as a daemon. Used by the Solaris SMF facility. Must be the only option.

FILES

/etc/opt/SUNWsamfs	Location of SAM-QFS configuration files.
mcf	The configuration file for SAM-QFS environments.

```
     samfs.cmd           Sun QFS and SAM-QFS mount commands file.

     defaults.conf       Set default values for SAM-QFS  environ-
                         ment.
```

SEE ALSO

     defaults.conf(4), diskvols.conf(4), mcf(4), samfs.cmd(4).

     trace_rotate(1M).

sam-ftpd(1M)

NAME

     sam-ftpd - Renamed to "sam-rftd"

SEE ALSO

     sam-rftd(1M).

sam-genericd(1M)

NAME

     sam-robotsd, sam-genericd, sam-stkd, sam-ibm3494d, sam-sonyd
     - SAM-QFS media changer daemons

SYNOPSIS

```
     /opt/SUNWsamfs/sbin/sam-robotsd mshmid pshmid

     /opt/SUNWsamfs/sbin/sam-genericd mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-stkd mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-ibm3494d mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-sonyd mshmid pshmid equip
```

AVAILABILITY

     SUNWsamfs

DESCRIPTION

     The sam-robotsd daemon starts and monitors the execution of
     the media changer library control daemons for SAM-QFS. The
     sam-robotsd daemon is started automatically by the sam-amld
     daemon if there are any libraries defined in the mcf file.
     The sam-robotsd daemon starts and monitors the correct
     daemon for all defined libraries. For more information on
     the mcf file, see the mcf(4) man page.

     Each library daemon is responsible for monitoring the
     preview table for the VSNs that are controlled by that
     daemon. If a request is found for one of its VSNs, the
     daemon finds an available drive under its control and moves
     the cartridge into that drive. When the device is ready,
     the daemon notifies the SAM-QFS library daemon, and the

device is assigned to the waiting process.

The identifiers are as follows:

mshmid	The identifier of the master shared memory segment created by the sam-amld daemon.
pshmid	The identifier of the preview shared memory segment created by the sam-amld daemon.
equip	The equipment number of the device.

The sam-genericd daemon controls libraries that conform to the SCSI II standard for media changers, and it is the daemon that controls the ADIC/Grau ABBA library through the grauaci interface. For more information on this interface, see the grauaci(7) man page.

The sam-stkd daemon controls StorageTek libraries through the ACSAPI interface and is included in the SAM-QFS software package. For more information on this interface, see the stk(7) man page.

The sam-ibm3494d daemon controls IBM 3494 tape libraries through the lmcpd interface and is included in the SAM-QFS software package. For more information on this interface, see the ibm3494(7) man page.

The sam-sonyd daemon controls Sony libraries through the Sony DZC-800S PetaSite Application Interface Library and is included in the SAM-QFS software package. For more information on this interface, see the sony(7) man page.

FILES

mcf	The master configuration file for SAM-QFS environments.

SEE ALSO

sam-amld(1M).

mcf(4).

acl2640(7), acl452(7), grauaci(7), ibm3494(7), ibm3584(7), sam-remote(7), sony(7), stk(7).

sam-grau_helper(1M)

NAME

grauaci - The ADIC/Grau Automated Tape Library through the ACI

AVAILABILITY

SUNWsamfs

DESCRIPTION

grauaci is the Sun QFS and SAM-QFS software interface to the

ADIC/Grau Network-attached library. This interface utilizes the DAS/ACI 3.10E interface supplied by ADIC. For more information on DAS/ACI, see the DAS/ACI 3.10E Interfacing Guide and the DAS Administration Guide. Both manuals are supplied by ADIC.

CONFIGURATION

Sun assumes that your site has the DAS server configured and operating with the ADIC/Grau library. In the DAS configuration file for this client, the avc (avoid volume contention) and the dismount parameters should both be set to true.

The Equipment Identifier field in the mcf file is the full path name to a parameters file used by grauaci. This file consists of a list of keyword = value pairs or a keyword followed by a drivename = value pair. For more information on the mcf file, see the mcf(4) man page.

All keywords and values, including the following, are case sensitive and must be entered as shown:

Keyword	Value
client	This is the name of this client as defined in the DAS configuration file. This is a required parameter.
server	This is the hostname of the server running the DAS server code. This is a required parameter.
acidrive	There is one acidrive line for every drive assigned to this client. Following the acidrive keyword is a drivename = path, string that is as follows:

drivename	The drive name as configured in the DAS configuration file.
path	The path name to the device. This name must match the Equipment Identifier of an entry in the mcf file.

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

EXAMPLE

The following example shows sample parameters files and mcf entries for a ADIC/Grau library supporting DLT tape and HP optical drives. The catalog files are placed in the default directory, which is /var/opt/SUNWsamfs/catalog.

```
#
# This is file: /etc/opt/SUNWsamfs/gr50
#
```

```
client = grau50
server = DAS-server
#
# the name "drive1" is from the DAS configuration file
#
acidrive drive1 = /dev/rmt/0cbn         # a comment
#
# the name "drive2" is from the DAS configuration file
#
acidrive drive2 = /dev/rmt/1cbn         # a comment

#
# This is file: /etc/opt/SUNWsamfs/gr60
#
client = grau60
server = DAS-server
#
# the name "DH03" is from the DAS configuration file
#
acidrive DH03 = /dev/samst/c1t1u0
```

The mcf file entries.

```
#
# Sample mcf file entries for an ADIC/Grau library - DLT
#
/etc/opt/SUNWsamfs/gr50  50      gr      gr50   -  gr50cat
/dev/rmt/0cbn           51      lt      gr50   -  /dev/samst/c2t5u0
/dev/rmt/1cbn           52      lt      gr50   -  /dev/samst/c2t6u0

#
# Sample mcf file entries for an ADIC/Grau library - HP optical
#
/etc/opt/SUNWsamfs/gr60  60      gr      gr60   -  gr60cat
/dev/samst/c1t1u0       61      od      gr60   -
```

IMPORT/EXPORT

The physical adding and removing of cartridges in an ADIC/Grau network-attached library is accomplished using the DAS utilities. The import(1M) and export(1M) commands affect only the library catalog. Therefore, importing and exporting cartridges with the ADIC/Grau network-attached library consists of the following two-step process:

1) Physically import or export the cartridge using the DAS utilities.

2) Virtually update the automated library catalog using the Sun QFS or SAM-QFS import and export utilities.

The import(1M) command has an optional -v parameter for supplying the VSN to be added. The grauaci interface verifies that DAS knows about the VSN before updating the catalog with the new entry. The export(1M) command removes the entry from the catalog. For more information on importing and exporting, see the import and export(1M) man pages.

CATALOG

There are several methods for building a catalog for an

ADIC/Grau network-attached library. You should use the method that best suits your system configuration, and this is typically determined by the size of the catalog that is needed.

Method 1: Create a catalog with existing VSN entries. (Please note this method only works for tapes. It does not work for barcoded optical media.) You can build a catalog that contains entries for many tapes by using the build_cat(1M) command. As input to build_cat(1M), you need to create a file that contains the slot number, VSN, barcode, and media type. For example, file input_vsns follows:

```
     0 TAPE01  TAPE01    lt
     1 TAPE02  TAPE02    lt
     2 TAPE03  TAPE03    lt
```

The input_vsns file can be used as input to the build_cat(1M) command, as follows:

```
    build_cat input_vsns /var/opt/SUNWsamfs/grau50cat
```

Method 2: Create a null catalog and import VSN entries. You can create an empty catalog and populate it. To create a catalog that will accommodate 1000 slots, use the build_cat command, as follows:

```
      build_cat -s 1000 /dev/null /var/opt/SUNWsamfs/catalog/grau50cat
```

Use the import(1M) command to add VSNs to this catalog, as follows:

```
      import -v TAPE01 50
```

For ADIC/Grau optical media, it is very important to import the A side of barcoded optical media. The Sun QFS and SAM-QFS software queries the ADIC/Grau database to find the barcode for the B side and fills in the catalog entry for the B side appropriately. The A side of optical media in the ADIC/Grau automated library is the left side of a slot as you face the slots.

Method 3: Use the default catalog and import VSN entries. If a catalog path name is not specified in the mcf file, a default catalog is created in /var/opt/SUNWsamfs/catalog/family_set_name when the Sun QFS or SAM-QFS software is initialized. Following initialization, you must import VSN entries to this catalog. Use the import(1M) command, as follows:

```
      import -v TAPE01 50
```

In the preceding import(1M) command, 50 is the Equipment Identifier of the automated library as specified in the mcf file.

FILES

mcf	The configuration file for the Sun QFS and SAM-QFS software.
/opt/SUNWsamfs/lib/libaci.so	

```
                              The ACI library supplied by ADIC.

SEE ALSO
     build_cat(1M), dump_cat(1M),  export(1M),  import(1M),  sam-
     robotsd(1M).

     mcf(4).
```

sam-ibm3494d(1M)

```
NAME
     sam-robotsd, sam-genericd, sam-stkd, sam-ibm3494d, sam-sonyd
     - SAM-QFS media changer daemons

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-robotsd mshmid pshmid

     /opt/SUNWsamfs/sbin/sam-genericd mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-stkd mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-ibm3494d mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-sonyd mshmid pshmid equip

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-robotsd daemon starts and monitors the execution of
     the media changer library control daemons for SAM-QFS.  The
     sam-robotsd daemon is started automatically by the sam-amld
     daemon if there are any libraries defined in the mcf file.
     The sam-robotsd daemon starts and monitors the correct
     daemon for all defined libraries.  For more information on
     the mcf file, see the mcf(4) man page.

     Each library daemon is responsible for monitoring the
     preview table for the VSNs that are controlled by that
     daemon.  If a request is found for one of its VSNs, the
     daemon finds an available drive under its control and moves
     the cartridge into that drive.  When the device is ready,
     the daemon notifies the SAM-QFS library daemon, and the
     device is assigned to the waiting process.

     The identifiers are as follows:

     mshmid    The identifier of the master shared memory segment
               created by the sam-amld daemon.

     pshmid    The identifier of the preview shared memory
               segment created by the sam-amld daemon.

     equip     The equipment number of the device.

     The sam-genericd daemon controls libraries that conform to
```

the SCSI II standard for media changers, and it is the daemon that controls the ADIC/Grau ABBA library through the grauaci interface. For more information on this interface, see the grauaci(7) man page.

The sam-stkd daemon controls StorageTek libraries through the ACSAPI interface and is included in the SAM-QFS software package. For more information on this interface, see the stk(7) man page.

The sam-ibm3494d daemon controls IBM 3494 tape libraries through the lmcpd interface and is included in the SAM-QFS software package. For more information on this interface, see the ibm3494(7) man page.

The sam-sonyd daemon controls Sony libraries through the Sony DZC-800S PetaSite Application Interface Library and is included in the SAM-QFS software package. For more information on this interface, see the sony(7) man page.

FILES

mcf	The master configuration file for SAM-QFS environments.

SEE ALSO

sam-amld(1M).

mcf(4).

acl2640(7), acl452(7), grauaci(7), ibm3494(7), ibm3584(7), sam-remote(7), sony(7), stk(7).

sam-nrecycler(1M)

NAME

sam-nrecycler - Recycles SAM-QFS volumes

SYNOPSIS

```
/opt/SUNWsamfs/sbin/sam-nrecycler [-n]
```

AVAILABILITY

SUNWsamfs

DESCRIPTION

The sam-nrecycler command invokes the nrecycler. The nrecycler removes expired archive copies and frees up archive volumes. Often, the nrecycler is invoked through root’s crontab(1) file at an off-peak time. However, the nrecycler can be invoked at any time.

The sam-nrecycler command provides additional support to aid in the ability to use SAM-QFS dump files for SAM-QFS archive retention capabilities. The nrecycler will scan file system metadata and SAM-QFS dump files to determine which removable media and disk archive volumes contain archive images so space on unused volumes can be reclaimed. The nrecycler will identify all the archive images present on a removable media volume or disk archive tar ball by scanning all file system .inodes files and specified SAM-QFS dump files. By scanning the file systems and SAM-QFS dump files, the nrecycler can determine if there are volumes which do not contain any archive images and the space on these volumes can be reclaimed. If a removable media volume does not contain any archive images, it is safe to relabel the cartridge. If a disk archive tar ball does not contain any archive images, it is safe to remove the tar ball from the disk archive directory.

You must provide directives to the nrecycler through lines entered in the /etc/opt/SUNWsamfs/nrecycler.cmd file. User must specify a path to directories containing all SAM-QFS dump files to be searched. If no directories are specified in the command file, recycling does not occur. The user is responsible for making sure the list of directories is complete and all SAM-QFS dump files are contained in the directory list. The nrecycler cannot validate the SAM-QFS dump file list. All removable media and disk volumes are eligible to be selected as obsolete, and thus eligible to be relabeled or unlinked.

After the nrecycler detects that a removable media volume contains only free and expired space, thus it is safe to relabel, the nrecycler invokes the sam-nrecycler.sh script. The script can relabel the cartridge using either the original VSN or a new VSN; or it can export the cartridge from the library; or it can perform another user-defined

action.

After the nrecycler detects that a disk archive volume contains only free and expired space, the nrecycler will unlink the unused disk archive tar ball.

OPTIONS

This command accepts the following options:

-n   Prevents any actions from being taken.

OPERATION

The sam-recycler command should not be used. The nrecycler will scan all file system .inodes files and specified SAM-QFS dump files. Since sam-recycler only will scan file system .inodes files it will incorrectly reclaim space on archive volumes that has space occupied by archive copies in the SAM-QFS dump files.

You must have the nrecycler command enabled by setting the

```
nrecycler = yes
```

option in the /etc/opt/SUNWsamfs/defaults.conf file.

The nrecycler is designed to run periodically. It performs as much work as it can each time it is invoked. Between executions, the nrecycler keeps SAM-QFS dump file information in a nrecycler dat file.

All files in SAM-QFS dump directories must be valid SAM-QFS dump files. Hidden files, files that begin with a dot, are skipped. During the first scan of a dump, the nrecycler will create a dat file. The nrecycler dat file will be created in the same directory as the dump file with the string 'SUNWsamfs' appended to the original dump file's name. A nrecycler dat file contains a summary of which removable media and and disk archive volumes contain archive images for the dump. This is a nrecycler performance optimization so the dump file does not need to be reread during every execution of the nrecycler. If a SAM-QFS dump should no longer be processed, the nrecycler's dat file for the file must be removed from the dump directory.

All removable media and disk archive volumes will be examined and must be owned by this instantiation of SAM. The nrecycler should not be used in a SAM-remote environment. However, if disk archive volumes are not shared between servers, the nrecycler will work correctly on disk volumes that are reside on other machines.

The nrecycler checks to see if there are removable media

volumes that were selected for recycling that have not yet been post-processed. If such volumes exist, and they are now devoid of active archive copies, the sam-nrecycler command invokes the /etc/opt/SUNWsamfs/scripts/nrecycler.sh(1M), which post-processes these volumes with arguments including the generic media type (tp or od), the VSN, the element address in the library, and the equipment number of the library in which the volume resides. The script can relabel the cartridge using either the original VSN or a new VSN; or it can export the cartridge from the library; or it can perform another user-defined action. The nrecycler.sh script will not be invoked if the amount of space used on a removable media volume is less than 50% of total space available on the volume.

Each time it is run, the nrecycler performs these steps:

1. Build a list of all removable media and disk archive volumes configured in SAM-QFS. For faster searching, a hash table will be used to hold volume information.

2. Collect a list of all file systems configured in SAM-QFS. All SAM-QFS file systems, or for which we are the metadata server, must be mounted to allow the .inodes file to be read.

3. Generate a list of specified SAM-QFS dump directories. Initialize samfsdump file processing by walking each of the specified directories and validating the contents of every file. Every file in the directory must be a valid samfsdump file or a nrecycler dat file must exist for a dump file.

4. Scan file systems' .inode file reading each inode in all

```
     file systems.  For each archive copy, the VSN on which the
     copy resides is accumulated into the VSN table.

     5. Scan all SAM-QFS dump files reading each inode in all
     dump files.  For each archive copy, the VSN on which the
     copy resides is accumulated into the VSN table.  During the
     first scan of a dump, the nrecycler will create a dat file.
     Subsequent execution of the nrecycler will use VSN summary
     information from the dat file.

     6. Depending on the disk archives' maximum sequence number,
     multiple file system .inodes and SAM-QFS dump file scans may
     be necessary.

     7. Select removable media and disk volumes that are obsolete
     and eligible to be relabeled or unlinked.

RECYCLER OUTPUT
     None.

SEE ALSO
     nrecycler.sh(1M).  nrecycler.cmd(4).
```

sam-recycler(1M)

```
NAME
     sam-recycler - Recycles SAM-QFS volumes

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-recycler [-b] [-c] [-C] [-d] [-E]
     [-n] [-s] [-t] [-v] [-V] [-x] [-X]
     [family_set | archive_set]

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-recycler command invokes the recycler.  The recycler
     removes expired archive copies and frees up archive volumes.
     Often, the recycler is invoked through root's crontab(1)
     file at an off-peak time.  However, the recycler can be
     invoked at any time.

     You can specify that only a specific library or archive set
     be recycled.  You can recycle by library only when archiving
     to tape or magneto optical cartridges in a library.  Note
     that you cannot recycle by library if you are using disk
     archiving.

     If you want to recycle by archive set, you must name the
     archive sets to be recycled in the
     /etc/opt/SUNWsamfs/archiver.cmd file.

     You can provide directives to the recycler through lines
     entered in the /etc/opt/SUNWsamfs/recycler.cmd file and in
     the /etc/opt/SUNWsamfs/archiver.cmd file.  If no directives
```

are present and no family_set or archive_set is specified on the command line, recycling does not occur. The following are the default recycler settings:

- The maximum data quantity to recycle (-dataquantity) is 1 gigabyte (1G).
- The high water mark (-hwm) is 95.
- The VSN gain (-mingain) is 60 for volumes <200GB and 90 for volumes >=200GB.
- The number of volumes (-vsncount) to recycle is 1.
- Automatic email is not sent.

NOTE: Extreme care must be taken when configuring the recycler if you are using disk archiving in an environment with multiple SAM-QFS servers. The diskvols.conf file for each SAM-QFS server must point to a unique set of disk volume resource specifications (disk archiving target

directories). If any of these are shared between different SAM-QFS servers, then running the recycler from one SAM-QFS server will destroy the disk archive data that is being managed by the other SAM-QFS server.

OPTIONS

The following options determine the volumes to be recycled and the content of the recycler log file.

-b   Displays the capacity and remaining space for each volume in base 10 units in the recycler log file. By default, space is displayed in base 2 units.

-c   Displays the extrapolated capacity of each volume. This is the volume’s capacity assuming the compression observed on the volume so far continues for the rest of the volume. This option produces an additional line for each volume with the heading Alpha:.

-C   Suppresses listing of initial catalog(s).

-d   Displays messages during the volume selection phase of processing. These messages indicate why each volume was, or was not, selected for recycling.

-E   Specifies that the volume section of the recycler’s log file list only volumes that are not 100% free.

-n   Prevents any actions from being taken. This option causes /opt/SUNWsamfs/sbin/sam-recycler to behave as if -recycle_ignore were specified in the /etc/opt/SUNWsamfs/archiver.cmd file for all archive sets.

-s   Suppresses the listing of individual volumes in the initial catalog section.

```
          -t   Recycle tape volumes only.

          -v   Displays information about which files are resident on
               the volume that is marked for recycling.  If no path
               name can be calculated for the inode, it lists the
               inode.  These files are on volumes that are being
               drained.  Using this option can consume a lot of CPU
               cycles.

          -V   Suppresses the volume section in the listing.

          -x   Displays messages for expired archive copies.  These
               are copies that are older than the time the volume upon
               which the copies reside was labeled.  Such copies
               generate an error message when staged.  The data for

               those copies is irrecoverable.  These archive copies
               must be unarchived.  If any such copies are discovered,
               the recycler stops.  This is the default behavior.
               Also see the -X option.

          -X   Inhibits the messages that indicate the existance of
               expired archive copies.  Typically, if the recycler
               detects expired archive copies, it stops.  Use this
               options if you want the recycler to continue in the
               presence of expired archive copies.  Also see the -x
               option.

          family_set | archive_set
               Recycles only the named family_set or archive_set.
               This is an optional argument.  If a family_set is
               specified, the library associated with the family set
               is recycled.  The family set is the fourth field in a
               server's mcf file.  If an archive_set is specified,
               that archive set is recycled.  The archive_set
               specified must include the copy number, as stated in
               the /etc/opt/SUNWsamfs/archiver.cmd file.  For example,
               arset.1.

               If no family_set or archive_set name is specified, the
               recycler recycles according to specifications in the
               /etc/opt/SUNWsamfs/archiver.cmd and the
               /etc/opt/SUNWsamfs/recycler.cmd files.  It examines
               each library and archive set specified.

               Regardless of a specification, only archive sets and
               family sets that have a current usage that is less than
               the high-water mark are recycled.

     OPERATION
          The recycler splits its work into two phases:  volume
          selection and volume recycling.

          Phase 1 - Volume Selection
                    The recycler selects volumes for recycling based
                    on the amount of space used by expired archive
                    copies as a percentage of total space on a volume.
                    For each library or archive set being recycled,
                    the volumes with the highest percentages of
                    expired copies are selected to bring the media
                    utilization in the library or archive set below
                    the configured high-water-mark.  This assumes that
                    each volume selected would contribute at least
                    VSN-minimum-percent-gain percent of its total
                    space if it were recycled.  If no such volumes
                    exist, the library or archive set cannot be
                    recycled.  Ties in expired space are resolved by
                    selecting the volumes with the least amount of

                    unexpired space.  For more information on setting
                    a high water mark, see the recycler.cmd(4) man
                    page.

                    A few conditions can prevent a volume from being
                    selected.  A volume cannot be recycled if it
                    contains data associated with a removable media
                    file created by the request(1) command.  In
                    addition, it cannot be recycled if it is listed in
                    the /etc/opt/SUNWsamfs/recycler.cmd file's
                    no_recycle section.

                    After volumes have been selected, they are
                    recycled.

          Phase 2 - Volume Recycling
                    Volume recycling differs depending upon whether
                    the archive media is a disk volume or whether it
                    is a removable cartridge in a library.  Archiving
                    to disk volumes is described first.

                    When a disk volume is selected for recycling, the
                    volume is not marked for recycling.  Additional
                    archive copies can be written to it.  Expired
                    archive copies on the disk volume are identified
                    and removed.  Valid archive copies are left alone.

                    When a tape or magneto optical volume is selected
                    for recycling, the system prevents additional
                    archive copies from being written to it.  If you
                    are recycling to cartridges in a library, all
                    files with active archive copies in volumes on the
                    cartridges are marked to be re-archived.  The
                    archiver moves these copies to other volumes.  In
                    subsequent runs, the recycler checks these volumes
                    and post-processes them when all valid archive
                    copies have been relocated.

                    The recycler checks to see if there are volumes
                    that were selected for recycling that have not yet
                    been post-processed.  If such volumes exist, and
                    they are now devoid of active archive copies, the
                    sam-recycler command invokes the
                    /etc/opt/SUNWsamfs/scripts/recycler.sh(1M), which
                    post-processes these volumes with arguments
                    including the generic media type (tp or od), the
                    VSN, the element address in the library, and the
                    equipment number of the library in which the
                    volume resides.  The script can relabel the
```

cartridge using either the original VSN or a new VSN; or it can export the cartridge from the library; or it can perform another user-defined

action.

The /etc/opt/SUNWsamfs/scripts/recycler.sh(1M) script clears the recycling flag to indicate that recycling has completed on the volume. The odlabel(1M) and tplabel(1M) commands clear this flag after the cartridge has been relabeled.

RECYCLER OUTPUT

The recycler log is divided into several sections.

The first section describes each library catalog and archive set. The header contains the family set name or archive set name and the vendor, product, and catalog path name. Then, the capacity and remaining space for each volume appears, in bytes, with suffixes k, M, G, and T representing kilobytes, megabytes, gigabytes, and terabytes, respectively. In this log file, a kilobyte=1024 bytes, a megabyte=1024*1024 bytes, and so on by default. If -b option is specified, the capacity and remaining space for each volume appears, in base 10 units. Then, a summary, containing the total capacity and total space remaining is shown in bytes and as a percentage of space used. The recycling parameters set in the recycler and archiver command files are also shown.

The second section is a series of tables, one for each library and archive set that has associated volumes. The name of the library or archive set is shown just to the right of the ----Percent---- label. A tape volume can be associated with only one physical library. But same as disk volumes it can belong to multiple archive sets. Attempts to assign a volume to multiple archive sets are marked with a in multiple sets label. The following fields are displayed:

Field Name	Meaning
Status	A phrase giving the volume’s recycle status, as follows:

empty VSN	The volume is empty of both expired and current archive images
full VSN	The volume has no free space, but it does have current archive images.
in multiple sets	The volume matches multiple archive sets in the /etc/opt/SUNWsamfs/archiver.cmd file.
new candidate	The volume was chosen for recycling during this recycler

```
                                 run.

                   no-data VSN    The volume contains only
                                  expired archive images and
                                  free space.

                   no_recycle VSN The volume is listed in the
                                  no_recycle section of the
                                  /etc/opt/SUNWsamfs/recycler.cmd
                                  file.

                   archive -n files
                                  The volume contains archive
                                  images for files now marked as
                                  archive -n.

                   old candidate  The volume was already marked
                                  for recycling before this
                                  recycler run.

                   request files  The volume contains archive
                                  images for removeable media
                                  files.

                   partially full The volume contains both
                                  current archive images and
                                  free space.

                   shelved VSN    The volume is not currently
                                  located in any library.

     Archives Count The number of archive copies that are
                    contained on this volume.

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

     Percent Use    The percentage of space in use on this volume
                    by current archive copies.  It is estimated
                    by summing up the sizes of the archive copies
                    on the medium.  Because of compression, this
                    value can overstate the amount of space
                    actually used by these images.  This is the
                    amount of data that would need to be moved if
                    the volume were selected for recycling.

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

                    recycling this cartridge.

                    The Percent Obsolete value is calculated as
                    follows:

                    100% - In Use - Free

                    Because In Use can overstate the actual space
```

```
               used (because of compression), the sum of In
               use + Free can exceed 100%, which renders
               Percent Obsolete to be a negative value.
               Although aesthetically unpleasing, this does
               not cause any problems in the operation of
               the recycler.

Percent Free   The percentage of free space remaining on
               this volume.  This value comes directly from
               the library catalog.  It gives the percent of
               the volume's total capacity that is available
               to hold new archive images.
```

For media that supports data compression, a best-guess value of the average compression is calculated from the ratio of the number of physical tape blocks consumed on the volume (that is, the difference of capacity - space) to the logical number of tape blocks written to the volume. The latter value is kept in the catalog. This ratio is then used to adjust the In Use value before it is written to the log file.

The first volume to appear in the log file, for each library or archive set, is the one most in need of recycling.

Here is an example recycler log file:

```
========== Recycler begins at Thu Feb  5 13:40:20 1998 ===========
3 catalogs:

0  Family: hy                   Path: /tmp/y
   Vendor: SAM-FS               Product: Historian
   EA                  ty    capacity         space vsn
      (no VSNs in this media changer)
   Total Capacity:  0    bytes, Total Space Available: 0    bytes
   Media utilization 0%, high 0% VSN_min 0%

1  Family: ad40                 Path: /var/opt/SUNWsamfs/catalog/ad40
   Vendor: ADIC                 Product: Scalar DLT 448
   EA                  ty    capacity         space vsn
      0                lt        19.2G         0    DLT3

      1                lt        17.7G        17.6G DLT4N
      5                lt        17.7G        17.6G DLT6
   Total Capacity:  54.6G bytes, Total Space Available: 35.2G bytes
   Media utilization 35%, high 75% VSN_min 50%

2  Family: arset0.1             Path: /etc/opt/SUNWsamfs/archiver.cmd
   Vendor: SAM-FS               Product: Archive set
   EA                  ty    capacity         space vsn
      0                lt         0            0    DLT5
      1                lt        19.2G         0    DLT3
      2                lt         0            0    DLT2
      3                lt        17.7G        17.6G DLT4N
      4                lt        17.7G        17.6G DLT6
   Total Capacity:  54.6G bytes, Total Space Available: 35.2G bytes
   Media utilization 35%, high 80% VSN_min 50%
   Send mail to root when this archive set needs recycling.
```

```
6 VSNs:

                    ---Archives---   -----Percent-----
-----Status-----    Count    Bytes   Use Obsolete Free   Library:Type:VSN
shelved VSN          677    648.9M                       <none>:lt:DLT0

                    ---Archives---   -----Percent-----   arset0.1
-----Status-----    Count    Bytes   Use Obsolete Free   Library:Type:VSN
no-data VSN            0      0        0   100      0    ad40:lt:DLT3
empty VSN              0      0        0     0      0  (NULL):lt:DLT2
empty VSN              0      0        0     0    100    ad40:lt:DLT6
full VSN               4     32.1k     0     0      0  (NULL):lt:DLT5
partially full         4     40.8k     0     0    100    ad40:lt:DLT4N

Recycler finished.

========== Recycler ends at Thu Feb  5 13:40:41 1998 ===========
```

Here is the corresponding archiver.cmd file:

```
interval = 2m
no_archive .
fs = samfs1
arset0 testdir0
     1 1s
     2 1s
     3 1s
     4 1s

no_archive .
fs = samfs2
no_archive .
vsns
arset0.1 lt DLT3 DLT4N DLT6 DLT1
arset0.2 lt DLT3 DLT4N DLT6 DLT1
arset0.3 lt DLT3 DLT4N DLT6 DLT1
arset0.4 lt DLT3 DLT4N DLT6 DLT1
samfs1.1 lt DLT3
samfs2.1 lt DLT4N
endvsns
params
arset0.1 -drives 4 -recycle_hwm 80 -recycle_mingain 50
endparams
```

Here is the corresponding /etc/opt/SUNWsamfs/recycler.cmd file:

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

RECYCLING HISTORIAN CARTRIDGES

The recycler recycles volumes listed in the historian's catalog. The volumes listed in the historian catalog have been exported from a library or have been or are currently in a manually-mounted device.

The /etc/opt/SUNWsamfs/scripts/recycler.sh(1M) script is passed the name hy, signifying volumes that reside in the historian catalog so that it can cope with the possibility of the volumes being recycled residing in an off-site storage facility. Typically, the /etc/opt/SUNWsamfs/scripts/recycler.sh(1M) script sends email to the administrator when this occurs to remind the administrator to bring the off-site volume back on site so that it can be reused. Volumes do not need to be on site to be drained of archive copies unless such a volume contains the only available archive copy of an off-line file.

RECYCLING BY ARCHIVE SET

When the recycler recycles by archive set, it treats each archive set as a small library that holds just the volumes assigned to the archive set in the /etc/opt/SUNWsamfs/archiver.cmd file. The volumes that are identified as belonging to a recycling archive set are removed from the recycler's version of the catalog for the library that physically contains the volume. Thus, only the volumes that are not part of an archive set remain in the library catalog.

To enable recycling for a given archive set, it must have one of the recycling options specified in the /etc/opt/SUNWsamfs/archiver.cmd file. For more information, see the archiver.cmd(4) man page.

MESSAGES

Consider the following message:

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

The preceding message means that the recycler could not set the rearchive flag for a file. When this happens, the recycler typically emits a message containing the path name, as follows:

```
Jan 22 10:17:17 jupiter sam-recycler[3400]: Cannot ioctl(F_IDSCF)
      /samfs1/testfile
```

However, in the first message, you see text beginning with Cannot find pathname.... This means that the recycler failed in its attempt to convert the inode number (in the preceding example message, it is inode number 406) and generation number (here, 25) into a path name in the /samfs1 file system.

The most likely reason for this to occur is that the file was deleted between the time that the recycler determined it needed to be rearchived and the time the recycler actually issued the system call to set the rearchive flag.

SEE ALSO

chmed(1M), odlabel(1M), recycler.sh(1M). sam-archiverd(1M), tplabel(1M).

archiver.cmd(4), mcf(4), recycler.cmd(4).

sam-releaser(1M)

```
NAME
     sam-releaser - SAM-QFS disk space releaser process

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-releaser file_system  low_water_mark
     weight_size [weight_age]

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-releaser process  controls  the  activities  of  the
     SAM-QFS releaser. The releaser makes disk cache available by
     identifying archived files and releasing  their  disk  cache
     copy.   This  process  is  started automatically by the file
     system when disk cache utilization  reaches  the  high-water
     mark.

     If   the   releaser   command   file   is   present   in
     /etc/opt/SUNWsamfs/releaser.cmd,  the  sam-releaser  process
     reads that file.  Directives in the  releaser.cmd  file  are
     overridden  by  the  equivalent  command-line  arguments, if
     present.  For more information on the releaser command file,
     see the releaser.cmd(4) man page.

OPTIONS
     This command accepts the following arguments:

     file_system This is the file system whose disk space  is  to
                 be  released.   The  argument  may be either the
                 name of the file  system,  or  its  mount_point.
                 The  releaser attempts to release the disk space
                 of archived files on the file system mounted  on
                 the mount_point until low_water_mark is reached.

     low_water_mark
                 A percentage of the file system that is  allowed
                 to  be  completely  occupied  with  files at all
                 times.  Specify an integer  number  that  is  at
                 least  0  but  no  more  than 100.  The releaser
                 attempts to release disk space  until  the  file
                 system is at or below this threshold.

     weight_size A weighting factor that is  used  to  prioritize
                 release  candidates.   Specify  a floating-point
                 value that is at least 0.0 but no more than 1.0.
                 For  more  information  on  weight_size, see the
                 PRIORITY WEIGHTS section of this man page.

     weight_age  A weighting factor that is  used  to  prioritize
                 release  candidates.   Specify  a floating-point
                 value that is at least 0.0 but no more than 1.0.

                 For  more  information  on  weight_age,  see the
                 PRIORITY WEIGHTS section of this man page.
```

ALGORITHM

The releaser reads the SAM-QFS .inodes file and builds an ordered list of the files that can be released. The position of each file on the list depends on a priority calculated for each inode by the releaser (see the PRIORITY WEIGHTS section of this man page.) Only the top list_size files are kept on the list. The default list_size is 30000 for file systems with less than 1M inodes, and 100000 with more than 1M inodes. See releaser.cmd(4) for a description of list_size.

Starting with the file with the numerically largest priority, the disk space used by each file is released until the low_water_mark has been reached. If the list is exhausted before the low_water_mark is reached, the process is repeated. If, while repeating the process, no files are found that can be released, the releaser stops. If the file system is still above high-water mark, the file system restarts the releaser.

PRIORITY WEIGHTS

Each inode is assigned a priority based on its size and age. The size of the file (expressed in units of 4-kilobyte blocks) is multiplied by the weight_size parameter. This result is added to the priority calculated for the age of the file to form the file’s final priority.

The releaser can use one of the following two methods for determining the contribution of the age of a file to the file’s release priority:

- o The first method is to take the most recent of the file’s access, modification, and residence-change age and multiply by weight_age.

- o The second method allows specification of weights for each of the access, modification, and residence-change times. These are specified by the weight_age_access=float, weight_age_modify=float, and weight_age_residence=float directives, respectively, in the releaser.cmd file. The sum of the product of the weight and corresponding age is the contribution of the age to the file’s priority. To specify any of these priority weights, you must use the releaser.cmd file. For information on the releaser.cmd file, see the releaser.cmd(4) man page.

For both methods, the ages are expressed in minutes.

LOG

Within the releaser.cmd file, you can specify a log file for each SAM-QFS file system. If the releaser.cmd file does not exist, or if no logfile=filename directive exists in the file, no logging occurs. For more information on the logfile=filename directive, see the releaser.cmd(4) man page.

The releaser creates the log file (if it does not exist) and appends the following to it for each run:

```
Releaser begins at Tue Sep 29 15:31:15 1998
inode pathname          /sam1/.inodes
low-water mark          40%
list_size               10000
weight_size             1
weight_age              0.5
fs equipment number     1
family-set name         samfs1
started by sam-fsd?     no
release files?          no
release rearch files?   yes
display_all_candidates? no
---before scan---
blocks_now_free:     117312
lwm_blocks:          233750
---scanning---
64122.5 (R: Tue Sep 29 11:33:21 CDT 1998) 237 min, 64004 blks S0 /sam1/250m
5131.5 (R: Tue Sep 22 17:39:47 CDT 1998) 9951 min, 156 blks S0 /sam1/filecq
5095.5 (R: Tue Sep 22 17:39:49 CDT 1998) 9951 min, 120 blks S0 /sam1/filecu
5062 (R: Tue Sep 22 18:38:50 CDT 1998) 9892 min, 116 blks S0 /sam1/filebz
5039.5 (R: Tue Sep 22 17:40:01 CDT 1998) 9951 min, 64 blks S0 /sam1/filedi
5036.5 (R: Tue Sep 22 17:37:34 CDT 1998) 9953 min, 60 blks S0 /sam1/fileio
5035.5 (R: Tue Sep 22 17:40:13 CDT 1998) 9951 min, 60 blks S0 /sam1/filedw
5032.5 (R: Tue Sep 22 17:38:08 CDT 1998) 9953 min, 56 blks S0 /sam1/filejq
5031.5 (R: Tue Sep 22 17:39:56 CDT 1998) 9951 min, 56 blks S0 /sam1/fileda
5024.5 (R: Tue Sep 22 17:38:00 CDT 1998) 9953 min, 48 blks S0 /sam1/filejh
5024 (R: Tue Sep 22 17:38:22 CDT 1998) 9952 min, 48 blks S0 /sam1/fileka
5023.5 (R: Tue Sep 22 17:40:07 CDT 1998) 9951 min, 48 blks S0 /sam1/filedn
5019 (R: Tue Sep 22 17:40:44 CDT 1998) 9950 min, 44 blks S0 /sam1/filefk
5015 (R: Tue Sep 22 17:40:28 CDT 1998) 9950 min, 40 blks S0 /sam1/fileep
5011.5 (R: Tue Sep 22 17:40:14 CDT 1998) 9951 min, 36 blks S0 /sam1/filedx
5011.5 (R: Tue Sep 22 17:39:58 CDT 1998) 9951 min, 36 blks S0 /sam1/filede
5011 (R: Tue Sep 22 17:41:07 CDT 1998) 9950 min, 36 blks S0 /sam1/filegk
5007.5 (R: Tue Sep 22 17:39:51 CDT 1998) 9951 min, 32 blks S0 /sam1/filecw
5007 (R: Tue Sep 22 17:41:10 CDT 1998) 9950 min, 32 blks S0 /sam1/filegr
5007 (R: Tue Sep 22 17:40:42 CDT 1998) 9950 min, 32 blks S0 /sam1/filefg
5007 (R: Tue Sep 22 17:40:30 CDT 1998) 9950 min, 32 blks S0 /sam1/filees
5004.5 (R: Tue Sep 22 17:38:14 CDT 1998) 9953 min, 28 blks S0 /sam1/filejv
5004 (R: Tue Sep 22 17:38:57 CDT 1998) 9952 min, 28 blks S0 /sam1/filelm
5002 (R: Tue Sep 22 18:38:54 CDT 1998) 9892 min, 56 blks S0 /sam1/filecd
4996.5 (R: Tue Sep 22 17:38:06 CDT 1998) 9953 min, 20 blks S0 /sam1/filejp

4995.5 (R: Tue Sep 22 17:39:57 CDT 1998) 9951 min, 20 blks S0 /sam1/filedc
4992.5 (R: Tue Sep 22 17:37:24 CDT 1998) 9953 min, 16 blks S0 /sam1/fileig
4992 (R: Tue Sep 22 17:39:06 CDT 1998) 9952 min, 16 blks S0 /sam1/filelv
4986 (R: Tue Sep 22 18:38:50 CDT 1998) 9892 min, 40 blks S0 /sam1/fileca
4982 (R: Tue Sep 22 17:36:54 CDT 1998) 9954 min, 5 blks S0 /sam1/filehk
4981 (R: Tue Sep 22 17:41:09 CDT 1998) 9950 min, 6 blks S0 /sam1/filegn
4980.5 (R: Tue Sep 22 17:40:15 CDT 1998) 9951 min, 5 blks S0 /sam1/filedz
---after scan---
blocks_now_free:       0
blocks_freed:          65452
lwm_blocks:            233750
archnodrop: 0
already_offline: 647
damaged: 0
extension_inode: 0
negative_age: 0
nodrop: 0
```

```
     not_regular: 7
     number_in_list: 32
     rearch: 1
     released_files: 32
     too_new_residence_time: 0
     too_small: 1
     total_candidates: 32
     total_inodes: 704
     wrong_inode_number: 14
     zero_arch_status: 3
     zero_inode_number: 0
     zero_mode: 0
     CPU time: 0 seconds.
     Elapsed time: 1 seconds.

     Releaser ends at Tue Sep 29 15:31:16 1998
```

The first block of lines shows the arguments with which the releaser was invoked, the name of the .inodes file, the low-water mark, the size and age weight parameters, the equipment number of the file system, the family set name of the file system, whether the releaser was started by sam-fsd or by the command line, whether files should be released, and whether each inode should be logged as encountered.

The second block of lines begins with the heading ---before scan---. It shows the number of blocks currently free in the cache and the number that would be free if the file system were exactly at the low-water mark. The goal of the releaser is to increase blocks_now_free so that it is equal to or larger than lwm_blocks.

The third block of lines begins with the heading ---scanning---. This block lists the files released by the

releaser and contains information for each file in separate fields. The fields are as follows:

Field Number	Content
1	This field contains the release priority.
2	This field contains the date and time in the following format: (tag: date_and_time). The tag is either A for access, M for modify, or R for residency, depending on if the date that follows represents the access, modify or residency time. The date_and_time is the most recent of the three dates listed.
3	This field contains the age and size of the file. The age of the file is expressed in minutes. The size of the file is expressed in blocks. These two figures are multiplied by their respective weights and the sum taken to yield the release priority.
4	This field contains an S followed by the seg-

	ment number. This is the number of the segment that was released.
5	This field contains the full path name of the released file.

Note that if the weight_age_access=float, weight_age_modify=float or weight_age_residence=float directives are specified in the releaser.cmd file, these lines show only the priority, size, and pathname.

The fourth block of lines begins with the heading ---after scan---. This block shows the statistics accumulated by the releaser during the previous scan pass are shown. These statistics are as follows:

Statistic	Meaning
archnodrop	The number of inodes marked archnodrop. These files are never released because the archiver is trying to keep them in cache.
already_offline	The number of inodes that were offline.
damaged	The number of inodes marked as damaged.
extension_inode	The number of extension inodes found. Used by volume overflow.
negative_age	The number of inodes that had an age in the future. This is usually caused by personal computers with incorrect clock settings acting as NFS clients.
nodrop	The number of inodes marked with release -n. For more information on marking files as never release, see the release(1) man page.
not_regular	The number of inodes that were not regular files.
number_in_list	The number of inodes that were on the releaser's candidate list when the releaser was finished scanning.
rearch	The number of files with a copy marked for rearchiving.
released_files	The number of files released.
too_new_residence_time	The number of inodes whose residence-change time was within minimum residence age of the current time as specified on the min_residence_age=time directive in the releaser.cmd file.

too_small	The number of files that were too small to be released.
total_candidates	The number of inodes found that were viable candidates for releasing.
total_inodes	The total number of inodes scanned.
wrong_inode_number	The number of inodes whose inode number did not match their offset in the inode file. This is usually not a concern, but you should run samfsck(1M) to rescue any orphan inodes. If you have already run samfsck(1M) and this field remains nonzero, no further action is required. For more information on the samfsck(1M) command, see the samfsck(1M) man page.
zero_arch_status	The number of inodes that had no archive copies.
zero_inode_number	The number of inodes that had zero as their inode number.
zero_mode	The number of inodes that were unused.
CPU time	The number of CPU seconds used in the current scan.
Elapsed time	The number of wall-clock seconds used in the current scan.

NOTES

When a file is created, the residency age is set to the creation time. The residency age of a file must be at least the value set by the min_residence_age=time directive before the file is considered for release. This is to prevent a file which was recently staged in from being released. The default time is 10 minutes.

If the releaser selects a file as a release candidate, and immediately thereafter the file is accessed, the file might still be released by the file system even though the file has been recently accessed. This can happen because the file system only prohibits release of a file that is currently in use. It does not check the access age of the file again when it is released.

SEE ALSO

release(1).

mount_samfs(1M), samfsck(1M).

releaser.cmd(4).

sam-rftd(1M)

```
NAME
     sam-rftd - SAM-QFS file transfer server process (was sam-
     ftpd)

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-rftd

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-rftd process is the file transfer server process for
     transferring SAM-QFS files to and from a remote network
     site.  The sam-rftd process is initiated by the sam-fsd
     daemon.

     By default, the file transfer daemon uses the default
     behaviors described on the rft.cmd(4) man page.

FILES
     If the daemon’s command file is present in
     /etc/opt/SUNWsamfs/rft.cmd, the sam-rftd process reads that
     file.

SEE ALSO
     sam-fsd(1M).

     rft.cmd(4).
```

sam-robotsd(1M)

```
NAME
     sam-robotsd, sam-genericd, sam-stkd, sam-ibm3494d, sam-sonyd
     - SAM-QFS media changer daemons

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-robotsd mshmid pshmid

     /opt/SUNWsamfs/sbin/sam-genericd mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-stkd mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-ibm3494d mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-sonyd mshmid pshmid equip

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-robotsd daemon starts and monitors the execution of
     the media changer library control daemons for SAM-QFS.  The
     sam-robotsd daemon is started automatically by the sam-amld
```

daemon if there are any libraries defined in the mcf file. The sam-robotsd daemon starts and monitors the correct daemon for all defined libraries. For more information on the mcf file, see the mcf(4) man page.

Each library daemon is responsible for monitoring the preview table for the VSNs that are controlled by that daemon. If a request is found for one of its VSNs, the daemon finds an available drive under its control and moves the cartridge into that drive. When the device is ready, the daemon notifies the SAM-QFS library daemon, and the device is assigned to the waiting process.

The identifiers are as follows:

mshmid	The identifier of the master shared memory segment created by the sam-amld daemon.
pshmid	The identifier of the preview shared memory segment created by the sam-amld daemon.
equip	The equipment number of the device.

The sam-genericd daemon controls libraries that conform to the SCSI II standard for media changers, and it is the daemon that controls the ADIC/Grau ABBA library through the grauaci interface. For more information on this interface, see the grauaci(7) man page.

The sam-stkd daemon controls StorageTek libraries through the ACSAPI interface and is included in the SAM-QFS software package. For more information on this interface, see the stk(7) man page.

The sam-ibm3494d daemon controls IBM 3494 tape libraries through the lmcpd interface and is included in the SAM-QFS software package. For more information on this interface, see the ibm3494(7) man page.

The sam-sonyd daemon controls Sony libraries through the Sony DZC-800S PetaSite Application Interface Library and is included in the SAM-QFS software package. For more information on this interface, see the sony(7) man page.

FILES

mcf	The master configuration file for SAM-QFS environments.

SEE ALSO

sam-amld(1M).

mcf(4).

acl2640(7), acl452(7), grauaci(7), ibm3494(7), ibm3584(7), sam-remote(7), sony(7), stk(7).

sam-rpcd(1M)

```
NAME
     sam-rpcd - SAM-QFS RPC API server process

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-rpcd

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     sam-rpcd is the RPC API (Application  Programmer  Interface)
     server process.  It is initiated by sam-amld.

     sam-rpcd uses the RPC program number that is paired with the
     RPC  program  name  samfs.   sam-rpcd  must  run on the same
     machine as the SAM-QFS file system.  You need  to  make  the
     following entry in /etc/services on the server:

     samfs       5012/tcp       # SAM-QFS 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)
```

sam-scannerd(1M)

```
NAME
     sam-scannerd - SAM-QFS daemon for manually-mounted devices

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-scannerd mshmid pshmid

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     sam-scannerd monitors the manually-mounted devices.  It will
     periodically check each device for newly inserted media.  If
     sam-scannerd finds media in the device, it will scan it  for
     a  label.   If  a  label is found, it will check the preview
     table to see if there are any requests for this  media.   If
     requests  are found, the SAM-QFS file system is notified and
     the device is assigned to the request.

     sam-scannerd is started automatically by sam-amld  if  there
     are  any  manually-mounted devices defined in the configura-
     tion file.  See mcf(4).
     mshmid is the id of the master shared memory segment created
     by  sam-amld.  pshmid is the id of the preview shared memory
     segment created by sam-amld.

SEE ALSO
     sam-amld(1M), mcf(4)
```

sam-serverd(1M)

```
NAME
     sam-sharefsd - Invokes the Sun QFS or SAM-QFS shared file
     system daemon

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-sharefsd

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-sharefsd process establishes connection to the
     current metadata server in a Sun QFS or SAM-QFS shared file
     system. The sam-sharefsd process on the metadata server
     opens a listener socket on the port associated with this
     file system. The shared file system port is defined in
     /etc/services as samsock.fs_name.

     The Sun QFS and SAM-QFS shared file system is a distributed
     file system that can be mounted on Solaris host systems.

     The sam-sharefsd process is initiated by the sam-fsd daemon.
     The sam-fsd daemon starts a shared file system daemon for
     each configured shared file system.

FILES
     Detailed trace information is written to the sam-sharefsd
     trace file.

SEE ALSO
     sam-fsd(1M).

     sammkfs(1M).

     samsharefs(1M).
```

sam-sharefsd(1M)

```
NAME
     sam-sharefsd - Invokes the Sun QFS or SAM-QFS shared file
     system daemon

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-sharefsd

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-sharefsd process establishes connection to the
     current metadata server in a Sun QFS or SAM-QFS shared file
     system. The sam-sharefsd process on the metadata server
     opens a listener socket on the port associated with this
     file system. The shared file system port is defined in
     /etc/services as samsock.fs_name.

     The Sun QFS and SAM-QFS shared file system is a distributed
     file system that can be mounted on Solaris host systems.

     The sam-sharefsd process is initiated by the sam-fsd daemon.
     The sam-fsd daemon starts a shared file system daemon for
     each configured shared file system.

FILES
     Detailed trace information is written to the sam-sharefsd
     trace file.

SEE ALSO
     sam-fsd(1M).

     sammkfs(1M).

     samsharefs(1M).
```

sam-shrink(1M)

```
NAME
     sam-shrink - Sun QFS and SAM-QFS disk space shrink process

SYNOPSIS
     /opt/SUNWsamfs/util/sam-shrink  file_system   |   family_set
     -remove | -release eq

AVAILABILITY
     SUNWqfs

     SUNWsamfs

DESCRIPTION
     This sam-shrink process is executed when a shrink of  a  Sun
     QFS  or SAM-QFS file system is required.  The samadm command
```

eq-remove or eq-release or the samu command remove or release cause the state of the specified device to be set to noalloc. Then, the file system requests the master daemon sam-fsd to start the process sam-shrink. The specified device (eq) must be a data device that resides within a ma file system. If the data device is a stripe group, the first eq of the stripe group must be specified. The eq state is changed to noalloc before sam-shrink is started. This means there will be no more allocation on this device.

The sam-shrink command should not be executed by the administrator separately. Pre and post processing that the SAM-QFS file system does automatically is necessary for a successful shrink.

The remove option copies all data that reside on eq to the other available data devices according to the mount parameters. Note, if eq is a stripe group, another stripe group must be available with the same number of devices.

The release option is only available on a SAM-QFS archiving file system. The release option marks all files that reside on eq offline. If any files have partial on-line, that data will be released, too. The release will fail if there are any files that have not been archived. The release will also fail if there are any files that are staging or being archived. If the release command fails, you may execute the release command again to release any newly achived files. If there are files that cannot be archived, then you may execute the remove command to move the data that resides on eq to the other available data devices according to the mount parameters.

After successful completion of the remove or release operation, the eq state will be off. It may take a long time to complete the release operation and an even longer time to complete the remove operation. You can monitor the logfile

and/or the /var/opt/SUNWsamfs/trace/sam-shrink file to check the status.

If the remove or release operation was not able to release or move all the files on the eq, the state will remain noalloc. The remove or release operation can be executed again on this eq. The shrink.log should be examined for reasons why the eq state could not be changed to off.

LOG

Within the shrink.cmd file, you can specify a log file for each Sun QFS or SAM-QFS file system. If no logfile=filename directive exists in the file, no logging occurs. For more information on the logfile=filename directive, see the shrink.cmd(4) man page.

The sam-shrink process creates the log file if it does not exist. The following example shows the log file entries for a release command followed by a remove command.

```
     Tue Sep 29 15:31:15 2008 Shrink process started: samfs5 release 15
```

```
          RE 6412.5  P S0 /sam1/250m
          RE 5131.5  P S0 /sam1/filecq
          NA 5095.4 -- S0 /sam1/filecu
          ER 5039.5 16 S0 /sam1/filedi
          NA 5039.2 -- S0 /sam1/lsc/filexx
          Tue Sep 29 15:31:55 2008 shrink process unsuccessful for samfs5 eq 15:
          busy files=1, unarchived files=2, total_errors=1
          Tue Sep 29 15:32:15 2008 Shrink process started: samfs5 remove 15
          MV 5095.4 -- S0 /sam1/filecu
          MV 5039.5 -- S0 /sam1/filedi
          MV 5039.2 -- S0 /sam1/lsc/filexx
          Tue Sep 29 15:33:21 2008 shrink process successful for samfs5 eq 15
```

The first line shows the arguments with which the shrink was invoked: file_system command equipment.

The next block of lines has one line per file processed: The fields are as follows:

Field Number	Content
1	This field contains the tag: RE for released, MV for removed, NA for not archived, or ER for error releasing or removing file. If the directive do_not_execute is set in the shrink.cmd file, this field contains the tag: NO.
2	This field contains the inode and generation number of the file.
3	This field contains the stage or errno tag. For the release command, tag is either S for file staged back on-line, P for partial staged back on-line, or -- for no stage action on this file. For a field with ER in the first field, this tag is the error number.
4	This field contains an S followed by the segment number. This is the number of the segment that was released.
5	This field contains the full path name of the released or moved file.

SEE ALSO

mcf(4). shrink.cmd(4).

sam-sony_helper(1M)

```
NAME
     sony - Attaches a Sony network-attached tape library through
     the DZC-8000S interface

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The SAM-QFS software package contains the Sun QFS and SAM-
     QFS interface to a Sony network-attached library.  This
     interface uses the DZC-8000S 3.01 interface supplied by
     Sony.  For more information on DZC-8000S, see the Sony
     PetaSite Application Interface DZC-8000S manual.  This
     manual is supplied by Sony.

CONFIGURATION
     It is assumed that the site has the PetaSite Controller
     (PSC) configured and operating with the Sony library.  In
     the Execute Mode of the PSC configuration, the following
     must be set to on:

     o  Thread With Load

     o  Unthread with Fast Unload

     o  Unthread with Eject

     o  Wait for Drive Use

     The Equipment Identifier field in the Sun QFS or SAM-QFS mcf
     file must be the full path name to a Sony parameters file.
     For more information on specifying a parameters file, see
     the mcf(4) man page.

     The parameters file consists of a list of keyword = value
     pairs.  All keyword and value specifications are
     case-sensitive and must be entered as shown on this man
     page.  The keyword and value specifications are as follows:

     userid = userid
               Identifies the user during initialization of the
               Sony library functions.  The userid values can be
               specified in hexadecimal or decimal.  The valid
               range is from 0 to PSCUSERIDMAX(0xfff), which is 0
               <= userid <= 65535 (decimal) or 0 <= userid <=
               0xffff (hexadecimal).  This is a required
               parameter.

     server = serverid
               Specifies the host name of the server running the
               PSC server code.  This is a required parameter.

     sonydrive binnum = path [ shared ]
               Specifies characteristics of the tape drive.
               There must be one sonydrive line for every drive
               assigned to Sun QFS or SAM-QFS in the mcf file.
```

```
                    This name must match the Equipment Identifier of
                    an entry in the mcf file.

                    The following arguments follow the sonydrive
                    keyword:

                    binnum    Specifies the bin number assigned to the
                              drive in the PSC configuration. The bin
                              number can be identified using the PSC
                              Monitoring and Maintenance terminal.
                              This is a required argument.

                    path      Specifies the Solaris /dev/rmt/ path
                              name to the device.  The path must match
                              the Equipment Identifier of an entry in
                              the mcf file.  This is a required
                              argument.

                    shared    Specifies that this drive is shared with
                              other processes.  For example, this
                              drive can be shared between multiple Sun
                              QFS or SAM-QFS servers.  This is an
                              optional argument.
```

EXAMPLE

The following example shows the configuration files for a network-attached Sony library with Sony DTF tapes.

Here are the sample entries in the mcf file. The catalog file is placed in the default directory, which is /var/opt/SUNWsamfs/catalog.

The mcf file is as follows:

```
#
# This is the file: /etc/opt/SUNWsamfs/mcf
# This file shows sample mcf entries for a Sony network-attached
# robot with Sony DTF tapes.
#
/etc/opt/SUNWsamfs/sonyfile 50 pe sony50 on /var/opt/SUNWsamfs/sony50cat
/dev/rmt/0cbn              51 so sony50 on
/dev/rmt/1cbn              52 so sony50 on
```

The parameters file for a Sony library supporting Sony DTF tapes is as follows:

```
#
# This is file: /etc/opt/SUNWsamfs/sonyfile

#
# The userid identifies the user during initialization of
# the PetaSite library functions. Valid IDs are 0 to
# PSCUSERIDMAX(0xfff).
#
userid = 65533
#
# The server identifies the hostname for the server running
# the DZC-8000S server code.
#
```

```
server = europa
#
# The sonydrive bin number 1001 is from the PSC configuration file
#
sonydrive 1001 = /dev/rmt/0cbn shared  # a comment
#
# The sonydrive bin number 1002 is from the PSC configuration file
#
sonydrive 1002 = /dev/rmt/1cbn         # a comment
```

IMPORT/EXPORT

The physical adding and removing of cartridges in a Sony network-attached library is accomplished using the PSC utilities. The import(1M) and export(1M) commands affect only the library catalog. Therefore, importing and exporting cartridges with the Sony network-attached library proceeds according to the following two-step process:

1. Physically import or export the cartridge using the PSC software.

2. Virtually update the library catalog using the Sun QFS or SAM-QFS import/export utilities.

The import(1M) command has an optional -v option that allows you to specify the VSN to be added. The samsony package verifies that PSC knows about the VSN before updating the catalog with the new entry. The export(1M) command removes the entry from the catalog.

CATALOG

There are several methods for building a catalog for a Sony network-attached library. You should use the method that best suits your system configuration, typically depending on the size of the catalog that is needed.

Method 1: Create a catalog with existing VSN entries. You can build a catalog that contains entries for many tapes by using the build_cat(1M) command. As input to the build_cat(1M) command, you need to create a file that contains the slot number, VSN, bar code label, and media type. For example, the file input_vsns follows:

```
0  "SEG001"  "SEG001"  so
1  "SEG002"  "SEG002"  so
2  TEST1     TEST1     so
3  TEST2     TEST2     so
```

The input_vsns file can be used as input to the build_cat(1M) command as follows:

```
build_cat input_vsns /var/opt/SUNWsamfs/sony50cat
```

Method 2: Create a null catalog and import VSN entries. You can create an empty catalog and populate it. To create a catalog that will accommodate 1000 slots, use the build_cat(1M) command as follows:

```
build_cat -s 1000 /dev/null /var/opt/SUNWsamfs/catalog/sony50cat
```

```
     Use the import(1M) command to add VSNs to this catalog, as
     follows:

     import -v "SEG005" 50

     Method 3: Use the default catalog and import VSN entries.
     If a catalog path name is not specified in the mcf file, a
     default catalog is created in
     /var/opt/SUNWsamfs/catalog/family_set_name when Sun QFS or
     SAM-QFS is initialized.  Following initialization, you must
     import VSN entries to this catalog by using the import
     command as follows:

     import -v "SEG005" 50

     In the previous import(1M) command, 50 is the Equipment
     number of the library as specified in the mcf file.

FILES
     mcf                      The configuration file for the Sun
                              QFS and SAM-QFS software.

     /opt/SUNWsamfs/lib/libpsc.so
                              The PSC library supplied by Sony.

     /opt/SUNWsamfs/sbin/sony_helper
                              A program to issue commands to the
                              Sony PSC.

SEE ALSO
     build_cat(1M), dump_cat(1M), export(1M), import(1M), sam-
     robotsd(1M).

     mcf(4).
```

sam-sonyd(1M)

```
NAME
     sam-robotsd, sam-genericd, sam-stkd, sam-ibm3494d, sam-sonyd
     - SAM-QFS media changer daemons

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-robotsd mshmid pshmid

     /opt/SUNWsamfs/sbin/sam-genericd mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-stkd mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-ibm3494d mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-sonyd mshmid pshmid equip

AVAILABILITY
     SUNWsamfs
```

DESCRIPTION

The sam-robotsd daemon starts and monitors the execution of the media changer library control daemons for SAM-QFS. The sam-robotsd daemon is started automatically by the sam-amld daemon if there are any libraries defined in the mcf file. The sam-robotsd daemon starts and monitors the correct daemon for all defined libraries. For more information on the mcf file, see the mcf(4) man page.

Each library daemon is responsible for monitoring the preview table for the VSNs that are controlled by that daemon. If a request is found for one of its VSNs, the daemon finds an available drive under its control and moves the cartridge into that drive. When the device is ready, the daemon notifies the SAM-QFS library daemon, and the device is assigned to the waiting process.

The identifiers are as follows:

mshmid	The identifier of the master shared memory segment created by the sam-amld daemon.
pshmid	The identifier of the preview shared memory segment created by the sam-amld daemon.
equip	The equipment number of the device.

The sam-genericd daemon controls libraries that conform to the SCSI II standard for media changers, and it is the daemon that controls the ADIC/Grau ABBA library through the grauaci interface. For more information on this interface, see the grauaci(7) man page.

The sam-stkd daemon controls StorageTek libraries through the ACSAPI interface and is included in the SAM-QFS software package. For more information on this interface, see the stk(7) man page.

The sam-ibm3494d daemon controls IBM 3494 tape libraries through the lmcpd interface and is included in the SAM-QFS software package. For more information on this interface, see the ibm3494(7) man page.

The sam-sonyd daemon controls Sony libraries through the Sony DZC-800S PetaSite Application Interface Library and is included in the SAM-QFS software package. For more information on this interface, see the sony(7) man page.

FILES

mcf	The master configuration file for SAM-QFS environments.

SEE ALSO

sam-amld(1M).

mcf(4).

acl2640(7), acl452(7), grauaci(7), ibm3494(7), ibm3584(7), sam-remote(7), sony(7), stk(7).

sam-stagealld(1M)

```
NAME
     sam-stagealld - SAM-QFS associative staging daemon

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-stagealld

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     sam-stagealld is responsible  for  the  associative  staging
     feature.   It  is initiated by sam-fsd.  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-fsd(1M)
```

sam-stagerd(1M)

```
NAME
     sam-stagerd - Invokes the SAM-QFS stage daemon

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-stagerd

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-stagerd process stages files in a SAM-QFS file
     system.  Staging is the process of copying a nearline or
     offline file from its archive storage back to online
     storage.

     The SAM-QFS file system staging capability allows you to
     stage files immediately, to never stage files, and specify
     other staging actions.  The sam-stagerd process is initiated
     by the sam-fsd daemon.

     By default, the stager uses the default behaviors described
     on the stager.cmd(4) man page.

OUTPUT FORMAT
     The stager can produce a log file containing information
     about files staged.
      Here is an example:

     E 2004/02/02 15:23:43 lt ST0004 d2.1 11228.7 10485760 /sam9/testa 1 124 sam root 0
```

```
F 2004/02/03 14:37:41 lt CFX598 5410.23 15339.5 15703 /sam9/rdump 1 hm129959 other root 42
```

Field	Description
1	C for stage cancel. E for error. F for stage finish. S for stage start.
2	Date of stage action.
3	Time of stage action.
4	Stage media.
5	VSN. For removable media cartridges, this is the volume serial name. For disk archives, this is the disk volume name and tar file path.
6	Physical position of start of archive file on media and file offset on the archive file / 512.
7	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.
8	Length of file if written on only 1 volume. Length of section if file is written on multiple volumes.
9	Name of file.
10	Copy number being staged.
11	User name of the file’s owner.
12	Group of the file’s owner.
13	User name of the requestor of the stage.
14	Equipment number from the mcf of the device on which the stage occurred.

FILES

If the stager command file is present in /etc/opt/SUNWsamfs/stager.cmd, the sam-stagerd process reads that file.

SEE ALSO

stage(1).

sam-fsd(1M).

stager.cmd(4).

sam-stagerd_copy(1M)

```
NAME
     sam-stagerd_copy - Invokes the SAM-QFS stage copy daemon

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-stagerd_copy

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-stagerd_copy process copies SAM-QFS files from
     removable media cartridges.  It is executed by the sam-
     stagerd(1M) process.

SEE ALSO
     sam-stagerd(1M)
```

sam-stk_helper(1M)

```
NAME
     stk - The StorageTek interface through ACSAPI

AVAILABILITY
     SUNWsamfs

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

CONFIGURATION
     It is assumed that the site has the server daemons (CSI  and
     ACSLM) configured and operating with the StorageTek library.

     The  Equipment  Identifier  field  in  the  mcf  file,  (see
     mcf(4)),  is the full path name to a parameters file used by
     stk.  This file consists of keyword = value and path_name  =
     value  pairs.  All keyword, path_name, and value entries are
     case-sensitive.

     The keywords are:

     access  This is the user_id used by this client  for  access
             control.   If  this  parameter  is not supplied, the
             access control string will  be  a  null  string  (no
             user_id).

     hostname
             This is the hostname for the server that is  running
```

```
        ACSLS.  If the hostname is not supplied, the default
        will be localhost.  All sites should set this value.

ssihost This is the name used for the SAM-QFS server when  a
        multihomed  SAM-QFS  server  is  used.   The ssihost
        would be the name of the SAM-QFS server on  the  lan
        connecting  to  the  ACSLS host.  Only sites where a
        multihomed SAM-QFS server is used need to supply  an
        ssihost value. The default will be localhost.

portnum This is the portnum for SSI services on  the  server
        that  is  running  ACSLS.  If the port number is not
        supplied, the default is 50004.  Please note that if
        you  are  running co-hosted ACSLS 5.3 or higher, the
        default value does  not  work  (try  a  higher  port
        number,  like  50014).   If you are running multiple
        connections to ACSLS servers, then the  port  number
        for  each  stk configuration file needs to be unique
        (for example, 50014  in  one,  50015  in  the  next,
        etc.).

ssi_inet_port
        This is the fixed port number for incoming responses
        and specifies the port the SSI will use for incoming
        ACSLS responses in a  firewall  environment.   Valid
        values  are  1024  -  65535,  and  0.  Setting  this
        environmental variable to a non-zero value makes the
        SSI  use  this  port  for  incoming ACSLS responses.
        This means that the firewall needs to allow incoming
        requests  on  that  port  in  order  for  the  ACSLS
        responses to be received by the  SSI.  Setting  this
        value to zero or leaving it unset indicates that the
        previous behavior of allowing the port to be dynami-
        cally allocated will remain in effect.

csi_hostport
        This firewall environmental variable  specifies  the
        port  to  which the SSI will send its ACSLS requests
        on the ACSLS server. Setting this variable eliminate
        queries  to  the  portmapper on the ACSLS server and
        instead, sends requests to this port  on  the  ACSLS
        server.  Valid  values are 1024 - 65535, and 0. Set-
        ting this variable to zero or leaving it unset indi-
        cates  that  the  previous  behavior of querying the
        portmapper on the ACSLS server will continue  to  be
        used.

capid   This specifies the CAP (Cartridge Access Port) to be
        used  for exporting of volumes when the -f option is
        used with export command.  Following  the  capid  is
        the   description  of  this  CAP  in  terms  of  the
        StorageTek library.  This description starts with an
        open parenthesis followed by 3 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 CAP  as  configured  in
          the StorageTek library.
     lsm  is the LSM number for this CAP  as  configured  in
          the StorageTek library.

     cap  is the CAP number for this CAP  as  configured  in
          the StorageTek library.

capacity
     This is used to set the capacity of the media supported
     by  the  StorageTek.   The  parameter  to capacity is a
     comma separated list of index = value pairs enclosed in
     parentheses.   index  is  the index into the media_type
     file (supplied by StorageTek and  located  on  the  ACS
     system) and value is the capacity of that media type in
     units of 1024 bytes.  You should only  need  to  supply
     this  entry  if  the  ACS  is not returning the correct
     media type or new media types have been added.  Sun QFS
     and  SAM-QFS  have  defaults for index values that were
     current at the  time  of  release.   Generally,  it  is
     necessary  to  supply  an  index only for new cartridge
     types.  For the capacity of each  cartridge  type,  see
     the SAM-QFS Storage and Archive Management Guide.

device_path_name
     There is one device_path_name  entry  for  every  drive
     attached  to  this client.  The device_path_name is the
     path to the device on the client. This name must  match
     the  Equipment  Identifier of an entry in the mcf file.
     Following the device_path_name is  the  description  of
     this  drive  in  terms of the StorageTek library.  This
     description starts with an open parenthesis followed by
     4   keyword   =   value   pairs  followed  by  a  close
     parenthesis.  The keyword =  value  pairs  between  the
     parentheses  may  be  separated by a comma (,), a colon
     (:) or by white space. Following the close  parenthesis
     is  an  optional  keyword  used  by Sun QFS and SAM-QFS
     software to designate when a drive is shared with other
     Sun  QFS  and SAM-QFS servers.  The keyword identifiers
     and their meanings are as follows:

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

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

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

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

     shared
          The shared keyword follows the close  parenthesis.
          This  keyword  is optional and is used to indicate
          the drive is shared with other Sun QFS and SAM-QFS
          servers.
```

EXAMPLE

Here is a sample parameters file and mcf entries for a StorageTek library:

```
#
# This is file: /etc/opt/SUNWsamfs/stk50
#
hostname = acsls_server_name
portnum = 50004
ssi_inet_port = 0
csi_hostport = 0
access = some_user  # No white space allowed in the user_id field
capid = (acs=0, lsm=1, cap=0)
/dev/rmt/0cbn = (acs=0, lsm=1, panel=0, drive=1)         #a comment
/dev/rmt/1cbn = (acs=0, lsm=1, panel=0, drive=2) shared  #a comment
capacity = (0=215040, 1=819200, 5=10485760)
```

The mcf file entries that reference this configuration file are:

```
#
# Sample mcf file entries for a StorageTek library
#
/etc/opt/SUNWsamfs/stk50      50  sk  sk50   - /var/opt/SUNWsamfs/catalog/sk50
/dev/rmt/0cbn                 51  st  sk50   -
/dev/rmt/1cbn                 52  st  sk50   -
```

IMPORT/EXPORT

Since the physical adding and removing of cartridges in the StorageTek library is done with ACSLM utilities, the import/export commands and GUI buttons will only affect the library catalog. The import command has optional parameters for supplying a single volume to be added or to add a number of volumes from a pool (see import(1M)). export (see export(1M)) will remove an entry from the catalog.

CATALOG

The Sun QFS and SAM-QFS systems automatically build a library catalog for a StorageTek automated library. However, you must populate the library catalog. For information on populating the library catalog, see the SAM-QFS Storage and Archive Management Guide.

FILES

mcf	The configuration file for the Sun QFS and SAM-QFS software.
/etc/opt/SUNWsamfs/scripts/ssi.sh	A shell script used to start ssi_so.
/opt/SUNWsamfs/sbin/ssi_so	A shared object version of the SSI daemon supplied by StorageTek.
/opt/SUNWsamfs/lib/stk/*	The libraries needed by the API interface supplied by

```
                                   StorageTek.

     /opt/SUNWsamfs/sbin/stk_helper
                                   A program  to  issue  commands
                                   for the StorageTek ACSAPI

SEE ALSO
     build_cat(1M), dump_cat(1M),  export(1M),  import(1M),  sam-
     robotsd(1M).

     mcf(4).

     ssi_so(7).

     SAM-QFS Configuration and Administration Guide.
```

sam-stkd(1M)

```
NAME
     sam-robotsd, sam-genericd, sam-stkd, sam-ibm3494d, sam-sonyd
     - SAM-QFS media changer daemons

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-robotsd mshmid pshmid

     /opt/SUNWsamfs/sbin/sam-genericd mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-stkd mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-ibm3494d mshmid pshmid equip

     /opt/SUNWsamfs/sbin/sam-sonyd mshmid pshmid equip

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sam-robotsd daemon starts and monitors the execution of
     the media changer library control daemons for SAM-QFS.  The
     sam-robotsd daemon is started automatically by the sam-amld
     daemon if there are any libraries defined in the mcf file.
     The sam-robotsd daemon starts and monitors the correct
     daemon for all defined libraries.  For more information on
     the mcf file, see the mcf(4) man page.

     Each library daemon is responsible for monitoring the
     preview table for the VSNs that are controlled by that
     daemon.  If a request is found for one of its VSNs, the
     daemon finds an available drive under its control and moves
     the cartridge into that drive.  When the device is ready,
     the daemon notifies the SAM-QFS library daemon, and the
     device is assigned to the waiting process.

     The identifiers are as follows:

     mshmid    The identifier of the master shared memory segment
```

```
                    created by the sam-amld daemon.

          pshmid    The identifier of the preview shared memory
                    segment created by the sam-amld daemon.

          equip     The equipment number of the device.

          The sam-genericd daemon controls libraries that conform to
          the SCSI II standard for media changers, and it is the
          daemon that controls the ADIC/Grau ABBA library through the
          grauaci interface.  For more information on this interface,
          see the grauaci(7) man page.

          The sam-stkd daemon controls StorageTek libraries through
          the ACSAPI interface and is included in the SAM-QFS software
          package.  For more information on this interface, see the
          stk(7) man page.

          The sam-ibm3494d daemon controls IBM 3494 tape libraries
          through the lmcpd interface and is included in the SAM-QFS
          software package.  For more information on this interface,
          see the ibm3494(7) man page.

          The sam-sonyd daemon controls Sony libraries through the
          Sony DZC-800S PetaSite Application Interface Library and is
          included in the SAM-QFS software package.  For more
          information on this interface, see the sony(7) man page.

     FILES
          mcf       The master configuration file for SAM-QFS
                    environments.

     SEE ALSO
          sam-amld(1M).

          mcf(4).

          acl2640(7), acl452(7), grauaci(7), ibm3494(7), ibm3584(7),
          sam-remote(7), sony(7), stk(7).
```

samadm(1M)

```
     NAME
          samadm - Sun QFS and SAM-QFS main administrative command

     SYNOPSIS
          samadm servicetag add|delete

          samadm eq-add eq_number#

          samadm eq-release eq_number#

          samadm eq-remove eq_number#

          samadm eq-alloc eq_number#
```

samadm eq-noalloc eq_number#

samadm -?|--help

DESCRIPTION

The samadm command is a single command line interface to many Sun QFS and SAM-QFS commands. Initially, it contains commands that are new to Sun QFS and SAM-QFS 5.0, but will in the future contain subcommands for most QFS administrative functions.

The detailed description of each subcommand follows.

SUBCOMMANDS

servicetag

The add operand to the servicetag subcommand adds service tags to the service tag repository depending on whether the SUNWqfs or SUNWsamfs package is installed. If the SUNWqfs package is installed, a Sun QFS service tag is added. If the SUNWsamfs package is installed, both a Sun QFS and a SAM-QFS service tag are added. See stclient(1M) for more information on service tags.

The delete operand to the servicetag subcommand deletes both Sun QFS and SAM-QFS service tags from the service tag repository.

The servicetag subcommand is not intended to be needed to be invoked by the administrator. The Sun QFS software automatically adds service tags when first configured or mounted, and deletes service tags when the Sun QFS or SAM-QFS package is removed.

eq-add

This subcommand adds an equipment to an existing mounted Sun QFS file system. The equipment must be added to the mcf(4) file, and samd config must be run prior to adding the equipment number to a file system. The equipment will be placed into the off state.

In a non-shared file system, after the equipment number is added, it is placed into the on state.

In a shared file system, after the equipment number is added, it is initially placed into the unavail state. All the mcf files on the clients must be updated to include the new/changed equipment number, and samd config run. After this has been done, To place this equipment number into full read/write status, change the state to alloc via samadm eq-alloc eq_number#.

eq-release

This subcommand releases the disk space associated with files that have valid archive copies from SAM-QFS. It can be used prior to an eq-remove command to quickly release space on a device that is to be removed for hardware failure or other reasons.

The eq-release subcommand starts a background process called sam-shrink which releases space on the equipment number The releasing process may take some time to complete. Progress can be monitored via samu(1M) (m display). When the release is started, the equipment is first put into noalloc state to prevent further data allocation on that equipment. When the releasing process is complete, the equipment is placed into off state if all space was successfully released.

eq-remove

This subcommand removes an equipment number from an existing mounted Sun QFS file system. It is intended to be used to remove an equipment for reuse, or to remove an equipment that needs replacement because of hardware failure.

The eq-remove subcommand starts a background process called sam-shrink which copies the data on the equipment number to be removed to other equipments in the file system. The removal process may take some time to complete. Progress can be monitored via samu(1M) (m

display). When the remove is started, the equipment to be removed is first put into noalloc state to prevent further data allocation on that equipment. When the removal process is complete, the equipment is placed into off state if all space was successfully moved to other equipments.

eq-alloc

This subcommand changes the state of an equipment number to alloc which allows new data allocations to be placed on it. This subcommand is only legal for equipments which are currently in the noalloc or unavail state.

eq-noalloc

This subcommand changes the state of an equipment number to noalloc which prevents new data allocations to be placed on it. This subcommand is only legal for equipments which are currently in the on state.

-?|--help

Displays a command syntax summary.

SEE ALSO

samservicetag(1M) samu(1M) samd(1M) mcf(4)

sambcheck(1M)

```
NAME
     sambcheck - Lists block use for a Sun QFS or SAM-QFS file
     system

SYNOPSIS
     sambcheck fs_name block_num[.ord] [block_num[.ord]] ...

AVAILABILITY
     SUNWqfs

     SUNWsamfs

DESCRIPTION
     The sambcheck command determines the current usage of each
     requested block_num in a Sun QFS or SAM-QFS file system.
     This command must be run as root.  For accurate results, the
     file system should be unmounted.

     This command accepts the following arguments:

     fsname    The family set name, as specified in the mcf file,
               for the file system for which the usage list is
               desired.

     block_num A number that identifies the blocks for which
               statistics should be obtained.  Blocks are in
               1024-byte (1 kilobyte) units.  Use one of the
               following formats:

               o Decimal.  Default.

               o Octal.  The block_num must be preceded by 0.

               o Hexadecimal.  The block_num must be preceded by
                 0x or 0X.

     ord       The partition number (ordinal) upon which the
               block use is to be found.  If no .ord is
               specified, all partitions are examined.  All ord
               specifications are assumed to be in decimal.

OUTPUT
     The output from this command is one line per requested block
     number for each explicit or implicit ordinal.  The block
     number is displayed as entered, followed by its decimal form
     in parentheses, followed by text indicating the usage
     determined for the block_num[.ord].

EXAMPLES
     bilbo# sambcheck samfs1 0x40 0x42.0 0x42.2 0x7a150 0x89cd0.01 512
     block 0x40 (64.0) is a data block for .inodes containing 1 - 32
     block 0x40 (64.1) is a data block for directory inode 26.1

     block 0x40 (64.2) is a data block for inode 934767.1
     block 0x40 (64.4) is a data block for inode 934766.1
     block 0x42.0 (66.0) is a data block for .inodes containing 1 - 32
     block 0x42.2 (66.2) is a free data block
     block 0x7a150 (500048.0) is a data block for .inodes containing 999969 - 1000000
     block 0x7a150 (500048.1) is a data block for directory inode 787628.1
     block 0x7a150 (500048.2) is a data block for inode 934767.1
     block 0x7a150 (500048.4) is a free data block
     block 0x89cd0.01 (564432.1) is an indirect block for inode 934767.1
     block 512 (512.0) is a data block for .inodes containing 897 - 928
     block 512 (512.1) is a data block for directory inode 65.1
     block 512 (512.2) is a data block for inode 934767.1
     block 512 (512.4) is a data block for inode 934766.1
```

samchaid(1M)

```
NAME
     samchaid - change file admin set ID attribute

SYNOPSIS
     samchaid [ -fhR ] aid filename...

AVAILABILITY
     SUNWsamfs

     SUNWqfs

DESCRIPTION
     samchaid sets the admin set ID attribute of files and direc-
     tories.

     If a directory’s admin set ID is set, files and  directories
     subsequently  created  in  that directory inherit that admin
     ID.  Only the superuser may set the admin ID.

OPTIONS
     -f   Force.  Do not report errors.

     -h   If the file is a symbolic link, change the admin set ID
          of the symbolic link.  Without this option, the group
          of the file referenced by the symbolic link is changed.

     -R   Recursive.  samchaid descends through any directories
          and subdirectories, setting the specified admin set ID
          as it proceeds.  When a symbolic link is encountered,
          the admin set ID of the target file is changed (unless
          the -h option is specified), but no recursion takes
          place.

SEE ALSO
     samquota(1), sls(1)
```

samcmd(1M)

```
NAME
     samcmd - Executes Sun QFS and SAM-QFS operator utility  com-
     mands

SYNOPSIS
     samcmd command

AVAILABILITY
     SUNWqfs
     SUNWsamfs

DESCRIPTION
     samcmd executes a single Sun QFS or SAM-QFS operator utility
     command.   Its  purpose is to provide shell script access to
     the commands and displays available in samu(1M).

     samcmd uses the  first  argument  as  the  samu  command  or
     display  name.   Succeeding  arguments are the arguments for
     that samu command.

COMMANDS
     The syntax for the commands is identical to  that  shown  in
     the  COMMANDS  section of samu(1M).  Note that the colon (:)
     hot key is not required for samcmd to  distinguish  commands
     from displays.

DISPLAYS
     samcmd can produce displays on standard  output  similar  to
     those  displayed by samu.  While for samu the information is
     paged to display a screen at a time if there  is  more  than
     one  screen  of  information  available, samcmd produces the
     entire amount of information for  a  given  display.   Hence
     there  is no need for equivalents of the control-f, control-
     b, control-d, and control-u hotkeys.  Note that the  format-
     ting  of  the  information  may be slightly different on the
     samcmd output file than on the samu display.  Since the for-
     mat  of  the display control (single letter) commands can be
     modified by other hotkeys under samu, some  equivalents  are
     provided for samcmd as follows:

      Display   Arguments

     a          filesystem
     n          mediatype
     p          mediatype
     r          mediatype
     u          mediatype [path]
     v          eq [sort] [I | I I]
     w          mediatype [path]

     The sort selections for the v display are:  1 slot, 2 count,
     3  usage,  4  VSN,  5  access time, 6 barcode, 7 label time.
     Specifying a single I for the v  display  shows  a  two-line
     display  with  the  barcode,  blocksize,  etc. in the second
     line.  Specifying two I's for the v display shows a two-line
     display  with the archiver volume reservation information in
     the second line.

EXAMPLES
     The following example loads  a  cartridge  from  slot  2  in
     automated library 30:

           samcmd load 30:2

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

           samcmd a samfs3

     The following example produces a display, on  standard  out-
     put,  of  the  staging queue restricted to stages from media
     type "lt", showing the full paths of the files to be staged.

           samcmd u lt path

     The  following  example  produces  a  display  of  automated
     library  50's  catalog, with the archiver volume reservation
     information, on standard output:

           samcmd v 50 I I

SEE ALSO
     samu(1M)
```

samcrondump(1M)

```
NAME
     samcrondump - Used to create a Sun Storage  Archive  Manager
     dump file.

SYNOPSIS
     /opt/SUNWsamfs/sbin/samcrondump

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     samcrondump is a program used to create a  dumpfile.  It  is
     invoked by cron and takes an id comprised of the file system
     name and an optional directory at which to start.

SEE ALSO
     samcronfix(1M) fsmadm(1M) fsmgmtd(1M)
```

samcronfix(1M)

```
NAME
     samcronfix - Program  to  update  crontab  for  Sun  Storage
     Archive Manager

SYNOPSIS
     /opt/SUNWsamfs/sbin/samcronfix

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     samcronfix is a program to update crontab.

SEE ALSO
     samcrondump(1M) fsmadm(1M) fsmgmtd(1M)
```

samd(1M)

```
NAME
     samd - SAM-QFS daemon management and configuration command

SYNOPSIS
     /opt/SUNWsamfs/sbin/samd buildmcf

     /opt/SUNWsamfs/sbin/samd config

     /opt/SUNWsamfs/sbin/samd start

     /opt/SUNWsamfs/sbin/samd stop

     /opt/SUNWsamfs/sbin/samd hastop

AVAILABILITY
     SUNWqfs

     SUNWsamfs

DESCRIPTION
     The samd utility starts up or shuts down the sam-amld
     daemon, or shuts down the HSM daemons for HA-SAM failover.
     This utility can also be used to reinitialize the Sun QFS
     and SAM-QFS configuration files and allow changes to take
     effect.

OPTIONS
     This command accepts the following options:

     buildmcf  Creates a new /etc/opt/SUNWsamfs/mcf file if one
               does not exist.  This is useful when configuring
               shared clients for the first time, after the file
               systems have been created on the metadata server.
               Only disk devices will be discovered and entered
               into the mcf.
```

config	Causes the sam-fsd daemon to (re)configure based on changes to the /etc/opt/SUNWsamfs/mcf, /etc/opt/SUNWsamfs/archiver.cmd, and /etc/opt/SUNWsamfs/defaults.conf files.
start	Starts up the sam-amld daemon if the /etc/opt/SUNWsamfs/mcf file exists and the sam-amld daemon is not already running. Implemented only in SAM-QFS archiving environments. Not a valid argument in a Sun QFS file system-only environment.
stop	Kills the sam-amld daemon. Implemented only in SAM-QFS archiving environments. Not a valid argument in a Sun QFS file system-only environment.
hastop	Kills the sam-archiverd, sam-stagealld, sam-stagerd and sam-amld daemon for HA-SAM failover. Daemons killed by ’hastop’ will not be restarted by the sam-fsd. Implemented only in SAM-QFS archiving environments. Not a valid argument in a Sun QFS file system-only environment.

SEE ALSO

sam-fsd(1M), sam-amld(1M).

defaults.conf(4), mcf(4).

samdb(1M)

NAME

samdb - SAM-QFS sideband database commands

SYNOPSIS

```
samdb check family_set [-f] [-s] [-q]

samdb create family_set [-s schema_file]

samdb dump family_set [-a] [-s] [-f file name]

samdb drop family_set

samdb load family_set [-i] [-f file name]

samdb query family_set [-t type] [-c] [-s] [-i inum]
    [-f file] [-v vsn]
```

DESCRIPTION

The samdb commands are used to configure and query a SAM-QFS MySQL database. This database retains metadata information for each file in the file system. Use of the SAM-QFS MySQL database implies the SAM-QFS server has access to a MySQL server. The SAM-QFS server need not host the MySQL server if

```
     as  network access to the database host system is available.
     The samdb.conf(4) configuration  file  contains  the  access
     parameters for the database.

     Use of the SAM-QFS MySQL database is optional and is  speci-
     fied  by the mount option sam_db. If set, the fsalogd daemon
     is started at mount time.  The file system sends  events  to
     the  fsalogd  who  writes the events to a log file. A second
     daemon, sam-dbupd, reads the events  from  the  fsalogd  log
     files and updates the SAM-QFS database.

     The database is initially populated by the samdb  load  com-
     mand.  Input  to  the  samdb  load  command is the load file
     created by samfsdump.

     Example 1: Generate load file from an existing dumpfile.

     # samfsrestore -S -Z /tmp/samfs1/dbload -f /path/to/dump/samfs1.dump

     Example 2: Generate load file while  performing  a  samfsre-
     store.

     # samfsrestore -Z /tmp/samfs1dbload -f /path/to/dump/samfs1.dump

     Example 3: Pipelining samfsdump to load database.

     # samfsdump -S -Z - /samfs1 | samdb load samfs1

     Once the SAM-QFS MySQL database is  populated,  the  perfor-
     mance of samfsdump(1M) can be improved by using the database
     for path name creation.  This is either  done  with  a  file
     created by samdb dump or pipelined together.

     Example: Pipelining samdb dump to samfsdump

     # samdb dump samfs1 | samfsdump -Y -f /path/samfs1.dump -

SAMDB COMMANDS
     A series of commands are provided to configure and query the
     SAM-QFS  MySQL database. The specific options to the indivi-
     dual commands are listed below.

     family set

         Specifies the family set name of the file  system.  This
         family  set  name  must  be configured in the samdb.conf
         file.

     help

         Displays a command syntax summary.

     Here is a list of the samdb commands and an  explanation  of
     the options.

     samdb check family_set [-f] [-s] [-q]

         Checks the database against specified  file  system  for
         consistency.   This  scans  the inodes of the filesystem
```

```
    making sure the entries in the database are correct.

-f

    Perform a fast consistency check.  This  skips  checking
    the  directory  namespace  in  the  database, using only
    information found in the inode.

-s

    Perform a scan without repairing database errors.

-q

    Quiet output, only display the number of problems found.

samdb create family_set [-s schema_file]

    Creates the database for the specified filesystem.

-s schema_file

    Specifies the schema file to use. The  default  file  is
    /opt/SUNWsamfs/etc/samdb.schema.  The  schema  file con-
    tains a series of CREATE TABLE commands.

samdb dump family_set [-a] [-s] [-f file name]

    Generates a list of files for samfsdump.

-a

    Use absolute pathnames in file list.   This  will  allow
    samfsdump  to be ran outside of the root of the filesys-
    tem.  The default is relative pathnames.

-s

    Sort dump file by parent  directory  ids.   This  option
    groups  files  together  in the dump file based on their
    parent directories.   Subsequent  samfsdump  performance
    will not be improved when using this option.

-f file name

    Specifies file to send output  to.  If  a  file  is  not

    supplied, output goes to standard out.

samdb drop family_set

    Drops the database for the  specified  file  system.   A
    confirmation prompt will appear.

samdb load family_set [-f file name]

    Loads a database from a samfsdump file.  After loading a
    database  a  samdb  check  should  be performed for that
    filesystem.
```

```
-i

    Use an inode scan instead of a load  file  to  load  the
    database.   This can be used if no recent samfsdump file
    is available to generate the load file with.

-f filename

    The filename of  the  load  file,  default  is  standard
    input.This    file    can    be    generated    by
    samfsdump(1M)/samfsrestore(1M) -Z option.

samdb query family_set [-t type] [-c] [-s] [-i inum]
    [-f file] [-v vsn]

    Queries a database for files or vsns based  on  provided
    file  or  vsn  information.   Multiple -ifv terms can be
    provided.  Like terms  are  OR'd  together,  and  unlike
    terms are AND'd.

-t {vsn,file}

    The query type to produce, either vsn or file.   If  vsn
    is chosen a list of vsns matching the -ifv terms will be
    output.  If file is chosen a list of files matching  the
    -ifv  terms  will  be  output.   If both query types are
    chosen, e.g. -t vsn -t file, then a list of files broken
    down by vsn will be output.  The default is file.

-c

    Produce  a  count  instead  outputing  a  result   list.
    Depending  on  the  query  type,  the output will be the
    number of either vsns  or  files  that  match  the  -ifv
    terms.

-s

    Sort the results.   Results  are  sorted  alphabetically
    ascending.

-i inum

    Match the provided inode number.  This  is  provided  to
    query  the  database  for  which  files or vsns have the
    given inode number.

-f filename

    Match the provided filename.  This queries the  database
    for files that match the given filename.  The % wildcard
    character can be used within a filename to match  multi-
    ple  files.  Paths must either being with a wildcard, or
    be absolute relative to the mount point.

    For example /dir1/file1 or %dir1/file1 are  valid.   The
    first would match the dir1 directory in the mount point.
    The second would match any directory ending in dir1.
```

```
     -v vsn

         Match the provided vsn. Depending on the query type this
         will  output a list of files on the vsn, or restrict the
         results to the provided vsn.

SEE ALSO
     samdb.conf(4) samfsdump(1M) samfsrestore(1M)
```

samexplorer(1M)

```
NAME
     samexplorer - Generates a Sun QFS or SAM-QFS diagnostic
     report

SYNOPSIS
     samexplorer [-u] [report_name] [num_lines]

AVAILABILITY
     SUNWqfs

     SUNWsamfs

DESCRIPTION
     The samexplorer command produces a diagnostic report of the
     Sun QFS or SAM-QFS server configuration and collects log
     information.

     The samexplorer command should be run as root.  The command
     generates a diagnostic report by default in file:
     /tmp/SAMreport.hostname.YYYYMMDD.HHMMZ.tar.gz

     The report should be sent to your Oracle Corporation
     authorized service provider or to Oracle Corporation technical
     support as specified in your maintenance contract.

OPTIONS
     This command accepts the following options:

     [-u]          Generate separate output files in an
                   unarchived/uncompressed format.

     report_name   The name of the diagnostic report file.  The
                   default is
                   /tmp/SAMreport.hostname.YYYYMMDD.HHMMZ.tar.gz

     num_lines     The number of lines to capture from each log
                   file.  The default is 1000.

EXAMPLE
     sunfire# samexplorer

     Report name:     /tmp/SAMreport.sunfire.20060530.1247CDT.tar.gz
     Lines per file:  1000
     Output format:   tar.gz (default) Use -u for unarchived/uncompressed.
     Please wait...........................................
     Please wait...........................................
     Please wait....................................

     The following files should now be ftp'ed to your support provider
     as ftp type binary.

     /tmp/SAMreport.sunfire.20060530.1247CDT.tar.gz

     sunfire# samexplorer -u

     Report name:     /tmp/SAMreport.sunfire.20060530.1252CDT
     Lines per file:  1000
     Output format:   unarchived/uncompressed

     Please wait...........................................
     Please wait...........................................
     Please wait....................................

     The following files should now be ftp'ed to your support provider
     as ftp type binary.

     /tmp/SAMreport.sunfire.20060530.1252CDT
     /tmp/SAMreport.sunfire.20060530.1252CDT.fsmgr_text
     /tmp/SAMreport.sunfire.20060530.1252CDT.dmpshm_data
     /tmp/SAMreport.sunfire.20060530.1252CDT.samtrace_text
     /tmp/SAMreport.sunfire.20060530.1252CDT.showqueue_text
     /tmp/SAMreport.sunfire.20060530.1252CDT.archiver_data.tar
     /tmp/SAMreport.sunfire.20060530.1252CDT.stager_data.tar
```

samexport(1M)

```
NAME
     export, samexport - Export a cartridge from a robot

SYNOPSIS
     /opt/SUNWsamfs/sbin/export [-f] eq:slot
     /opt/SUNWsamfs/sbin/export [-f] mediatype.vsn
     /opt/SUNWsamfs/sbin/samexport [-f] eq:slot
     /opt/SUNWsamfs/sbin/samexport [-f] mediatype.vsn

AVAILABILITY
     SUNWsamfs

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

OPTIONS
     -f   The -f option is used for  network-attached  StorageTek
          automated  libraries only. The -f option will cause the
          volume specified to be exported to the  CAP  (Cartridge
          Access  Port)  and  the SAM-QFS catalog updated accord-
          ingly. The CAPID must be defined in the stk  parameters
          file.  See  the stk(7) man page for details on defining
          the CAPID.

     For the network-controlled libraries such as the GRAU  using
     the  GRAU  ACI  interface,  IBM 3494, or STK libraries using
     ACSLS and not specifying the -f option,  this  utility  only
     removes  the  catalog entry for the cartridge from the cata-
     log. Physical removal  and  addition  of  cartridges  within
     these  libraries is performed by utilities supplied by GRAU,
     IBM, and STK.

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

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

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

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

FILES
     mcf                 The  configuration  file   for   SAM-QFS
                         environments

SEE ALSO
     import(1M),  build_cat(1M),  dump_cat(1M),  sam-robotsd(1M),
     mcf(4), stk(7), historian(7)
```

samfsck(1M)

```
NAME
     samfsck - Checks and repairs a Sun QFS or SAM-QFS file  sys-
     tem

SYNOPSIS
     samfsck [ -s scratch_dir ] [ -F [ -R ] ] [ -G ] [ -S ] [  -U
     ] [ -u fs_version ] [ -V ] [ -p ] [ -A ] fs_name

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The samfsck command checks and optionally repairs a Sun  QFS
     or  SAM-QFS file system from the disk partitions that belong
```

to fs_name. For fs_name, specify either a family set name from the mcf file or a mount point absolute path name from the /etc/vfstab 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. To repair the file system, the file system must be unmounted, and the -F option specified.

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
: Specifies 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. If the file system is not unmounted samfsck will exit with an error.

-G
: Generate directory entry hash. In SAM-FS 3.5.0 and above, a hash code was added to directory entries to speed up directory searches. This is particularly useful for longer file names. The -G option, when used in conjunction with the -F option, will modify directory entries which do not have a proper hash value to have a hash. When the -G option is used without the -F option, the number of directory entries which could be hashed is reported. The presence of a hash value has no effect on versions of SAM-FS prior to 3.5.0.

-S
: Convert the filesystem from a non-shared filesystem to a shared filesystem. This option is not available to filesystems with a version 1 superblock. The -F option must also be specified to convert a filesystem. This will cause samfsck to update the on-disk structures to make the filesystem shared. Note that samfsck does not

```
          update the /etc/vfstab entry (see vfstab(4)),  the  mcf
          entry  (see  mcf(4)),  or  the  shared  hosts file (see
          samsharefs(1M)) for the filesystem, nor does it config-
          ure  the services file (see services(4)) for shared SAM
          operations.   These  must  be  configured  and  updated
          before the filesystem is converted.

     -U   Convert the filesystem from a shared  filesystem  to  a
          non-shared  filesystem.   The  -F  option  must also be
          specified to convert a filesystem.  The on-disk  struc-
          tures  of  the  filesystem  are  updated  to  make  the
          filesystem non-shared.   Note  that  samfsck  does  not
          update  the  /etc/vfstab  entry (see vfstab(4)), or the
          mcf entry (see mcf(4)).  These must be  configured  and
          updated before the filesystem is converted.

     -u  fs_version
          Convert the filesystem to the  given  fs_version.   The
          only  value that is valid for 5.0 is 2A.  Only Filesys-
          tems version 2 can be  converted  to  2A.   Filesystems
          version 2A can use 5.0 features like Online add/remove,
          Large  Host  Table  and  Project  IDs.   Note  that  2A
          filesystems are only mountable on 5.0 and not backwards
          compatible.  The -F option must also  be  specified  to
          convert  a  filesystem.   The on-disk structures of the
          filesystem are updated to make the  filesystem  version
          2A.   Note that version 2A filesystems are not backward
          compatible or reversible.

     -V   Turns on a verbose display of DEBUG  information.  This
          information is useful to Oracle analysts.

     -R   Rename the file system.  When specified along with  the
          -F  option,  the -R option will rewrite the super block
          with  the  disk  cache  family  set   name   found   in
          /etc/opt/SUNWsamfs/mcf.  No action will be taken if the
          -R option is used without the -F option. It  is  impor-
          tant  that  sam-fsd  be  notified  after  any change to
          /etc/opt/SUNWsamfs/mcf (see samd(1M)).

     -p   Return an indication of the filesystem’s health.   Non-
          zero return indicates that the filesystem should not be
          mounted without first using samfsck to check and repair
          the  filesystem (see EXIT STATUS).  A zero return value
          indicates that the filesystem can  be  mounted  immedi-
          ately.

     -A   Convert the POSIX exteneded ACL to the  NFSV4  extended
          ACL.   This  operation  is  not  reversible,  and  only
          applies to file system versions  V2  or  V2A.   The  -F
          option  must also be specified to convert a filesystem.
          NOTE: This option is only available for Solaris release
          after Solaris 10.

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  pro-
               cessing. 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.

     41        Fatal: Filesystem option  mask  has  non-backwards
               compatible options.

     45        Fatal: Filesystem .inodes file is invalid.

     50        Fatal: I/O Errors terminated samfsck.

     55        Nonfatal:  The -p option was  specified,  and  the
               filesystem should be checked and repaired prior to
               mounting.

FILES
     /etc/opt/SUNWsamfs/mcf
                         The configuration file for samfs

     /etc/vfstab         File system defaults table

SEE ALSO
     samd(1M).  samsharefs(1M).

     mcf(4), services(4), vfstab(4).

Caret>
```

samfsconfig(1M)

```
NAME
     samfsconfig - Recovers configuration information

SYNOPSIS
     /opt/SUNWsamfs/sbin/samfsconfig [-b] [-d] [-h] [-s] [-v]
     device [device] ...
```

```
AVAILABILITY
     SUNWqfs
     SUNWsamfs

DESCRIPTION
     The samfsconfig utility opens the device(s) listed on the
     command line, attempts to read the Sun QFS file system
     superblock on each, and generates output in a format similar
     to an editable mcf(4) file.  A Sun QFS file system
     superblock is a record that the sammkfs(1M) utility writes
     to the beginning of every device in a Sun QFS file system.
     This record identifies the devices to the file system.

     By default, the output is written to stdout, but the output
     can be redirected to a file and edited to regenerate the
     file system portions of the mcf file in the event of a
     system reconfiguration or disaster.

     samd buildmcf executes samfsconfig to build the mcf file.
     samd config must be executed to reconfigure the changes
     after the samd buildmcf has built the mcf file.

OPTIONS
     This command accepts the following options:

     -b        Lists the size of the associated partition,
               according to its superblock, in the last output
               column.  This may be useful when multiple disk
               partitions of different sizes start at the same
               offset.

     -d        Generates detailed information about all the Sun
               QFS superblocks found, including the content of
               each superblock.

     -h        Generates a usage message and exits.

     -s        Print the host file contents of QFS shared
               filesystems.

     -v        Generates messages regarding the disposition of
               each device.

     device    One or more device identifiers from which
               configuration information is to be recovered.  Use
               a space character to separate multiple device
               identifiers on the command line.

               It can be desireable to save a list of device
               identifiers to a file and use this file for
               command line input to the program.

     The samfsconfig utility generates information about all the
     Sun QFS file systems and file system components it finds.
     The file system name, creation time, generation, devices
     count, and metadata devices count are listed.  The
     samfsconfig utility flags irregularities as follows:
```

o For any incomplete devices that have superblocks, but are in an file system where all the devices are not present, it prefixes a pound sign (#) to indicate problems.

o For any duplicate devices that have the same superblock information with the same file system generation number, it prefixes a greater-than sign (>). This is common if, for instance, multiple paths exist or whole disk partitions are specified on the command line.

o For devices, one /dev/did/dsk and the other device /dev/dsk, which have the the same superblock information with the same file system generation number, it prefixes a greater-than sign (>) for the /dev/dsk devices. This occurs for did file systems. This line is only generated when the -v option is specified.

o For any duplicate devices that have the same superblock information but have different file system generation numbers, it prefixes a less-than sign (<). This can occur if a device has been removed from the file system and a new device has been added to the same file system in the same ordinal position. Note, the file system generation number is incremented when a device is added to the file system. This line is only generated when the -v option is specified.

EXAMPLES

Example 1.

```
ceres# samfsconfig /dev/dsk/*
#
# Family Set ’samfs5’ Created Fri Aug 29 12:05:15 2008
# Generation 0 Eq count 7 Eq meta count 3
#
# zoned-off or missing metadata device
#
# Missing slices
# nodev        11    mm   samfs5  -
# Ordinal 1
# /dev/dsk/c6t600A0B80002AC18A000006A048A1A0BEd0s1    12    mr   samfs5  -

#
# Family Set ’samfs5’ Created Wed Sep 17 01:56:27 2008
# Generation 0 Eq count 7 Eq meta count 3
#
# zoned-off or missing metadata device
#
# Missing slices
# nodev        12    mm   samfs5  -
# nodev        13    mm   samfs5  -
# Ordinal 0
# /dev/dsk/c6t600A0B80002AC18A000006A648A1A0F6d0s2    11    mm   samfs5  -

#
# Family Set ’samfs5’ Created Fri Dec 19 20:17:48 2008
# Generation 0 Eq count 2 Eq meta count 1
#
# zoned-off or missing metadata device
```

```
#
# Missing slices
# nodev        501    mm   samfs5  -
# Ordinal 9

#
# Family Set ’samfs5’ Created Tue Dec 23 16:41:07 2008
# Generation 1 Eq count 5 Eq meta count 1
#
# Missing slices
# Ordinal 0
# /dev/dsk/c6t600A0B80002AC18A000006A048A1A0BEd0s6    501    mm   samfs5  -
# Ordinal 1
# /dev/dsk/c6t600A0B80002AC18A000006A348A1A0DAd0s6    502    mr   samfs5  -
# Ordinal 2
# /dev/dsk/c6t600A0B80002AC18A000006A648A1A0F6d0s6    503    mr   samfs5  -
# Ordinal 3
# /dev/dsk/c6t600A0B80002AC18A0000069D48A1A0A2d0s6    504    mr   samfs5  -

#
# Family Set ’samfs1’ Created Sat Dec 27 19:56:26 2008
# Generation 0 Eq count 4 Eq meta count 1
#
# Foreign byte order (super-blocks byte-reversed).
#
# Missing slices
# Ordinal 0
# /dev/dsk/c6t600A0B80002AC18A000006A048A1A0BEd0s0    101    mm   samfs1  -

#
# Family Set ’samfs5’ Created Sun Jan  4 11:15:00 2009
# Generation 2 Eq count 7 Eq meta count 2
#
# zoned-off or missing metadata device
#
# Missing slices
# nodev        501    mm   samfs5  -
# Ordinal 5
# /dev/dsk/c6t600A0B80002AC18A000006A648A1A0F6d0s4    506    mr   samfs5  -
# Ordinal 7
# /dev/dsk/c6t600A0B80002AC18A000006A048A1A0BEd0s2    508    mr   samfs5  -

#
# Family Set ’stand’ Created Thu Jan  8 20:23:11 2009
# Generation 0 Eq count 1 Eq meta count 0
#
stand 100 ms stand -
/dev/dsk/c6t600A0B80002AC18A000006A048A1A0BEd0s3    101    md   stand  -

#
# Family Set ’samfs6’ Created Fri Jan  9 21:27:48 2009
# Generation 0 Eq count 1 Eq meta count 0
#
samfs6 600 ms samfs6 -
> /dev/dsk/c0t1d0s2    601    md   samfs6  -
> /dev/dsk/c0t1d0s5    601    md   samfs6  -

#
# Family Set ’samfs5’ Created Fri Jan  9 21:32:28 2009
# Generation 2 Eq count 7 Eq meta count 2
#
samfs5 500 ma samfs5 - shared
/dev/dsk/c6t600A0B80002AC18A000006A348A1A0DAd0s5    501    mm   samfs5  -
/dev/dsk/c6t600A0B80002AC18A000006A648A1A0F6d0s5    502    mr   samfs5  -
/dev/dsk/c6t600A0B80002AC18A0000069D48A1A0A2d0s5    503    mm   samfs5  -
/dev/dsk/c6t600A0B80002AC18A0000069D48A1A0A2d0s3    504    mr   samfs5  -
/dev/dsk/c6t600A0B80002AC18A000006A648A1A0F6d0s3    505    mr   samfs5  -
/dev/dsk/c6t600A0B80002AC18A000006A348A1A0DAd0s4    506    mr   samfs5  -
/dev/dsk/c6t600A0B80002AC18A0000069D48A1A0A2d0s4    507    mr   samfs5  -

Example 2.  Another example, this from a saved list of
devices:

ceres# samfsconfig -v ‘cat /tmp/dev_files‘
Device ’/dev/dsk/c6t600A0B80002AC18A000006A348A1A0DAd0s5’ has a QFS superblock.
Device ’/dev/dsk/c6t600A0B80002AC18A000006A648A1A0F6d0s5’ has a QFS superblock.
Device ’/dev/dsk/c6t600A0B80002AC18A0000069D48A1A0A2d0s5’ has a QFS superblock.
Device ’/dev/dsk/c6t600A0B80002AC18A0000069D48A1A0A2d0s3’ has a QFS superblock.
Device ’/dev/dsk/c6t600A0B80002AC18A000006A648A1A0F6d0s3’ has a QFS superblock.
Device ’/dev/dsk/c6t600A0B80002AC18A000006A348A1A0DAd0s3’ has a QFS superblock.
Device ’/dev/dsk/c6t600A0B80002AC18A000006A348A1A0DAd0s4’ has a QFS superblock.
Device ’/dev/dsk/c6t600A0B80002AC18A0000069D48A1A0A2d0s4’ has a QFS superblock.
Device ’/dev/dsk/c6t600A0B80002AC18A000006A348A1A0DAd0s2’ has a QFS superblock.

Device ’/dev/dsk/c6t600A0B80002AC18A000006A648A1A0F6d0s4’ has a QFS superblock.
10 QFS devices found.
#
# Family Set ’samfs5’ Created Fri Dec 19 20:17:48 2008
# Generation 2 Eq count 7 Eq meta count 2
#
# zoned-off or missing metadata device
#
# Missing slices
# nodev        501    mm   samfs5  -
# Ordinal 9
< /dev/dsk/c6t600A0B80002AC18A000006A348A1A0DAd0s2    512    g7   samfs5  -

#
# Family Set ’samfs5’ Created Sun Jan  4 11:15:00 2009
# Generation 2 Eq count 7 Eq meta count 2
#
# zoned-off or missing metadata device
#
# Missing slices
# nodev        501    mm   samfs5  -
# Ordinal 5
# /dev/dsk/c6t600A0B80002AC18A000006A648A1A0F6d0s4    506    mr   samfs5  -

#
# Family Set ’samfs5’ Created Fri Jan  9 21:32:28 2009
# Generation 2 Eq count 7 Eq meta count 2
#
samfs5 500 ma samfs5 - shared
/dev/dsk/c6t600A0B80002AC18A000006A348A1A0DAd0s5    501    mm   samfs5  -
/dev/dsk/c6t600A0B80002AC18A000006A648A1A0F6d0s5    502    mr   samfs5  -
/dev/dsk/c6t600A0B80002AC18A0000069D48A1A0A2d0s5    503    mm   samfs5  -
/dev/dsk/c6t600A0B80002AC18A0000069D48A1A0A2d0s3    504    mr   samfs5  -
/dev/dsk/c6t600A0B80002AC18A000006A648A1A0F6d0s3    505    mr   samfs5  -
```

```
          < /dev/dsk/c6t600A0B80002AC18A000006A348A1A0DAd0s3    506    mr   samfs5  -
          /dev/dsk/c6t600A0B80002AC18A000006A348A1A0DAd0s4    506    mr   samfs5  -
          /dev/dsk/c6t600A0B80002AC18A0000069D48A1A0A2d0s4    507    mr   samfs5  -
```

SEE ALSO
     sammkfs(1M) samd(1M)

     mcf(4)

samfsdump(1M)

NAME
     samfsdump, samfsrestore - Dumps or restores SAM-QFS file
     control structure data

SYNOPSIS
```
     samfsdump [-b bl_factor] [-d] -f dump_file [-n] [-q] [-P]
     [-u] [-U] [-v] [-B size] [-H] [-I include_file] [-S] [-T]
     [-W] [-X excluded_dir] [-Y] [-Z db_loadfile]
     [file ...]

     samfsrestore [-b bl_factor] [-d] -f dump_file [-g log_file]
     [-i] [-l] [-r] [-s] [-t] [-v] [-B size] [-H] [-R] [-S] [-T]
     [-Z db_loadfile] [-2] [file ...]
```

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The samfsdump command creates a dump file containing control
     structure information for each specified file. This command
     must be entered after you have used the cd(1) command to
     change to the mount point of a SAM-QFS file system.

     The samfsdump command creates a dump file, as follows:

     o If nothing is specified for file, the samfsdump command
       creates a dump file containing the control structures for
       every file in the current directory and also for every
       file in the current directory’s subdirectories.

     o If an individual file is specified for file, the samfsdump
       command creates a dump file containing the control
       structures for that individual file.

     o If a directory is specified for file, the samfsdump
       command creates a dump file containing the control
       structures for every file in that directory and also for
       every file in that directory’s subdirectories.

     Any file specified with an absolute path is stored in the
     dump file with an absolute path. Any file specified with a
     relative path is stored in the dump file with its relative
     path.

     The samfsrestore command uses the contents of the dump file to restore control structures for all the files in the dump file or for each specified file. If a file is specified, its path and file name must match exactly what exists in the dump file. By default, all files are restored to the absolute or relative location as each file is described in the dump file. If the -s option is specified, however, all file names with an absolute path in the dump file are restored relative to the current directory, using the entire path as contained in the dump file.

The samfsdump command does not create a dump of any data associated with the files (unless the -P, -u or -U options are specified), so no data can be restored from this dump file. It is assumed that the data associated with the dumped files has been archived in some way. If a file for which no archive copy is available is dumped, a warning message is issued noting that this file will be marked as damaged when restored. When that file is restored from the dump file, it is marked as damaged by samfsrestore. Note that this warning can be explicitly suppressed by using the -q option.

If dump file contains ACLs, they could be either of POSIX ACLs or NFSv4 ACLs. Each type of ACL would normally be restored to the filesystem supporting that type of ACL. If the dump file contains NFSv4 ACLs and the filesystem supports POSIX ACLs, or the dump file contains POSIX ACLs and the filesystem supports NFSv4 ACLs, no conversion will be performed, a warning will be issued, and files will be restored with empty ACLs.

You must be logged in as superuser (root) in order to execute the samfsdump and samfsrestore commands. Sun Microsystems recommends that a site create samfsdump dumps on a periodic basis as part of a disaster recovery plan.

OPTIONS

This command accepts the following options:

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

-d
: Enables debugging messages. This option is useful only to Oracle Corporation and is used to trace execution for verification purposes.

-f dump_file
: Names the file to which the control structure data dump is written to (by samfsdump) or read from (by samfsrestore). You must specify a dump_file.

  If a dash character (-) is specified for the dump_file, samfsdump writes the dump file to stdout and samfsrestore reads the dump file from stdin.

```
               The dump file data can be passed through
               appropriate filters, such as compression or
               encryption, after being written by samfsdump or
               before being read by samfsrestore.

     -g log_file
               (samfsrestore only) Generates a file of online
               directories and files.  For information on the
               format of this file, see the NOTES section of this
               man page.

     -i        (samfsrestore only) Prints the inode numbers of
               the files when listing the contents of the dump.
               For more listing options, see -l, -t, and -2
               options.

     -I include_file
               (samfsdump only) Takes the list of files to dump
               from include_file.  This file has one relative or
               absolute path to be dumped per line.  After
               processing include_file, any [file] arguments from
               the command line are processed.

     -l        (samfsrestore only) Prints one line per file.
               This option is similar to the sls(1M) command's -l
               option when listing the dump contents.  Note that
               this option is identified by the lowercase letter
               'l', not a number '1'.  For more listing options,
               see the -i, -t, and -2 options.

     -n        (Obsolete. samfsdump only.) Always uses the new
               header format.  The new header is incompatible
               with samfsrestore prior to the 3.5.0 release
               level.

     -P        (samfsdump only) Dumps the online data portions of
               files which are offline, but have partial data
               online.  This option can considerably increase the
               size of the dump file, as data and metadata are
               both being dumped.  You must take care to manage
               the increased size of the dump.  This option can
               be used to move file partial data by piping the
               output of samfsdump to the input of samfsrestore.

     -q        (samfsdump only) Suppresses warning messages for
               damaged files.  By default, samfsdump writes
               warning messages for each file that would be
               considered damaged if the dump were restored.

     -r        (samfsrestore only) Replaces existing files when
               restoring control structures if the existing files
               have an older modification time than the dumped
               files.

     -s        (samfsrestore only) Removes leading slashes from
               file names prior to restoring them.  This is
               useful if the dump was made with an absolute path
               name and the dump is being restored to a different
               location.  Any directories required for the
```

```
          restoration and not defined in the dump file are
          automatically created.

-t        (samfsrestore only) Lists the content of the dump
          file rather than restoring the dump.  For more
          listing options, see the -i, -l, and -2 options.

-u        (samfsdump only) Dumps the data portions of files
          without at least one archive copy.  This option
          can considerably increase the size of the dump
          file, as data and metadata are both being dumped.
          You must take care to manage the increased size of
          the dump.

-U        (samfsdump only) Dumps the data portions of files
          which are online.  This option can considerably
          increase the size of the dump file, as data and
          metadata are both being dumped.  If this option is
          used with segmented files, the archive copy
          information is not preserved when the file is
          restored.  You must take care to manage the
          increased size of the dump.  This option can be
          used to move file systems by piping the output of
          samfsdump to the input of samfsrestore.

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

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

-H        For samfsdump, creates the dump file without a
          dump header record.  For samfsrestore, declares
          that the existing dump file has no header record.
          This option can be used to create control
          structure dump files that can be concatenated
          using the cat command.  For more information on
          this command, see the cat(1) man page.

-R        (samfsrestore only) Replaces existing files when
          restoring control structures.

-S        Perform only a scan to create a db_loadfile with
          the -Z option.  When using -S during samfsdump, no
          dump file is created and -f is not needed.  During
          samfsrestore, -S used with -Z will create a
          db_loadfile from the dump file specified by -f and
          no restore is performed.

-T        Displays statistics at command termination.  These
          statistics include the number of files and
          directories processed, the number of errors and
          warnings, and other information.  Example:

          samfsdump statistics:
                        Files:              52020
```

```
              Directories:        36031
              Symbolic links:     0
              Resource files:     8
              File segments:      0
              File archives:      0
              Damaged files:      0
              Files with data:    24102
              File warnings:      0
              Errors:             0
              Unprocessed dirs:   0
              File data bytes:    0
```

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

File segments refers to the number of data segments associated with segmented files from the dump.

File archives refers to the number of archive images associated with the preceding Files, Directories, Symbolic links, and Resource files.

Damaged files refers to the number of Files, Directories, Symbolic links, and Resource files that are either already marked damaged (for a samfsdump) or were damaged during a restore because they had no archive image (for a samfsrestore).

Files with data refers to the number of Files that have online (full or partial) data dumped or restored.

File warnings refers to the number of Files,

Directories, Symbolic links, and Resource files that would be damaged should the dump be restored because they had no archive images at the time of the dump.

Errors refers to the number of error messages that were printed during the dump or restore. These errors indicate a problem, but the problem is not severe enough to cause an early exit from samfsdump or samfsrestore. Examples of errors during a restore are failing to create a symbolic link and failing to change the owner or group of a file. Errors that might occur during a dump include having a path name too long, failing to open a directory for reading, failing to read a symbolic link or resource file, or finding a file with an invalid mode.

Unprocessed dirs refers to the number of directories that were not processed due to an

```
              error, such as being unable to create the
              directory.

              File data bytes refers to the size of data that
              was dumped (using options -P, -U, or -u) or
              restored.

     -W       (Obsolete. samfsdump only.)  Writes warning
              messages during the dump process for files that
              would be damaged if the dump were restored.  This
              option is retained for compatibility.  By default,
              these warning messages are now issued
              automatically.  For more information on
              controlling this behavior, see the -q option,
              which suppresses warning messages.

     -X excluded_dir
              (samfsdump only) Specifies directory paths to be
              excluded from the dump.  Relative paths without
              leading characters must be used, for example
              dir1/dir2.  The result is an empty directory
              dir1/dir2 in the dump file.  A directory that
              resolves to . or NULL generates an error message.
              Multiple (up to 10) directories can be excluded by
              using multiple -X options.

     -Y       (samfsdump only) Specifies that the trailing list
              of files are lists of files to dump.  Using this
              option helps improve samfsdump performance by
              reducing the number of path lookups.  If - is
              specified as the trailing list, standard input is
              used.

              Each list must have one line per file, with tab
              separated inode number, generation number, and
              file path.  The path must is relative to where
              samfsdump is executed.

              Example line: 1039 11 testdir2/rtest_f_61

              Example usage: samfsdump -Y -f samfs1.dump
              /path/to/filelist

              Example pipelined: samdb dump samfs1 | samfsdump
              -Y -f samfs1.dump -

              If a sideband mysql database is being used by the
              target SAM filesystem, then the file list can be
              generated using the samdb(1M) dump command.

     -Z db_loadfile
              Specifies that a samdb(1M) db_loadfile should be
              created as part of a samfsdump or samfsrestore.
              This file is used to populate a sideband mysql
              database using the samdb(1M) load command.

              Use the -S option to only produce the db_loadfile
              without performing the usual samfsdump or
              samfsrestore operations.  If - is specified for
```

```
               the load file standard output is used.

     -2        (samfsrestore only) Writes two lines per file,
               similar to the sls(1) command’s -2 option, when
               listing the contents of the dump.  For more
               listing options, see the -i, -l, and -t options.

     file ...  Lists files to be dumped or restored.  Note that
               the names given to restore must match exactly the
               names as they are stored in the dump.  You can use
               samfsrestore -t to see how the names are stored.
```

NOTES

A samfsrestore should not be attempted on a Sun QFS shared file system client.

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

If the -g option is used, a log file is generated during file system restoration. This file contains one line per file that was online, or partially online, at the time the file was dumped. This line is divided into fields and contains the following information:

Field	Description
1	The file type, which is indicated by one of the following letters: o d indicates a directory. o f indicates a regular file. o l indiactes a symbolic link. o R indicates a removable media file. o I indicates a segment index. o S indicates a data segment.
2	The media type and Volume Serial Name (VSN) in media_type.vsn format.
3	The position on the media.
4	Either online or partial.
5	The path relative to the file system mount point.

After a samfsrestore command is issued, it is possible to restore files that were online, prior to the dump, back to their online state. You do this by using the script in /opt/SUNWsamfs/examples/restore.sh.

EXAMPLES

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

```
     example# cd /sam
     example# samfsdump -f /destination/of/the/dump/samfsdump.today
```

To restore a control structure dump to /sam:

```
     example# cd /sam
     example# samfsrestore -f /source/of/the/dump/samfsdump.yesterday
```

     To create a new samdb(1M) database load file of /sam:

```
     example# cd /sam
     example# samfsdump -SZ /destination/samfsdbload.today
```

     To create a dump of /sam using a list of files:

```
     example# cd /sam
     example# samfsdump -Y -f /destination/of/samfsdump.today /source/of/samfslist.today
```

     To create a new samdb(1M) load file from an existing dump file:

```
     example# samfsrestore -SZ /destination/samfsdbload.today -f /source/samfsdump.yesterday
```

SEE ALSO
     cat(1), sls(1), samdb(1M).

DIAGNOSTICS
     You may encounter messages while using the samfsdump or samfsrestore command. The following list shows several possible messages and their explanations:

```
     Message             Explanation

     file: Unrecognised mode (0x..)
                         samfsdump is being asked to dump a file
                         that is not a regular file, directory,
                         symbolic link, or removable media file.
                         The Sun QFS and SAM-QFS file systems
                         allow the creation of block special,
                         character special, fifo, and other
                         special files, but they do not function
                         correctly.  samfsdump does not attempt
                         to dump them.

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

     file: Warning! File is already damaged.
                         If received during a samfsdump, means
                         that the file is currently marked
                         damaged.  During restoration, the file
                         will still be damaged.

     file: File was already damaged prior to dump
                         If received during a samfsrestore, this
                         means that the file was dumped with the
                         damaged flag set.
     file: File is now damaged
                         If received during a samfsrestore, this
                         means that the file was dumped when it
                         had no archive images.  samfsdump and
                         samfsrestore do not dump file data.
                         They rely on the file’s data having been
                         archived.  Because the file no longer
                         has any data associated with it, it is
                         marked damaged.

     .: Not a SAM-FS file.
                         You are attempting to dump files from a
                         file system that is not a Sun QFS or
                         SAM-QFS file system, or you are
                         attempting to restore files from a
                         samfsdump dump file into a file system
                         that is not a Sun QFS or SAM-QFS file
                         system.

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

     Corrupt samfsdump file.  name length %d
                         If received during a restore, this means
                         that the path name of a file to be
                         restored was less than zero or larger
                         than MAXPATHLEN.  This should not occur.
                         samfsrestore aborts.

     Corrupt samfsdump file. %s inode version incorrect
                         During a restore, this means that a the
                         inode for the indicated file was in an
                         old format.  This should not occur.
                         samfsrestore aborts.

     file: pathname too long
                         If received during a dump, this
                         indicates that the path name of the
                         indicated file is longer than 1024
                         characters.  The file is not dumped.
```

samfsinfo(1M)

```
NAME
     sammkfs, samfsinfo - Constructs or displays information for
     a Sun QFS or SAM-QFS file system

SYNOPSIS
     /opt/SUNWsamfs/sbin/sammkfs [-a allocation_unit] [-i inodes]
     [-A] [-P] [-S] [-V] fs_name

     /opt/SUNWsamfs/sbin/samfsinfo fs_name

AVAILABILITY
     SUNWqfs

     SUNWsamfs

DESCRIPTION
     The sammkfs command creates a Sun QFS or SAM-QFS file system
     from the disk partitions that belong to the family set
     fs_name, where fs_name is the family set name as defined in
     the mcf file.  Up to 252 disk partitions can be specified in
     the mcf file for a Sun QFS or SAM-QFS file system.  The
     sammkfs command can also be used to recreate a file system
     after a disaster.

     The sammkfs command can create either a version 2 file
     system that is backwards compatible with previous releases,
     or a version 2A file system that has new features, but is
     not compatible with previous releases.  By default, a
     version 2A file system is created.  See -P parameter below
     for details on the new features, and how to create a version
     2 file system.

     The sammkfs command aligns the block allocation bit maps and
     round robins them on the metadata devices for improved
     performance.  This behavior is  backwards compatible with
     previous releases.  The option feature Aligned Maps is set.

     The samfsinfo command displays the structure of an existing
     Sun QFS or SAM-QFS file system.  The output is similar to
     that obtained by using the -V option to the sammkfs command.

OPTIONS
     These commands accept the following options:

     -a allocation_unit
               Specifies the disk allocation unit (DAU).  The DAU
               is the basic unit of online storage.  When you
               specify a DAU size, you specify the number of
               1024-byte (1 kilobyte) blocks to be allocated for
               a file.

               The DAU size you can specify depends on the type
               of file system being initialized, as follows:

               o  The SAM-QFS file system is an ms file system.
                  The disk devices in it are all md devices.
                  Both data and metadata are written to the md
                  devices.  The allocation_unit specifies the DAU
                  to be used for the md devices.  Possible
                  allocation_unit specifications are 16, 32, or
                  64 (the default).

               o  The Sun QFS or SAM-QFS file systems are ma file
                  systems.  The metadata in these file systems is
                  written to mm devices.  The disk devices in
                  these file systems are specified as either md,
                  mr, or gXXX devices, as follows:

                  -  For the md devices, possible allocation_unit
                     specifications are 16, 32, or 64 (the
                     default).  A single file system cannot have
                     md devices mixed among the mr and gXXX
                     devices.

                  -  For mr devices, the DAU is fully adjustable.
                     Specify an allocation_unit that is a
                     multiple of 8 in the following range for mr
                     devices:  8 < allocation_unit < 65528.  The
                     default is 64.

                  -  For gXXX devices, which specify striped
                     groups, the DAU is fully adjustable.  If the
                     file system contains striped groups, the
                     minimum unit of disk space allocated is the
                     DAU multiplied by the number of members in
                     the striped group.  Specify an
                     allocation_unit that is a multiple of 8 in
                     the following range for gXXX devices:
                     8 < allocation_unit < 65528.  The default is
                     256.

                  You can mix mr and gXXX devices in a single Sun
                  QFS or SAM-QFS file system.  If these device
                  types are mixed, the allocation_unit specified
                  is used for both device types.  If no
                  allocation_unit is specified, the DAU size used
                  for each type of device is 256.

     -i inodes Specifies the number of inodes to be allocated for
               this file system.  This is the total number of
               user inodes that can be used for the life of this
               file system. In Sun QFS and SAM-QFS version 2
               superblock file systems, a number of inodes are
               reserved for file system usage, and are
               unavailable to the user. This number is in
               addition to the specified number of user inodes.
               The actual number of inodes available vary from
               that specified, due to rounding to metadata DAU
               size.

               NOTE:  By specifying this option, you eliminate
               the possibility of ever increasing the number of
               inodes for the file system.  Therefore, Sun does
               not recommend the use of this option.
```

```
                    When this option is specified, later use of the
                    samgrowfs(1M) command increases the size of the
                    file system, but it cannot increase the number of
                    allowable inodes.  For more information on
                    enlarging file systems, see the WARNINGS section
                    of this man page and the samgrowfs(1M) man page.

          -A        Uses NFSv4 ACL style for the filesystem ACLs
                    instead of POSIX ACL style. This feature is
                    available only in releases of Solaris beyond
                    Solaris 10.

          -P        Specifies that a previous version of the file
                    system be created.  This version creates a version
                    2 superblock and is compatible with SAM-QFS
                    version 4.6.  This version cannot use the
                    following features however:  large host table,
                    extended attributes, and online grow.  Without the
                    -P parameter, a version 2A superblock is created,
                    the above features are available, and the file
                    system is not usable with SAM-QFS version 4.6 or
                    previous.

          -S        Indicates that this file system is shared.  In
                    order to mount the file system as a Sun QFS shared
                    file system, you must also create a hosts.fs_name
                    configuration file.  For more information on this
                    configuration file and other aspects of the Sun
                    QFS shared file system, see the Sun QFS File
                    System Configuration and Administration Guide.
                    For information on configuring a hosts file, see
                    the hosts.fs(4) man page.

          -V        Writes configuration information to standard
                    output but does not execute the sammkfs command.
                    This information can be used to create a new file
                    system.

                    The samfsinfo command should be used to generate
                    configuration information for an existing file
                    system.

     EXAMPLES
          Example 1.  The following command creates SAM-QFS file
          system with a DAU size of 128 kilobytes:

          server#  sammkfs -a 128 samfs1

     FILES
          /etc/opt/SUNWsamfs/mcf   The configuration file for a Sun
                                   QFS or SAM-QFS file system

     WARNINGS
          As with creating any type of file system, if you specify the
          wrong partition names, you risk damaging user or system
          data.  Be sure to specify partitions that are otherwise
          unused on your system.  Do not use overlapping partitions.

          With SAM-QFS 4.1 and greater AND Solaris 64bit kernels which
```

support large disk devices (greater than 1 TB), it is possible to have partitions that are greater than 1 TB. Note that these file systems are not usable on Solaris systems that do not support large disk devices.

SEE ALSO

dd(1M), samd(1M), samgrowfs(1M), undamage(1M).

mcf(4).

Sun QFS File System Configuration and Administration Guide.

Sun Storage Archive Manager Configuration and Administration Guide.

WARNINGS

Be careful when using the -i inodes option for this command. By using this option, you dictate the maximum number of inodes allowed for the life of this file system. This eliminates the possibility of ever using the samgrowfs(1M) command to increase the number of files in this file system. After a file system is made with -i specified, the samgrowfs(1M) command can only be used to increase the size of the file system in terms of bytes.

NOTES

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

For example, if a RAID-5 unit has a total of 8 disks with 1 of the 8 being the parity disk, the number of data disks is 7. If the RAID stripe width is 64 kilobytes, then the optimal allocation_unit is 64 * 7 = 448.

samfsrestore(1M)

NAME

samfsdump, samfsrestore - Dumps or restores SAM-QFS file control structure data

SYNOPSIS

```
samfsdump [-b bl_factor] [-d] -f dump_file [-n] [-q] [-P]
[-u] [-U] [-v] [-B size] [-H] [-I include_file] [-S] [-T]
[-W] [-X excluded_dir] [-Y] [-Z db_loadfile]
[file ...]

samfsrestore [-b bl_factor] [-d] -f dump_file [-g log_file]
[-i] [-l] [-r] [-s] [-t] [-v] [-B size] [-H] [-R] [-S] [-T]
[-Z db_loadfile] [-2] [file ...]
```

AVAILABILITY

SUNWsamfs

DESCRIPTION

The samfsdump command creates a dump file containing control structure information for each specified file. This command must be entered after you have used the cd(1) command to change to the mount point of a SAM-QFS file system.

The samfsdump command creates a dump file, as follows:

- If nothing is specified for file, the samfsdump command creates a dump file containing the control structures for every file in the current directory and also for every file in the current directory's subdirectories.

- If an individual file is specified for file, the samfsdump command creates a dump file containing the control structures for that individual file.

- If a directory is specified for file, the samfsdump command creates a dump file containing the control structures for every file in that directory and also for every file in that directory's subdirectories.

Any file specified with an absolute path is stored in the dump file with an absolute path. Any file specified with a relative path is stored in the dump file with its relative path.

The samfsrestore command uses the contents of the dump file to restore control structures for all the files in the dump file or for each specified file. If a file is specified, its path and file name must match exactly what exists in the dump file. By default, all files are restored to the absolute or relative location as each file is described in the dump file. If the -s option is specified, however, all file names with an absolute path in the dump file are restored relative to the current directory, using the entire path as contained in the dump file.

The samfsdump command does not create a dump of any data associated with the files (unless the -P, -u or -U options are specified), so no data can be restored from this dump file. It is assumed that the data associated with the dumped files has been archived in some way. If a file for which no archive copy is available is dumped, a warning message is issued noting that this file will be marked as damaged when restored. When that file is restored from the dump file, it is marked as damaged by samfsrestore. Note that this warning can be explicitly suppressed by using the -q option.

If dump file contains ACLs, they could be either of POSIX ACLs or NFSv4 ACLs. Each type of ACL would normally be restored to the filesystem supporting that type of ACL. If the dump file contains NFSv4 ACLs and the filesystem supports POSIX ACLs, or the dump file contains POSIX ACLs and the filesystem supports NFSv4 ACLs, no conversion will be performed, a warning will be issued, and files will be restored with empty ACLs.

```
     You must be logged in as superuser (root) in order to
     execute the samfsdump and samfsrestore commands.  Sun
     Microsystems recommends that a site create samfsdump dumps
     on a periodic basis as part of a disaster recovery plan.

OPTIONS
     This command accepts the following options:

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

     -d        Enables debugging messages.  This option is useful
               only to Oracle Corporation and is used to trace
               execution for verification purposes.

     -f dump_file
               Names the file to which the control structure data
               dump is written to (by samfsdump) or read from (by
               samfsrestore).  You must specify a dump_file.

               If a dash character (-) is specified for the
               dump_file, samfsdump writes the dump file to
               stdout and samfsrestore reads the dump file from
               stdin.

               The dump file data can be passed through
               appropriate filters, such as compression or
               encryption, after being written by samfsdump or
               before being read by samfsrestore.

     -g log_file
               (samfsrestore only) Generates a file of online
               directories and files.  For information on the
               format of this file, see the NOTES section of this
               man page.

     -i        (samfsrestore only) Prints the inode numbers of
               the files when listing the contents of the dump.
               For more listing options, see -l, -t, and -2
               options.

     -I include_file
               (samfsdump only) Takes the list of files to dump
               from include_file.  This file has one relative or
               absolute path to be dumped per line.  After
               processing include_file, any [file] arguments from
               the command line are processed.

     -l        (samfsrestore only) Prints one line per file.
               This option is similar to the sls(1M) command's -l
               option when listing the dump contents.  Note that
               this option is identified by the lowercase letter
               'l', not a number '1'.  For more listing options,
               see the -i, -t, and -2 options.
```

```
-n        (Obsolete. samfsdump only.) Always uses the new
          header format.  The new header is incompatible
          with samfsrestore prior to the 3.5.0 release
          level.

-P        (samfsdump only) Dumps the online data portions of
          files which are offline, but have partial data
          online.  This option can considerably increase the
          size of the dump file, as data and metadata are
          both being dumped.  You must take care to manage
          the increased size of the dump.  This option can
          be used to move file partial data by piping the
          output of samfsdump to the input of samfsrestore.

-q        (samfsdump only) Suppresses warning messages for
          damaged files.  By default, samfsdump writes
          warning messages for each file that would be
          considered damaged if the dump were restored.

-r        (samfsrestore only) Replaces existing files when
          restoring control structures if the existing files
          have an older modification time than the dumped
          files.

-s        (samfsrestore only) Removes leading slashes from
          file names prior to restoring them.  This is
          useful if the dump was made with an absolute path
          name and the dump is being restored to a different
          location.  Any directories required for the
          restoration and not defined in the dump file are
          automatically created.

-t        (samfsrestore only) Lists the content of the dump
          file rather than restoring the dump.  For more
          listing options, see the -i, -l, and -2 options.

-u        (samfsdump only) Dumps the data portions of files
          without at least one archive copy.  This option
          can considerably increase the size of the dump
          file, as data and metadata are both being dumped.
          You must take care to manage the increased size of
          the dump.

-U        (samfsdump only) Dumps the data portions of files
          which are online.  This option can considerably
          increase the size of the dump file, as data and
          metadata are both being dumped.  If this option is
          used with segmented files, the archive copy
          information is not preserved when the file is
          restored.  You must take care to manage the
          increased size of the dump.  This option can be
          used to move file systems by piping the output of
          samfsdump to the input of samfsrestore.

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

-B size   Specifies a buffer size in units of 512 bytes.
          Note that there are limits on the buffer size, as
```

```
               specified in the error message when the limits
               have been exceeded.  The default buffer size is
               512 * 512 bytes.

     -H        For samfsdump, creates the dump file without a
               dump header record.  For samfsrestore, declares
               that the existing dump file has no header record.
               This option can be used to create control
               structure dump files that can be concatenated
               using the cat command.  For more information on
               this command, see the cat(1) man page.

     -R        (samfsrestore only) Replaces existing files when
               restoring control structures.

     -S        Perform only a scan to create a db_loadfile with
               the -Z option.  When using -S during samfsdump, no
               dump file is created and -f is not needed.  During
               samfsrestore, -S used with -Z will create a
               db_loadfile from the dump file specified by -f and
               no restore is performed.

     -T        Displays statistics at command termination.  These
               statistics include the number of files and
               directories processed, the number of errors and
               warnings, and other information.  Example:

               samfsdump statistics:
                             Files:              52020
                             Directories:        36031
                             Symbolic links:     0
                             Resource files:     8
                             File segments:      0
                             File archives:      0
                             Damaged files:      0
                             Files with data:    24102
                             File warnings:      0
                             Errors:             0
                             Unprocessed dirs:   0
                             File data bytes:    0

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

               File segments refers to the number of data
               segments associated with segmented files from the
               dump.

               File archives refers to the number of archive
               images associated with the preceding Files,
               Directories, Symbolic links, and Resource files.

               Damaged files refers to the number of Files,
               Directories, Symbolic links, and Resource files
               that are either already marked damaged (for a
               samfsdump) or were damaged during a restore
```

```
          because they had no archive image (for a
          samfsrestore).

          Files with data refers to the number of Files that
          have online (full or partial) data dumped or
          restored.

          File warnings refers to the number of Files,

          Directories, Symbolic links, and Resource files
          that would be damaged should the dump be restored
          because they had no archive images at the time of
          the dump.

          Errors refers to the number of error messages that
          were printed during the dump or restore.  These
          errors indicate a problem, but the problem is not
          severe enough to cause an early exit from
          samfsdump or samfsrestore.  Examples of errors
          during a restore are failing to create a symbolic
          link and failing to change the owner or group of a
          file.  Errors that might occur during a dump
          include having a path name too long, failing to
          open a directory for reading, failing to read a
          symbolic link or resource file, or finding a file
          with an invalid mode.

          Unprocessed dirs refers to the number of
          directories that were not processed due to an
          error, such as being unable to create the
          directory.

          File data bytes refers to the size of data that
          was dumped (using options -P, -U, or -u) or
          restored.

-W        (Obsolete. samfsdump only.)  Writes warning
          messages during the dump process for files that
          would be damaged if the dump were restored.  This
          option is retained for compatibility.  By default,
          these warning messages are now issued
          automatically.  For more information on
          controlling this behavior, see the -q option,
          which suppresses warning messages.

-X excluded_dir
          (samfsdump only) Specifies directory paths to be
          excluded from the dump.  Relative paths without
          leading characters must be used, for example
          dir1/dir2.  The result is an empty directory
          dir1/dir2 in the dump file.  A directory that
          resolves to . or NULL generates an error message.
          Multiple (up to 10) directories can be excluded by
          using multiple -X options.

-Y        (samfsdump only) Specifies that the trailing list
          of files are lists of files to dump.  Using this
          option helps improve samfsdump performance by
          reducing the number of path lookups.  If - is
```

```
               specified as the trailing list, standard input is
               used.

               Each list must have one line per file, with tab
               separated inode number, generation number, and
               file path.  The path must is relative to where
               samfsdump is executed.

               Example line: 1039 11 testdir2/rtest_f_61

               Example usage: samfsdump -Y -f samfs1.dump
               /path/to/filelist

               Example pipelined: samdb dump samfs1 | samfsdump
               -Y -f samfs1.dump -

               If a sideband mysql database is being used by the
               target SAM filesystem, then the file list can be
               generated using the samdb(1M) dump command.

     -Z db_loadfile
               Specifies that a samdb(1M) db_loadfile should be
               created as part of a samfsdump or samfsrestore.
               This file is used to populate a sideband mysql
               database using the samdb(1M) load command.

               Use the -S option to only produce the db_loadfile
               without performing the usual samfsdump or
               samfsrestore operations.  If - is specified for
               the load file standard output is used.

     -2        (samfsrestore only) Writes two lines per file,
               similar to the sls(1) command's -2 option, when
               listing the contents of the dump.  For more
               listing options, see the -i, -l, and -t options.

     file ...  Lists files to be dumped or restored.  Note that
               the names given to restore must match exactly the
               names as they are stored in the dump.  You can use
               samfsrestore -t to see how the names are stored.
```

NOTES

```
     A samfsrestore should not be attempted on a Sun QFS shared
     file system client.

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

     If the -g option is used, a log file is generated during
     file system restoration.  This file contains one line per
     file that was online, or partially online, at the time the
     file was dumped.  This line is divided into fields and
     contains the following information:

     Field  Description

     1      The file type, which is indicated by one of the
            following letters:
             o d indicates a directory.
             o f indicates a regular file.
             o l indiactes a symbolic link.
             o R indicates a removable media file.
             o I indicates a segment index.
             o S indicates a data segment.

     2       The media type and Volume Serial Name (VSN) in
             media_type.vsn format.

     3       The position on the media.

     4       Either online or partial.

     5       The path relative to the file system mount point.

     After a samfsrestore command is issued, it is possible to
     restore files that were online, prior to the dump, back to
     their online state.  You do this by using the script in
     /opt/SUNWsamfs/examples/restore.sh.

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

     example# cd /sam
     example# samfsdump -f /destination/of/the/dump/samfsdump.today

     To restore a control structure dump to /sam:

     example# cd /sam
     example# samfsrestore -f /source/of/the/dump/samfsdump.yesterday

     To create a new samdb(1M) database load file of /sam:

     example# cd /sam
     example# samfsdump -SZ /destination/samfsdbload.today

     To create a dump of /sam using a list of files:

     example# cd /sam
     example# samfsdump -Y -f /destination/of/samfsdump.today /source/of/samfslist.today

     To create a new samdb(1M) load file from an existing dump file:

     example# samfsrestore -SZ /destination/samfsdbload.today -f /source/samfsdump.yesterday

SEE ALSO
     cat(1), sls(1), samdb(1M).

DIAGNOSTICS
     You may encounter messages while using the samfsdump or
     samfsrestore command.  The following list shows several
     possible messages and their explanations:

     Message             Explanation

     file: Unrecognised mode (0x..)
                         samfsdump is being asked to dump a file
```

```
                    that is not a regular file, directory,
                    symbolic link, or removable media file.
                    The Sun QFS and SAM-QFS file systems
                    allow the creation of block special,
                    character special, fifo, and other
                    special files, but they do not function
                    correctly.  samfsdump does not attempt
                    to dump them.

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

     file: Warning! File is already damaged.
                    If received during a samfsdump, means
                    that the file is currently marked
                    damaged.  During restoration, the file
                    will still be damaged.

     file: File was already damaged prior to dump
                    If received during a samfsrestore, this
                    means that the file was dumped with the
                    damaged flag set.

     file: File is now damaged
                    If received during a samfsrestore, this
                    means that the file was dumped when it
                    had no archive images.  samfsdump and
                    samfsrestore do not dump file data.
                    They rely on the file’s data having been
                    archived.  Because the file no longer
                    has any data associated with it, it is
                    marked damaged.

     .: Not a SAM-FS file.
                    You are attempting to dump files from a
                    file system that is not a Sun QFS or
                    SAM-QFS file system, or you are
                    attempting to restore files from a
                    samfsdump dump file into a file system
                    that is not a Sun QFS or SAM-QFS file
                    system.

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

```
Corrupt samfsdump file.  name length %d
                    If received during a restore, this means
                    that the path name of a file to be
                    restored was less than zero or larger
                    than MAXPATHLEN.  This should not occur.
                    samfsrestore aborts.

Corrupt samfsdump file. %s inode version incorrect
                    During a restore, this means that a the
                    inode for the indicated file was in an
                    old format.  This should not occur.
                    samfsrestore aborts.

file: pathname too long
                    If received during a dump, this
                    indicates that the path name of the
                    indicated file is longer than 1024
                    characters.  The file is not dumped.
```

samfstyp(1M)

```
NAME
     samfstyp - Determines Sun QFS or SAM-QFS file system type

SYNOPSIS
     /opt/SUNWsamfs/sbin/samfstyp [-v] device

AVAILABILITY
     SUNWqfs
     SUNWsamfs

DESCRIPTION
     The samfstyp utility displays the Sun QFS or SAM-QFS file
     system type of the file system identified by device.
     Optionally, samfstyp displays detailed information about
     that file system.

     You must be the Superuser to use this utility.  If the file
     system is not a Sun QFS or SAM-QFS file system, or if you
     are not the Superuser, no output is generated.

     The first line of samfstyp output identifies the file system
     type of the specified device.  Available file system types
     are:

     sam-fs-sbv1  Sun QFS file system with superblock version 1
     sam-fs       Sun QFS file system with current superblock
     sam-qfs-sbv1 SAM-QFS archiving file system with superblock version 1
     sam-qfs      SAM-QFS archiving file system with current superblock

     The samfstyp utility displays detailed information about the
     identified Sun QFS or SAM-QFS file system.  Information may
     be displayed for some, or all, of the following items,
     subject to file system configuration:
          Superblock (General)
          Family Set Members
          I-node Information
          Volume Table of Contents
          Host Table
          Controller
          Disk Geometry
```

OPTIONS

This command accepts the following options:

-v	Generates detailed information about the Sun QFS or SAM-QFS file system identified by device.
device	Identifies the device from which the file system is analyzed.

EXAMPLES

Example 1:

```
fireball# cat /etc/opt/SUNWsamfs/mcf
qfs1                  10      ma      qfs1    on
/dev/dsk/c6t0d0s3     11      mm      qfs1    on
/dev/dsk/c6t0d0s4     15      mr      qfs1    on
fireball# samfstyp /dev/rdsk/c6t0d0s4
qfs
fireball#
```

Example 2:

```
fireball# samfstyp -v /dev/rdsk/c6t0d0s4
sam-qfs
/dev/rdsk/c6t0d0s4 {
  name                = SBLK
  magic               = 0x76657232
  gen                 = 0
  id                  = 0x3f3426798333ada1
  init                = Fri Aug  8 17:38:49
  update              = Fri Aug  8 17:38:49
  state               = clean
  sb1_offset          = 0
  sb2_offset          = 0
  host_offset         = 0
  inode_offset        = 0
  user_min_inode      = 1025
  ext_shift           = 12
  sm_meta_blocks      = 4
  lg_meta_blocks      = 16
  sm_data_blocks      = 64
  lg_data_blocks      = 64
  eq_id               = 10
  fset_name           = qfs1
  fset_ord            = 1
  fset_blks_free      = 0
  fset_blks           = 0
  fset_meta_count     = 1
  fset_data_count     = 1
  fset_count          = 2
  fset 0 {
    ord                     = 0
    eq_id                   = 11
    dev_type                = mm
    slice_state             = clean
    meta_ord                = 0
    stripe_count            = 1
    part_blocks_free        = 2098496
    part_blocks             = 2098928
    alloc_map_offset        = 18
    alloc_map_blocks        = 73014444050
    lg_dau_next             = 0
    lg_dau_count            = 131183
    sys_blocks              = 274878038127
  }
  fset 1 {
    ord                     = 1
    eq_id                   = 15
    dev_type                = mr
    slice_state             = clean
    meta_ord                = 0
    stripe_count            = 1
    part_blocks_free        = 2098816
    part_blocks             = 2098880
    alloc_map_offset        = 35
    alloc_map_blocks        = 21474836515
    lg_dau_next             = 0
    lg_dau_count            = 32795
    sys_blocks              = 274877939739
  }
  vtoc {
    label                   = SUN9.0G cyl 4924 alt
    boot                    = 0x0/0x0/0x0
    sanity                  = 0x600ddeee
    layout                  = 1
    name                    = ''
    sector_size             = 512
    part_count              = 8
    part 0 {
      id                      = unassigned
      permissions             = (none)
      first_sector            = 0
      blocks                  = 132867
    }
    part 1 {
      id                      = unassigned
      permissions             = (none)
      first_sector            = 132867
      blocks                  = 4197879
    }
    part 2 {
      id                      = backup
      permissions             = (none)
      first_sector            = 0
      blocks                  = 17682084
    }
    part 3 {
      id                      = unassigned
      permissions             = (none)
      first_sector            = 4330746
```

```
            blocks               = 4197879
          }
          part 4 {
            id                   = unassigned
            permissions          = (none)
            first_sector         = 8528625
            blocks               = 4197879
          }
          part 5 {
            id                   = unassigned
            permissions          = (none)
            first_sector         = 12726504
            blocks               = 4197879
          }
          part 6 {
            id                   = unassigned
            permissions          = (none)
            first_sector         = 16924383
            blocks               = 757701
          }
          part 7 {
            id                   = unassigned
            permissions          = unmountable
            first_sector         = 0
            blocks               = 0
          }
        }
        controller {
          name                   = pci1000,f
          type                   = scsi-ccs
          flags                  = 0x8
          number                 = 3
          address                = 0x0
          bus                    = 0x0
          intr_pri               = 0
          intr_vec               = 0x0
          drive_name             = sd
          unit_num               = 45
          slave_num              = 0
          part_num               = 4
          max_trans              = 2048
        }
        geometry {
          data_cyl               = 4924
          alt_cyl                = 2
          cyl_offset             = 0
          heads                  = 27
          track_sect             = 133
          interleave             = 1
          cyl_alt                = 0
          rpm                    = 7200
          phys_cyl               = 4926
          sect_read_skip         = 0
          sect_write_skip        = 63
        }
      }
      fireball#
```

SEE ALSO

fstyp(1M)

samgetmap(1M)

```
NAME
     samgetmap - Obtains disk file storage information

SYNOPSIS
     /opt/SUNWsamfs/tools/samgetmap [-a allocsize] [-c] [-f] [-h]
     [-l setlen] [-m minalloc] [-n nbytes] [-s startaddr] [-u]
     [-w] [-M] [-U] [-V] file

AVAILABILITY
     Oracle Corporation Internal

DESCRIPTION
     The samgetmap command provides a test interface to the
     SANergy File Map API routines.  The samgetmap command must
     be run as root.

OPTIONS
     This command accepts the following options:

     -a allocsize
               Requests allocation of allocsize bytes of storage
               to file through the FS_SetFileSizes request.  The
               allocsize must be an integer in the following
               range:

               0 < allocsize < 2**31

               For more information, see the OPERATIONS section
               of this man page.

     -c        Requests a canonical map.  FS_M_FLAG_CANONICAL is
               set for the FS_GetLockedMap request.  For more
               information, see the OPERATIONS section of this
               man page.

     -f        Requests a non-sparse allocation map.
               FS_M_FLAG_NO_HOLE is set for the FS_GetLockedMap
               request.  For more information, see the OPERATIONS
               section of this man page.

     -h        Causes samgetmap to write a short usage message to
               stdout.

     -l setlen Sets the file length to setlen bytes.

     -m minalloc
               Uses FS_SetFileSizes to request the allocation of
               minalloc bytes of storage for file.  minalloc must
               be an integer in the following range:

               -1 < minalloc < allocsize

               If minalloc is within this range and the
               allocation request partially succeeds, such that
               minalloc or more bytes are allocated but fewer
               than allocsize bytes are allocated, a unique error
```

```
                    code is returned to indicate partial success.  For
                    more information, see the OPERATIONS section of
                    this man page.

                    samgetmap's behavior is undefined if minalloc is
                    less than -1 or greater than allocsize.  For more
                    information on allocsize, see the -a option.

          -n nbytes Specifies that the allocation map returned by the
                    FS_GetLockedMap call return information about the
                    location of at least nbytes of data.  For more
                    information, see the OPERATIONS section of this
                    man page.

          -s startaddr
                    Requests that the allocation map returned by the
                    FS_GetLockedMap request return the information
                    about file's storage beginning at byte startaddr
                    in the file.  For more information, see the
                    OPERATIONS section of this man page.

          -u        Sets FS_M_FLAG_UNLOCKED in the call to
                    FS_GetLockedMap.  file is not locked by the
                    FS_GetLockedMap call.  For more information, see
                    the OPERATIONS section of this man page.

          -w        Sets FS_M_FLAG_WAIT for the call to
                    FS_GetLockedMap.  If the file is online, this
                    option has no effect.  If the file is offline, the
                    file is staged in before the file's map is
                    returned.  For more information, see the
                    OPERATIONS section of this man page.

          -M        Prevents the call to FS_GetLockedMap from being
                    made.  For more information, see the OPERATIONS
                    section of this man page.

          -U        Prevents the call to FS_UnlockMap from being made,
                    leaving the file locked (unless the -u option is
                    also specified).  For more information, see the
                    OPERATIONS section of this man page.

                    NOTE:  Use of this option makes it impossible to
                    unmount the file system that file resides in until
                    the file is somehow unlocked (see also
                    samunhold(1M) to remove all file locks (holds) on
                    a file system).  Specifying the -u and -U options
                    together on the command line insures that the file
                    is unlocked upon exit from samgetmap.

          -V        Report the value returned by the FS_GetVersion(),
                    added to the SANergy 2.2 API, which indicates the
                    library revision value.  Present valid values are
                    220 (for SANergy 2.2), and 230 (for SANergy
                    2.3/3.1).

          file      Specifies the file for which disk storage
                    information is requested.
```

The samgetmap command should not be executed if SANergy File Sharing is running on the file system. Specifically, the administrator should ensure that the following conditions are true:

o There are no SANergy applications running on any client, possibly including the server itself.

o The file system in question is not fused on any SANergy clients.

OPERATIONS

The samgetmap program operates as follows:

1. The program calls the AFS_GetCookies routine to obtain two cookies, one for the file system that file resides on (a volume cookie) and the other for file itself (a file cookie).

2. The program calls FS_GetLockedMap, using the cookies obtained by FS_GetCookies. This call is not made if the -M option is present. The parameters to this call can be specified or modified by the command line options -s, -n, -c, -f, -u, and -w.

3. The program calls FS_SetFileSizes. The parameters to this call can be specified or modified by the command line options -a, -l, and -m.

4. The program calls FS_UnlockMap. This call is not made if the -U option is present on the command line.

The result of each call is written in a message to stdout.

EXAMPLES

The following example shows how to use samgetmap:

```
ceres# samgetmap /qfs1/foo
AFS_GetCookies("/qfs1/foo", &vc, &fc) = 0 (FS_E_SUCCESS (OK))

Volcookie:

          0  0  0 20 73 56 6d 43          0  b  0  0 3a 94 44 4d
          0  1  5  6  0  0  0  0          0  0  3  0  2 ed a0  0

Filecookie:
          0  0  0 10 73 46 6d 43          0  0  0  7  0  0  0  1
          0  0  0  0  0  0  0  0          0  0  0  0  0  0  0  0

FS_GetLockedMap(&vc, &fc, 0, -1, 0, ffbaf448, ffbaf340)  = 0 (FS_E_SUCCESS
(OK))
returned buflen = 88
msgLen       = 88
vendorStatus = 0
fileSize     = 16384000
allocation   = 16416768
nExtents     = 1
extentType   = 2        (SIMPLE)
```

```
     sExtent[0]
             volumeOrdinal = 1
             blockOffset   = 20
             nBlocks = 7d40

     FS_SetFileSizes(&vc, &fc, 0, ffffffffffffffff, ffffffffffffffff, ffffffffff
     ffffff)  = 0 (FS_E_SUCCESS (OK))
     FS_UnlockMap(&vc, &fc) = 0 (FS_E_SUCCESS (OK))
```

NOTES
     The samgetmap command typically issues holds (locks) to
     file. This can interfere with the operation of SANergy File
     Sharing, possibly causing file system corruption.

SEE ALSO
     samgetvol(1M), samunhold(1M).

samgetvol(1M)

NAME
     samgetvol - Obtains disk file system storage information

SYNOPSIS
     samgetvol [-h] [-r] [-w] mntpoint

AVAILABILITY
     Oracle Corporation Internal

DESCRIPTION
     The samgetvol command provides a test interface to the
     SANergy Volume Map API routine. The samgetvol command must
     be run as root.

OPTIONS
     This command accepts the following options:

     -h   Causes samgetvol to print a short usage message
               and exit.

     -r   Causes samgetvol to query the FS_GetMaxLeases API
               for the read lease period. The result is written
               to the output.

     -w   Causes samgetvol to query the FS_GetMaxLeases API
               for the write lease period. The result is written
               to the output.

     mntpoint A file system mount point. For example, /qfs1.
               The samgetvol program operates by calling the
               routine AFS_GetVol to obtain information about the
               file system mounted on mntpoint. The result of
               this operation is written to stdout.

EXAMPLES
     The following example shows output from the samgetvol
     command:

```
ceres# samgetvol -rw /qfs1
msgLen       =      504 [0x1f8]
vendorStatus =        0 [0]
VolCookie =
                  0  0  0 20 73 56 6d 43     0  b  0  0 3a 94 59 aa
                  0  1  4  5  0  0  0  0     0  0  3  0  2 ed a0  0
fsType       =        3 (SPARC SAM-FS)
system       = endian= 1/cpu= 2///os= 3/fs= 2//
glomType     =        3 (SAM_RAID0)
glomInfo     =    16384 [0x4000]
nDisks       =        5 [0x5]
blockSize    =      512 [0x200]

Disk[0]

        idOffset    =    268699648 [0x10040800]
        blockOffset =       524800 [0x80200]
        idLength    =           16 [0x10]
        nBlocks     =     10486400 [0xa00280]
        flags       = 0x2               ( META )
        diskID:
                 53 42 4c 4b fd 18 7e 20
                 3a 94 59 aa  0  0  0  0

Disk[1]
        idOffset    =    268699648 [0x10040800]
        blockOffset =       524800 [0x80200]
        idLength    =           16 [0x10]
        nBlocks     =     10486400 [0xa00280]
        flags       = 0
        diskID:
                 53 42 4c 4b fd 18 7e 20
                 3a 94 59 aa  0  0  0  1

Disk[2]
        idOffset    =   5637736448 [0x150090800]
        blockOffset =     11011200 [0xa80480]
        idLength    =           16 [0x10]
        nBlocks     =     10486400 [0xa00280]
        flags       = 0x1               ( STRIPE_SUB )
        diskID:
                 53 42 4c 4b fd 18 7e 20
                 3a 94 59 aa  0  0  0  2

Disk[3]
        idOffset    =         2048 [0x800]
        blockOffset =            0 [0]
        idLength    =           16 [0x10]
        nBlocks     =       494109 [0x78a1d]
        flags       = 0
        diskID:
                 53 42 4c 4b fd 18 7e 20
                 3a 94 59 aa  0  0  0  3

Disk[4]
        idOffset    =    253340672 [0xf19ac00]
        blockOffset =       494802 [0x78cd2]
        idLength    =           16 [0x10]
```

```
                nBlocks     =       494109 [0x78a1d]
                flags       = 0x1               ( STRIPE_SUB )
                diskID:
                         53 42 4c 4b fd 18 7e 20
                         3a 94 59 aa  0  0  0  4

           read lease = 30s  write lease = 30s
```

SEE ALSO
     samgetmap(1M), samunhold(1M).

samgrowfs(1M)

NAME
     samgrowfs - Adds disk partitions to an existing Sun QFS or SAM-QFS file system

SYNOPSIS
```
     samgrowfs [-V] fsname
```

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The samgrowfs command adds disk partitions to an Sun QFS and SAM-QFS file system and allows the file system to grow.

     The following procedure uses the samgrowfs command to increase the size of a Sun QFS or SAM-QFS file system:

1. Unmount all the file systems you want to grow.

2. In a Sun QFS or SAM-QFS environment, idle all drives by entering a samcmd idle eq and a samd stop command. For more information on these commands, see the samcmd(1M) and samd(1M) man pages.

3. Edit the mcf file, save the changes, and quit the editor. Up to 252 disk partitions can be specified in the mcf file for a Sun QFS or SAM-QFS file system. The new partitions must be placed after the existing partitions for the specified family set fsname.

4. Run the samd config command to notify sam-fsd of the change to /etc/opt/SUNWsamfs/mcf (see samd(1M)).

5. Run the samgrowfs(1M) command on the fsname file system.

6. Mount the fsname file system.

     For more information on this procedure, see the Sun QFS File System Configuration and Administration Guide.

OPTIONS
     This command accepts the following arguments:

```
     -V        Lists configuration information but does not  exe-
               cute the command.

     fsname    Specifies the existing family set name of the file
               system  that  is  to grow.  This is the family set
               name as specified in the mcf file.
```

EXAMPLE

The following example adds 2 partitions to an existing 1-partition Sun QFS file system.   The mcf file for the existing 1-partition file system with a family set name of samfs1 is as follows:

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

The procedure is as follows:

1. Unmount the samfs1 file system.

   ```
   server# umount samfs1
   ```

2. Kill the sam-amld process:

   ```
   server# samd stop
   ```

3. Edit the mcf file and add the 2 new partitions for the file system with family set name of samfs1:

   ```
   samfs1  10  ms  samfs1
   /dev/dsk/c0t3d0s7  11  md  samfs1  -
   /dev/dsk/c2t3d0s2  12  md  samfs1  -
   /dev/dsk/c2t4d0s2  13  md  samfs1  -
   ```

4. Use the samd(1M) config command to propagate the file changes and restart the system:

   ```
   server# samd config
   ```

5. Grow and mount the file system by entering the following commands:

   ```
   server# samgrowfs samfs1
   server# mount samfs1
   ```

FILES

```
     /etc/opt/SUNWsamfs/mcf   The configuration file for Sun  QFS
                              and SAM-QFS file systems.
```

SEE ALSO

samcmd(1M), samd(1M), sammkfs(1M).

mcf(4).

Sun QFS File System Configuration and Administration Guide.

WARNINGS

As with creating any type of file system, if you specify the wrong partition names, you risk damaging user or system data. Be sure to specify partitions which are otherwise unused on your system. Do not use overlapping partitions.

To grow a Sun QFS file system, you must add a metadata partition (mm) prior to issuing a samgrowfs command. Data partitions can be added as well as metadata partitions. The added metadata partition contains block reservation information for all added partitions. When adding a small metadata partition with large data partitions, the small metadata partition may be too small to hold the block reservation as well as other information, depending on total storage added and DAU size. This condition may cause an error, or a very full metadata partition after samgrowfs.

If the file system is not unmounted prior to changing the configuration, you can end up in a situation where you can not mount the file system. In that case, take the eq out of mcf and run samd config.

samimport(1M)

NAME

import - Imports cartridges into a library or the historian

SYNOPSIS

```
/opt/SUNWsamfs/sbin/import [[-v volser] | [-c num -s pool]]

[-e] [-l] [-n] eq

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

AVAILABILITY

SUNWsamfs

DESCRIPTION

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

```
import 27
```

The second form of the import command can be used only when eq is the Equipment Identifier of the default historian(7) and the cartridge is neither two-sided nor partitioned. This form adds an entry to the historian's catalog for the given type and the given barcode or volser. At least one of the -b barcode or -v volser identifiers must be present. For example:

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

OPTIONS

This command accepts several options. Some of the options affect only certain automated libraries. See the option

```
descriptions and the NOTES section for information pertinent
to vendor-specific automated libraries.  The options for the
import command are as follows:

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

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

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

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

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

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

-l        (Network-attached StorageTek automated libraries
          only.)

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

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

-n        Specifies that the media is unlabeled foreign tape
          (not SAM-QFS media). It is write protected and can
          be only used for read access.

-v volser (Network-attached ADIC/GRAU, StorageTek, and IBM
          3494 automated libraries only.  For the IBM 3494
          library, this option is accepted only when running
          in shared mode; for more information, see the
          ibm3494(7) man page.)

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

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

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

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

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

FILES
     mcf       The configuration file for SAM-QFS environments.

SEE ALSO
     export(1M), sam-robotsd(1M).

     mcf(4).

     historian(7), ibm3494(7).
```

samload(1M)

```
NAME
     samload, load - Loads media into a device

SYNOPSIS
     /opt/SUNWsamfs/sbin/samload [ -w ] eq:slot[:partition] [ deq
     ]

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

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

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

AVAILABILITY
     SUNWsamfs

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

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

     The load and samload commands are identical; samload is pro-
     vided  as an alternative to avoid conflict with the Tcl com-
     mand of the same name.

OPTIONS
     -w        load will  wait  for  the  operation  to  complete
               before terminating.

FILES
     mcf                 The  configuration  file   for   SAM-QFS
                         environments

SEE ALSO
     unload(1M), set_state(1M), mcf(4), sam-robotsd(1M)
```

sammkfs(1M)

```
NAME
     sammkfs, samfsinfo - Constructs or displays information for
     a Sun QFS or SAM-QFS file system

SYNOPSIS
     /opt/SUNWsamfs/sbin/sammkfs [-a allocation_unit] [-i inodes]
     [-A] [-P] [-S] [-V] fs_name

     /opt/SUNWsamfs/sbin/samfsinfo fs_name

AVAILABILITY
     SUNWqfs

     SUNWsamfs

DESCRIPTION
     The sammkfs command creates a Sun QFS or SAM-QFS file system
     from the disk partitions that belong to the family set
     fs_name, where fs_name is the family set name as defined in
     the mcf file.  Up to 252 disk partitions can be specified in
     the mcf file for a Sun QFS or SAM-QFS file system.  The
     sammkfs command can also be used to recreate a file system
     after a disaster.

     The sammkfs command can create either a version 2 file
     system that is backwards compatible with previous releases,
     or a version 2A file system that has new features, but is
     not compatible with previous releases.  By default, a
     version 2A file system is created.  See -P parameter below
     for details on the new features, and how to create a version
     2 file system.

     The sammkfs command aligns the block allocation bit maps and
     round robins them on the metadata devices for improved
     performance.  This behavior is  backwards compatible with
     previous releases.  The option feature Aligned Maps is set.

     The samfsinfo command displays the structure of an existing
     Sun QFS or SAM-QFS file system.  The output is similar to
     that obtained by using the -V option to the sammkfs command.

OPTIONS
     These commands accept the following options:

     -a allocation_unit
               Specifies the disk allocation unit (DAU).  The DAU
               is the basic unit of online storage.  When you
               specify a DAU size, you specify the number of
               1024-byte (1 kilobyte) blocks to be allocated for
               a file.

               The DAU size you can specify depends on the type
               of file system being initialized, as follows:

               o  The SAM-QFS file system is an ms file system.
                  The disk devices in it are all md devices.
                  Both data and metadata are written to the md
                  devices.  The allocation_unit specifies the DAU
                  to be used for the md devices.  Possible
                  allocation_unit specifications are 16, 32, or
                  64 (the default).

               o  The Sun QFS or SAM-QFS file systems are ma file
                  systems.  The metadata in these file systems is
                  written to mm devices.  The disk devices in
                  these file systems are specified as either md,
                  mr, or gXXX devices, as follows:

                  -  For the md devices, possible allocation_unit
                     specifications are 16, 32, or 64 (the
                     default).  A single file system cannot have
                     md devices mixed among the mr and gXXX
                     devices.

                  -  For mr devices, the DAU is fully adjustable.
                     Specify an allocation_unit that is a
                     multiple of 8 in the following range for mr
                     devices:  8 < allocation_unit < 65528.  The
                     default is 64.

                  -  For gXXX devices, which specify striped
                     groups, the DAU is fully adjustable.  If the
                     file system contains striped groups, the
                     minimum unit of disk space allocated is the
                     DAU multiplied by the number of members in
                     the striped group.  Specify an
                     allocation_unit that is a multiple of 8 in
                     the following range for gXXX devices:
                     8 < allocation_unit < 65528.  The default is
```

```
                 256.

              You can mix mr and gXXX devices in a single Sun
              QFS or SAM-QFS file system.  If these device
              types are mixed, the allocation_unit specified
              is used for both device types.  If no
              allocation_unit is specified, the DAU size used
              for each type of device is 256.

     -i inodes Specifies the number of inodes to be allocated for
               this file system.  This is the total number of
               user inodes that can be used for the life of this
               file system. In Sun QFS and SAM-QFS version 2
               superblock file systems, a number of inodes are
               reserved for file system usage, and are
               unavailable to the user. This number is in
               addition to the specified number of user inodes.
               The actual number of inodes available vary from
               that specified, due to rounding to metadata DAU
               size.

               NOTE:  By specifying this option, you eliminate
               the possibility of ever increasing the number of
               inodes for the file system.  Therefore, Sun does
               not recommend the use of this option.

               When this option is specified, later use of the
               samgrowfs(1M) command increases the size of the
               file system, but it cannot increase the number of
               allowable inodes.  For more information on
               enlarging file systems, see the WARNINGS section
               of this man page and the samgrowfs(1M) man page.

     -A        Uses NFSv4 ACL style for the filesystem ACLs
               instead of POSIX ACL style. This feature is
               available only in releases of Solaris beyond
               Solaris 10.

     -P        Specifies that a previous version of the file
               system be created.  This version creates a version
               2 superblock and is compatible with SAM-QFS
               version 4.6.  This version cannot use the
               following features however:  large host table,
               extended attributes, and online grow.  Without the
               -P parameter, a version 2A superblock is created,
               the above features are available, and the file
               system is not usable with SAM-QFS version 4.6 or
               previous.

     -S        Indicates that this file system is shared.  In
               order to mount the file system as a Sun QFS shared
               file system, you must also create a hosts.fs_name
               configuration file.  For more information on this
               configuration file and other aspects of the Sun
               QFS shared file system, see the Sun QFS File
               System Configuration and Administration Guide.
               For information on configuring a hosts file, see
               the hosts.fs(4) man page.
     -V        Writes configuration information to standard
               output but does not execute the sammkfs command.
               This information can be used to create a new file
               system.

               The samfsinfo command should be used to generate
               configuration information for an existing file
               system.

EXAMPLES
     Example 1.  The following command creates SAM-QFS file
     system with a DAU size of 128 kilobytes:

     server#  sammkfs -a 128 samfs1

FILES
     /etc/opt/SUNWsamfs/mcf   The configuration file for a Sun
                              QFS or SAM-QFS file system

WARNINGS
     As with creating any type of file system, if you specify the
     wrong partition names, you risk damaging user or system
     data.  Be sure to specify partitions that are otherwise
     unused on your system.  Do not use overlapping partitions.

     With SAM-QFS 4.1 and greater AND Solaris 64bit kernels which
     support large disk devices (greater than 1 TB), it is
     possible to have partitions that are greater than 1 TB. Note
     that these file systems are not usable on Solaris systems
     that do not support large disk devices.

SEE ALSO
     dd(1M), samd(1M), samgrowfs(1M), undamage(1M).
     mcf(4).
     Sun QFS File System Configuration and Administration Guide.
     Sun Storage Archive Manager Configuration and Administration
     Guide.

WARNINGS
     Be careful when using the -i inodes option for this command.
     By using this option, you dictate the maximum number of
     inodes allowed for the life of this file system.  This
     eliminates the possibility of ever using the samgrowfs(1M)
     command to increase the number of files in this file system.
     After a file system is made with -i specified, the
     samgrowfs(1M) command can only be used to increase the size
     of the file system in terms of bytes.

NOTES
     Data alignment refers to matching the allocation unit of the
     RAID controller with the allocation_unit of the file system.
     A mismatched alignment causes a read-modify-write operation
     for I/O that is less than the block size.  The optimal
     alignment formula is as follows:
     allocation_unit = RAID_stripe_width * number_of_data_disks

     For example, if a RAID-5 unit has a total of 8 disks with 1
     of the 8 being the parity disk, the number of data disks is
     7.  If the RAID stripe width is 64 kilobytes, then the
     optimal allocation_unit is 64 * 7 = 448.
```

samncheck(1M)

NAME
     samncheck - Generates pathnames versus i-numbers for Sun QFS and SAM-QFS file systems

SYNOPSIS

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

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     samncheck generates a pathname in the Sun QFS or SAM-QFS file system 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)

samquota(1M)

NAME
     samquota - Reports, sets, or resets quota information

SYNOPSIS

```
samquota [-a | -A adminsetID] [-e] [-g | -G groupID] [-h]
[-k] [-u | -U userID] [file]

samquota [-b count:type[:scope]] [-f count:type[:scope]]
[-h] [-i] [-k] [-p] [-t interval:scope] [-w]
[-x action:scope] [-A adminsetID] [-G groupID] [-O]
[-U userID] [file]
```

AVAILABILITY
     SUNWsamfs

SUNWqfs

DESCRIPTION

The samquota command displays quota usage statistics and can be used to edit quotas, grace periods, and usages for users, groups, and admin sets. This command supports file counts and online block counts. Note that some options are mutually exclusive.

Only a superuser can use this command to change quotas. End users can use a subset of this command's options to display quota usage and to display limit information. For more information on the end-user version of this command, see the squota(1) man page.

By default, samquota(1M) writes the user's applicable GID/UID quotas and usages on all mounted Sun QFS and SAM-QFS file systems to stdout.

ADMIN SETS AND DIRECTORY/PROJECT QUOTAS

An admin set quota applies to all files and directories on a file system that have their admin set attribute set to the given value. The main use of admin set quotas is to effect directory or project quotas. They can be used to effect directory quotas by setting a directory's admin set ID to a unique value and using samquota(1M) to set quotas for that value. All subdirectories and files subsequently created beneath the directory then inherit the value, and the admin set's quota limits apply to them. Conversely, a project quota can be effected by choosing a set of project directories, setting their admin set ID values to a single unique value, and using samquota(1M) to set quotas for that ID. Note in either case that newly created files inherit an admin set ID from the directory in which they are created; the admin set IDs do not change if the file is moved to a new directory with a different admin set ID.

You can use the samchaid(1M) command to set admin set IDs. The samchaid(1M) command allows system administrators to assign files and directories to individual admin sets. Admin set IDs are not tied to any set of permissions associated with the user. That is, a user can have a set of directories and files on one Sun QFS or SAM-QFS file system with a particular admin set ID, and the same user can have another set of directories and files on another file system (or even the same one) with a completely different admin set ID. A writable file is therefore used as a surrogate to determine that a user has permission to view an admin set's quota values.

OPTIONS

This command accepts the following options:

-a Specifies admin set quota statistics for file. This option is not allowed in combination with the -A option or any of the setting options.

-b count:type[:scope]
Sets soft, hard, or in-use block allocation

limits. This setting can pertain to either online files or to the total number of files. Note that a colon (:) is used to separate each component.

count specifies the number of blocks for the limit and must be an integer number in the following range:
0 < count < (2**63) -1.

By default, the count specification indicates a number of 512-byte blocks. If the -k option is also specified, the count specification is interpreted as a number of 1024-byte blocks.

By default, the integer specified for count is interpreted as it is written. You can append a unit multiplier to the count value, however, to force the system to interpret count as a larger number. These unit multipliers are as follows:

Multiplier	Interpretation
k or K	Specifies 1000. For example, specifying 2k is interpreted as 2000.
m or M	Specifies 1,000,000. For example, specifying 80M is interpreted as 80,000,000.
g or G	Specifies 1,000,000,000.
t or T	Specifies 10**12.
p or P	Specifies 10**15.

type specifies the type of limit. Possible type specifications are as follows:

type	Interpretation
s or soft	Specifies that the samquota command is being used to reset a soft limit.
h or hard	Specifies that the samquota command is being used to reset a hard limit.
u or inuse	Specifies that the samquota command is being used to reset the in-use counter. Typically, this is set only by the samfsck(1M) command and other system administration tools.

scope specifies the scope of the limit. Possible scope specifications are as follows:

scope Interpretation

```
                    o or online    Specifies that the samquota command
                                   is being used to reset an online
                                   limit.  For Sun QFS and SAM-QFS
                                   file systems, files that are
                                   released (offline) are not counted
                                   in the online block usage.

                    t or total     Specifies that the samquota command
                                   is being used to reset a total
                                   limit.  For Sun QFS and SAM-QFS
                                   file systems, both online and
                                   offline files are used to compute
                                   the total block usage.

                                   If no scope is specified both the
                                   online and total limits are set.

                    Example.  The following command line sets a soft
                    limit of 120,000 512-byte blocks to be occupied by
                    user george's files in file system qfs22:

                    samquota -b 120k:s -U george /qfs22

          -e        Writes the quota information from this command
                    line in an executable format.  You can use this
                    option if you want the system to put the
                    information from this command into a file for
                    editing.

                    server# samquota -eG sam /qfs1
                    # Type  ID
                    #                   Limits
                    #             soft            hard
                    # Files
                    # Blocks
                    # Grace Periods
                    #
                    samquota -G 101 \
                          -f     1000:s -f     1200:h \
                          -b   100000:s -b   120000:h \
                                      -t  1d   /qfs1

          -f count:type[:scope]
                    Sets soft, hard, or in-use file limits for a file
                    system.  Note that a colon (:) is used to separate
                    each component.

                    count specifies the number of files for the limit
                    and must be an integer number in the following
                    range:
                    0 < count < (2**63) -1.

                    If the -k option is also specified, any count
                    specification referring to blocks is interpreted
                    in 1024-byte blocks instead of 512-byte blocks (by
                    multiplying by 2).

                    By default, the integer specified for count is
```

interpreted as it is written. You can append a unit multiplier to the count value, however, to force the system to interpret count as a larger number. These unit multipliers are as follows:

Multiplier	Interpretation
k or K	Specifies 1000. For example, specifying 2k is interpreted as 2000.
m or M	Specifies 1,000,000. For example, specifying 80M is interpreted as 80,000,000.
g or G	Specifies 1,000,000,000.
t or T	Specifies 10**12.
p or P	Specifies 10**15.

type specifies the type of limit. Possible type specifications are as follows:

type	Interpretation
s or soft	Specifies that the samquota command is being used to reset a soft limit.
h or hard	Specifies that the samquota command is being used to reset a hard limit,
u or inuse	Specifies that the samquota command is being used to reset the in-use counter. Typically, this is set only by the samfsck(1M) command and other system administration tools.

scope specifies the scope of the limit. Possible scope specifications are as follows:

scope	Interpretation
o or online	Specifies that the samquota command is being used to reset an online limit. There is no difference between online and total file usage.
t or total	Specifies that the samquota command is being used to reset a total limit. There is no difference between online and total file usage.

If no scope is specified both the online and total limits are set.

```
                    Example.  The following command line sets a soft
                    limit of 120 files for user martha in file system
                    qfs222:

                    samquota -U martha -b 120:s /qfs222

          -g        Returns group quota statistics for file.  This
                    option is not allowed in combination with the -G
                    option or any of the setting options.

          -h        Provides a brief usage summary.

          -i        Zeros all limits.  This option reinitializes the
                    quota specifications by clearing all fields in the
                    quota records except the in-use fields.  It then
                    resets the fields to conform to the new
                    specifications on the command line.

          -k        Specifies that the command interpret or display
                    all storage units (block quantities) in units of
                    1024-byte blocks.  When specified, all information
                    on the command line is assumed to be in units of
                    1024 bytes, and all information is returned in
                    multiples of 1024 bytes.

                    Example 1.  The following command line specifies a
                    hard quota limit of 256,000 1024-byte blocks (or,
                    equivalently, 512,000 512-byte blocks) for group
                    adm, in file system qfs4:

                    samquota -G adm -k -b 256k:hard /qfs4

                    Example 2.  The following command line sets a soft
                    limit of 120 1024-byte blocks (or, equivalently,
                    240 512-byte blocks) to be occupied by the files
                    for user fred in file system qfs2:

                    samquota -U fred -k -b 120:soft /qfs2

          -p        Writes updated quota statistics to stdout if you
                    are changing preestablished quota values or
                    limits.

          -t interval:scope
                    Specifies the time to be used for the soft limit
                    grace periods.

                    interval specifies the interval to use for the
                    grace periods.  By default, the integer specified
                    for interval is interpreted in units of seconds.
                    You can append a unit multiplier to the interval
                    value, however, to force the system to interpret
                    interval as a larger unit.  These unit multipliers
                    are as follows:

                    Multiplier     Interpretation

                    w              Specifies weeks.  For example,
                                   specifying 10w is interpreted as
                                   ten weeks.

                    d             Specifies days.

                    h             Specifies hours.

                    m             Specifies minutes.

                    s (default)   Specifies seconds.

                    The interval must be an integer number in the
                    following range:
                    0 < interval < (2**31) - 1.

                    Note that (2**31) - 1 = 2,147,483,647, which means
                    that the maximum specification, in seconds, would
                    be 2147483647, which is about 68 years.

                    Example.  The following command line specifies an
                    interval of 7 days and 12 hours for the online and
                    total grace periods of user adele in the myqfs
                    file system:

                    samquota -U adele -t 7d12h /myqfs

          -u         Returns user quota statistics for the owner of
                    file.  This option is not allowed in combination
                    with the -U option or any of the setting options.

          -w         Suppresses messages.  By default, samquota
                    generates warning messages and requests
                    confirmation before changing any quota values
                    maintained by the system.  When this option is
                    specified on the command line in conjunction with
                    the -b, -f, or -x options, it suppresses both the
                    warning messages and the confirmation requests.

          -x action:scope
                    Adjusts the soft limit grace period timers.  After
                    a user reaches a soft limit, a certain amount of
                    time can elapse before a user is not allowed to
                    create any more files in the file system.  This
                    option allows you to override the existing quota
                    mechanism and temporarily respecify the
                    consequences of having reached the soft limit.

                    action specifies what to do with the grace period
                    timer.  Note that the soft limit grace period is
                    set with the -t option.  Possible action
                    specifications are as follows:

                    action        Interpretation

                    clear         Specifies that the current grace
                                   period be ended and the grace
                                   period counter be reset to zero.

                                   The grace period counter is
                                   restarted the next time a file or
                                   block is allocated.

                    reset          Specifies that the current grace
                                   period be ended and that the grace
                                   period counter be restarted
                                   immediately.

                    expire         Specifies that the current grace
                                   period be ended and that no new
                                   files or blocks be allocated until
                                   the user, group, or admin set frees
                                   blocks and/or files and is again
                                   under the soft limit.

                    interval       interval specifies the interval to
                                   use for the grace period.
                                   Specifying an interval sets the
                                   grace period to expire at a new
                                   time.  The interval must be an
                                   integer number in the following
                                   range:
                                   0 < interval < (2**31) - 1.

                                   Note that (2**31) - 1 =
                                   2,147,483,647, which means that the
                                   maximum specification, in seconds,
                                   would be 2147483647, which is about
                                   68 years.

                                   The timer is set to the given
                                   value, and starts counting
                                   immediately.  If the quota goes
                                   under the soft limit, it will be
                                   reset to zero at that time.

                                   By default, the integer specified
                                   for interval is interpreted in
                                   units of seconds.  You can append a
                                   unit multiplier to the interval
                                   value, however, to force the system
                                   to interpret interval as a larger
                                   unit, and can concatenate these
                                   units.  These unit multipliers are
                                   as follows:

                                   Multiplier     Interpretation

                                   w              Specifies weeks
                                                  (times 7*24*60*60).
                                                  For example,

                                                  specifying 10w is
                                                  interpreted as ten
                                                  weeks or
                                                  10*7*24*60*60
                                                  seconds.

                                   d              Specifies days
                                   (times 24*60*60).

                     h             Specifies hours
                                   (times 60*60).

                     m             Specifies minutes
                                   (times 60).

                     s (default)   Specifies seconds.

               Example.  Admin set pubs is over its soft limit on
               file system qfs50, and its grace period has
               expired.  You can reset the grace periods by using
               the following command:

               samquota -x 1d2h -A pubs /qfs50

               If the preceding command is executed at 1100 on
               Thursday, the grace period for pubs is reset to
               expire at 1300 on Friday.

-A adminsetID
          Generates a quota report for an admin set, or,
          when specified in conjunction with options that
          reset values, resets the values for the admin set
          specified.  Specify an integer for the adminsetID.

-G groupID
          Generates a quota report for a group, or when
          specified in conjunction with options that reset
          values, resets the values for the group specified.
          Specify an integer identifier or a group name for
          the groupID.

-O        Lists only online values in reports.  The default
          is to list both online and total values.

-U userID Generates a quota report for a user, or, when
          specified in conjunction with options that reset
          values, resets the values for the user specified.
          Specify an integer identifier or a user name for
          the userID.

file      Specifies that the quota information pertain to a
          specific file.  A user is allowed to examine the
          group, user, or admin set quotas of any file for
          which the user has write permissions.  The
          information displayed differs depending on whether
          or not the command is issued by a user who has
          write permission to file, as follows:

          o  If the user issuing this command has write
             permission to file, the command generates
             information on the applicable admin set, group,
             and user quotas that apply to file.

          o  If the user issuing this command does not have
             write permission to file, the command generates
             information for only the user's user ID and
```

```
                  group ID quotas for the file system on which
                  file resides.

EXAMPLES
     Example 1.  The following command initializes a quota for
     group sam on the file system mounted on /qfs1:

     server# samquota -G sam -f 1000:s -f 1200:h -b 100k:s -b 120k:h -t 1d /qfs1

     The group is given the following:

     o  Soft limits of 1000 files and 100,000 512-byte blocks
        (about 50 megabytes)

     o  Hard limits of 1200 files and 120,000 512-byte blocks

     o  A grace period of 1 day (24 hours)

     Example 2.  The following example initializes a quota for
     admin set 17 on the file system that /qfs1/sol is part of:

     server# samquota -A 17 -k -f 10k:s -f 20k:h -b 10m:s -b 15m:h -t 1w /qfs1/sol

     The admin set is given the following:

     o  Soft limits of 10,000 files and 10,000,000 1024-byte
        blocks (10.24 gigabytes)

     o  Hard limits of 20,000 files and 15,000,000 1024-byte
        blocks (15.36 gigabytes)

     o  A grace period of 1 week (168 hours)

EXIT STATUS
     This command returns the following:

     o  0 on successful completion.

     o  1 on a usage or argument error.

     o  10 on an execution error.

FILES
     filesytem/.quota_a  Admin set quota information

     filesystem/.quota_g Group quota information

     filesystem/.quota_u User quota information

SEE ALSO
     squota(1)

     samfsck(1M)

     passwd(4) - User ID information

     group(4) - Group ID information

DIAGNOSTICS
```

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

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

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

samquotastat(1M)

```
NAME
     samquotastat - Reports on active and inactive file system
     quotas

SYNOPSIS
     samquotastat [-a] [-g] [-h] [-u] file

AVAILABILITY
     SUNWsamfs

     SUNWqfs

DESCRIPTION
     The samquotastat command reports whether user, group, or
     admin set quotas are enabled on the file system that
     contains file.  If only the file argument is specified,
     output is generated as if the -a, -g, and -u arguments had
     all been specified.  This command accepts the following
     arguments:

     -a        Generates information on admin set quotas.

     -g        Generates information on group quotas.

     -h        Generates a brief usage summary.

     -u        Generates information on user quotas.

     file      Specify either a specific file name, a path to a
               file, or the file system mount point.  If a file
               name or path to a file is specified, the command
               generates the report for the file system in which
               the file resides.

EXAMPLES
     server% samquotastat /qfs1
     admin quota enabled
     group quota enabled
     user quota disabled

EXIT STATUS
     This command exits with a status of zero if any queried
     quota types are enabled.

SEE ALSO
```

squota(1).

samquota(1M), samfsck(1M).

NOTES

samset(1M)

NAME
samset - Change the Sun QFS or SAM-QFS environment

SYNOPSIS
```
samset [keyword [parameter...]]
```

AVAILABILITY
SUNWsamfs

DESCRIPTION
samset is used to change or display variables that control SSun QFS or SAM-QFS operation. Without any arguments, samset displays current settings to stdout. If samset is executed with a keyword but with no parameter..., then the current value for just that keyword is displayed to stdout.

The keywords all have values assigned to them at startup. These values come from the defaults.conf file. samset allows you to change keywords while sam-fsd is running. Any changes made remain effective only during the current instance of sam-fsd; values revert to the defaults in defaults.conf at the next startup.

The following keywords are supported:

attended yes

attended no
attended tells the Sun QFS or SAM-QFS library daemon if an operator is available to manually mount media. Regardless of the attended setting, requests for media which are mounted in a drive, or present in a media changer, will be satisfied as soon as possible. attended affects the behavior of Sun QFS or SAM-QFS library daemon when a medium is requested which is not currently present in either a manually mounted drive, or in a library. The usual action taken by the library daemon when such a request occurs is to place it into the preview display (see samu (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 reques-

```
                ter.

          exported_media +u eq...

          exported_media -u eq...
                This  option  controls  the  flagging   of   media
                exported   (see   export(1M))   from   the  listed
                libraries as unavailable (+u) or available (-u) in
                the historian's catalog.  See attended, above, for
                the effect of this flag.  The setting of the  flag
                for  a  given  medium  may be changed after export
                using chmed.

          idle_unload
                This is the time (in seconds) that a media changer
                controlled  device may be idle before the media in
                that device is unloaded.  A  value  of  zero  will
                disable this feature.

          labels label-option
                This  option  applies   only   to   barcode-reader
                equipped tape libraries.

                The media daemon can obtain the  tape  label  from
                the  upper-cased characters of the tape's barcode.
                label-option may be: barcodes, to  use  the  first
                six   characters   of   the   barcode   as  label;
                barcodes_low, to use the trailing six  characters;
                or read, to disable barcode processing and to read
                the magnetic label from the tape.

                When labels is set to  barcodes  or  barcodes_low,
                any tape robotically mounted for a write operation
                that is write enabled, unlabeled, has  never  been
                mounted  before,  and  has a readable barcode will
                have a magnetic label written before the write  is
                started.

          stale_time minutes
                Sets the  amount  of  time  (in  minutes)  that  a
                request  for  media will wait in the preview table
                before being canceled with  an  ETIME.   The  file
                system  will  react  to an ETIME error in the same
                way as an ESRCH error (see attended, above).

          timeout seconds
                Sets the time (in seconds) that will be allowed to
                elapse  between  I/O requests for direct access to
                removable media (see request(1)).   If  a  process
                fails  to  issue the next I/O to the device within
                this time, the device will be closed and,  on  the
                next I/O, the process will receive an ETIME error.
                A value of 0 implies no timeout will occur.

          debug     debug manipulates the debug/trace flags within Sun
                QFS  or  SAM-QFS  environments 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 Oracle Corporation support.

debug     This  is  catch-all  for  messages  that
          might  be  of  interest but generally do
          not show a problem.

moves     Log move-media commands issued to  media
          changers.

events    This  should  only  be   used   by   Sun
          Microsystems  analysts to trace the flow
          of events used by the media changer dae-
          mons.   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  exe-
          cuted  by a device.  This information is
          appended to any  existing  message.   If
          the  length  of the existing message and
          the cdb would overflow the message area,
          the  cdb  is not displayed.  The message
          area for a device  can  be  viewed  with
          samu  (see  samu(1M))  in the "s" or "r"
          displays.
          messages  This is  used   by   Sun   Microsystems
                    analysts  to  trace the flow of messages
                    used  by  the  media  changer   daemons.
                    These  messages  are coded and of little
                    use to customers.   These  messages  are
                    logged to syslog at LOG_NOTICE priority.

          migkit    Log events connected with  the  Sun  Sam
                    Migration Toolkit.

          mounts    Log media mount requests.

          opens     Log open and close  of  removable  media
                    devices.

          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 dev-
                    ice  after  this  flag  is  set.   It is
                    closed when the option  is  cleared  and
                    the next I/O to that device occurs.  Sun
                    Microsystems does not recommend  running
                    with  this option for long periods.  The
                    format of the trace information is:

                      struct {
                        int    eq;      /* equipment number */
                        int    what;    /* 0 - issue, 1 - response */
                        time_t now;     /* unix time */
                        int    fd;      /* the fd the ioctl was issued on */
                        char   cdb[12]; /* the cdb */
                        char   sense[20]; /* returned sense(valid if what=1) */
                      }cdb_trace;

                    Oracle Corporation does not recommend set-
                    ting  this  option  indiscriminately, as
                    large output files are quickly produced.

          stageall  This  should  be  used   only   by   Sun
                    Microsystems  analysts to trace stageall
                    processing.

     devlog eq [ event ...]
          devlog manipulates the device log event flags  for
          device  eq.   eq  is either an equipment number or
          "all";  if "all", then the flags are set or listed
          for all devices.  These flags control which events
          get written to the device log files.  [ event ...]
          is  a space separated list of event names.  To set
          an event flag, give its name.  To  clear  a  flag,
          give  its  name  prefixed  with a '-'.  The events
          are:

          all       Turn on all events.

          none      Turn off all events.

          default   Set the event flags to the default which
                    are: err, retry, syserr, and date.

          detail    events which may be used  to  track  the
                    progress of operations.

          err       Error messages.

          label     Labeling operations.

          mig       Migration toolkit messages.

          msg       Thread/process communication.

          retry     Device operation retries.

          syserr    System library errors.

          time      Time device operations.

          module    Include module name and source  line  in
                    messages.

          event     Include the event name in the message.

          date      Include the date in the message.

tapealert eq [on|off|default]
          tapealert allows the user  to  enable  or  disable
          support for device implemented TapeAlert.

          eq        is either an equipment number or  "all";
                    if  "all",  then  the  flags  are set or
                    listed for all devices.

          on        Enable TapeAlert if the device  supports
                    it.

          off       Disable requesting TapeAlert information
                    from the device.

          default   Return TapeAlert to the factory setting.

sef eq [on|off|default] interval
          sef allows the user to enable or  disable  support
          for tape drive implemented Log Sense delivered via
          sysevents.

          eq        is either an equipment number or  "all";
                    if  "all",  then  the  flags  are set or
                    listed for all devices.
```

```
               on        Enable requesting tape drive  Log  Sense
                         sysevents if the drive supports it.

               off       Disable requesting tape drive Log  Sense
                         sysevents.

               default   Return tape drive Log Sense sysevents to
                         the factory setting.

               interval  Tape drive Log Sense polling interval in
                         seconds.   A  value  of 300 is a polling
                         interval once  every  five  minutes.   A
                         string  value  of  "once"  specifies one
                         time just before media unload and is the
                         default.   A  value of 3600 is a polling
                         interval once every hour.  The  smallest
                         polling interval is five minutes.
```

SEE ALSO
     request(1), chmed(1M), export(1M), samu(1M), defaults.conf(4), mcf(4), tapealert(1M), sefsysevent(4).

NOTES
     A complete description of SEF sysevents is in the Sun Storage Archive Manager (SAM-QFS) Configuration and Administration Guide.

samsharefs(1M)

NAME
     samsharefs - Manipulates the Sun QFS shared file system configuration

SYNOPSIS
```
     samsharefs [-f host] [-h] [-o host] [-q] [-R] [-s host] [-u]
     fs_name
```

AVAILABILITY
     SUNWsamfs
     SUNWqfs

DESCRIPTION
     The samsharefs command prints and modifies the host configuration for a Sun QFS shared file system. The printed hosts configuration identifies the metadata server and the client hosts included in the Sun QFS shared file system. This command is only valid from the metadata server or potential metadata server.

     You create an initial hosts configuration file using vi(1) or another text editor. The sammkfs(1M) command reads this initial hosts configuration from /etc/opt/SUNWsamfs/hosts.fs_name when the SAM-QFS shared file system is created.

     To subsequently change the host configuration you must use

```
     the samsharefs command.  Typically, you use an editor to
     edit the ASCII hosts configuration as printed by the
     samsharefs command and use the samsharefs command to update
     the file system host configuration.

OPTIONS
     This command accepts the following options:

     -f host   Marks host "off" in the hosts file.  This option
               rewrites the on-disk hosts file and causes the
               SAM-QFS daemon to reread the hosts file.  Marking
               the host off disallows that host to access the
               specified fs_name, and is reversed by using the -o
               option.  -f is incompatible with -u parameter.

               Marking a host client "off" allows that client to
               remain in the host file, but not access the
               specified file system.  It is intended to be used
               to remove clients and not require the file system
               to be unmounted on all other clients.  The removed
               host remains in the host file as a placeholder and
               can later be restored by using the -o parameter.
               Note that the client will need to be marked "off"
               from the metadata server for each file system that
               it mounts.

               The host client's "on" or "off" status can be seen
               in the 4th column of the host file (as printed by
               the samsharefs command).  For backwards
               compatibility, a "-", "0", or blank in this column
               indicates "on".  Also, if a client is marked off,
               it is indicated by an "OFF" flag on the samu "g"
               display (or the samcmd g command).

               CAUTIONS & LIMITATIONS:  A file system that is to
               be shared to other clients must be mounted on the
               metadata server and also be mountable to potential
               metadata servers.  Thus a client that is an actual
               metadata server cannot be marked off.

               The only supported way to mark a client host off
               is to unmount its file systems and shutdown and
               halt the client.  Then issue the samsharefs -f
               host fs command from the metadata server.

               The only supported way to restore a client host is
               to mark the client host on (using the -o
               parameter) prior to booting that client.  The
               client is then free to remount the affected file
               system.

               Clients, while marked off, will not be able to
               contact the metadata server for that file system.
               If a marked-off client tries to contact the
               metadata server for that file system, its messages
               will be discarded and system hangs may occur.  The
               result of trying to talk to a metadata server from
               a marked-off client is undefined and not
               supported.
     -h        Writes a short usage message to stdout.

     -o host   Marks host "on" in the hosts file.  This option
               rewrites the on-disk hosts file and causes the
               SAM-QFS daemon to reread the hosts file.  Marking
               the host on allows that host to access to the
               specified fs_name, and reverses the effect of the
               -f option.  -o is incompatible with -u parameter.
               See -f option above for cautions & limitations.

     -q        Suppresses host configuration output.  By default,
               the command writes the file system host
               configuration, possibly modified, to stdout.

     -R        Specifies that the file system's host
               configuration should be manipulated using the raw
               disk device associated with the file system,
               rather than the file system interfaces.  This
               option can be used to change hosts information
               when the file system is not or cannot be mounted.
               This option can also be used to change hosts
               information when the file system is mounted, but
               the active metadata server is down.

               CAUTION:  This option must not be executed on a
               potential metadata server to change the metadata
               server host without first stopping, disabling, or
               disconnecting the active metadata server.  Doing
               so will cause file system corruption.

     -s host   Sets the server flag for the specified host in the
               system configuration.  This option declares host
               to be the new metadata server host.  All other
               hosts's server flags are cleared.

     -u        Specifies that the file system's configuration is
               to be updated from
               /etc/opt/SUNWsamfs/hosts.fs_name.  When updating
               the configuration of a mounted file system, new
               host entries can only be added to the end of the
               existing configuration.  If the server or any
               host's position differs between hosts.fs_name and
               the active configuration (i.e., the order of the
               hosts is changed), the command issues an error
               message and exits; changing these characteristics
               can be done safely only on an idle, unmounted file
               system.  (See the -R option.)

     fs_name   Specifies the family set name of the Sun QFS
               shared file system.

EXAMPLES
     Example 1.  The following example shows how to use the
     samsharefs to examine the hosts information on a mounted Sun
     QFS shared file system:

     tethys# samsharefs share1
     #
```

```
# Host file for family set 'share1'
#
# Version: 4    Generation: 14    Count: 3
# Server = host 0/titan, length = 112
#
titan titan.xyzco.com 1 0
tethys tethys.xyzco.com 2 0
mimas mimas.xyzco.com 0 0
```

Example 2. The following example shows how the hosts configuration can be modified to add new hosts to the shared file system. The administrator has edited /etc/opt/SUNWsamfs/hosts.share1 and added new hosts for the shared file system as shown. samsharefs is then run with the -u option to update the (mounted) file system's configuration.

```
titan# samsharefs share1
#
# Host file for family set 'share1'
#
# Version: 4    Generation: 14    Count: 3
# Server = host 0/titan, length = 112
#
titan titan.xyzco.com 1 0
tethys tethys.xyzco.com 2 0
mimas mimas.xyzco.com 0 0

titan# cat /etc/opt/SUNWsamfs/hosts.share1
#
# New share1 config, adds dione and rhea
#
titan   titan.xyzco.com 1 0 server
tethys tethys.xyzco.com 2 0
mimas   mimas.xyzco.com 0 0
dione   dione.xyzco.com 0 0
rhea     rhea.xyzco.com 0 0

titan# samsharefs -u share1
#
# Host file for family set 'share1'
#
# Version: 4    Generation: 15    Count: 5
# Server = host 0/titan, length = 162
#
titan titan.xyzco.com 1 0
tethys tethys.xyzco.com 2 0
mimas mimas.xyzco.com 0 0
dione dione.xyzco.com 0 0
rhea rhea.xyzco.com 0 0
```

Example 3. The following example shows how the hosts configuration can be modified to change the Sun QFS shared file system server while the file system is mounted.

```
tethys# samsharefs -s tethys share1
#
# Host file for family set 'share1'
#
```

```
# Version: 4    Generation: 16    Count: 5
# Server = host 0/titan, length = 162
# Pending Server = host 1/tethys
#
titan titan.xyzco.com 1 0
tethys tethys.xyzco.com 2 0
mimas mimas.xyzco.com 0 0
dione dione.xyzco.com 0 0
rhea rhea.xyzco.com 0 0
```

Example 4. The following example shows how the hosts configuration can be modified to add a new Sun QFS shared file system server. Because the new server's entry is being inserted into the existing list rather than appended to the end, the file system must be unmounted on all hosts before executing this command, and the -R option must be specified. Note also that this command changes the file system server back to titan (from tethys).

```
tethys# samsharefs -R share1
#
# Host file for family set 'share1'
#
# Version: 4    Generation: 17    Count: 5
# Server = host 1/tethys, length = 162
#
titan titan.xyzco.com 1 0
tethys tethys.xyzco.com 2 0
mimas mimas.xyzco.com 0 0
dione dione.xyzco.com 0 0
rhea rhea.xyzco.com 0 0

tethys# cat /etc/opt/SUNWsamfs/hosts.share1
#
# New share1 config, adds server iapetus
#
titan     titan.xyzco.com 1 0 server
tethys   tethys.xyzco.com 2 0
iapetus iapetus.xyzco.com 3 0
mimas     mimas.xyzco.com 0 0
dione     dione.xyzco.com 0 0
rhea       rhea.xyzco.com 0 0

tethys# samsharefs -u -R share1
#
# Host file for family set 'share1'
#
# Version: 4    Generation: 18    Count: 6
# Server = host 0/titan, length = 192
#
titan titan.xyzco.com 1 0
tethys tethys.xyzco.com 2 0
iapetus iapetus.xyzco.com 3 0
mimas mimas.xyzco.com 0 0
dione dione.xyzco.com 0 0
rhea rhea.xyzco.com 0 0
```

FILES

The hosts configuration for a Sun QFS shared file system is initialized from:

/etc/opt/SUNWsamfs/hosts.fs_name

This file is used at the time of file system creation by sammkfs(1M) and subsequently when the -u option is specified to samsharefs(1M).

NOTE

In SAM-QFS shared file system environments, archiving operations should be stopped on the metadata server before changing the metadata server.

CAUTION

The -R option must not be used on a mounted file system to change the metadata server host without first stopping, disabling, or disconnecting the active metadata server and ensuring that it is restarted before accessing the file system again. Doing so will cause file system corruption.

SEE ALSO

sammkfs(1M).

samsnoop(1M)

NAME

samsnoop - Sun QFS version of snoop

SYNOPSIS

```
samsnoop [-aCDNPSvV]  [-t [r | a | d ] [ -c maxcount ]
     [ -d device  ]   [  -i filename  ]  [ -n filename ]
     [ -o filename ] [ -p first [ , last ] ]  [ -s snaplen ]
     [ -x offset [ , length ] ]  [ expression ]
```

AVAILABILITY

SUNWsamfs

DESCRIPTION

samsnoop is a version of snoop(1M) modified to capture and display packets from the Sun QFS shared file system. The arguments are identical to those of snoop(1M).

SEE ALSO

snoop(1M)

samstorade(1M)

NAME

samstorade - StorADE API

SYNOPSIS

```
samstorade [-r hostname] [-t timeout] [-s xml_message] [-d]
```

```
AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The samstorade program is a XML interface for the Sun
     StorADE (Storage Automated Diagnostic Environment) program
     to access SAM-QFS attributes and health information.

     The XML interface uses messages contained in the message DTD
     /opt/SUNWsamfs/doc/message.dtd.

OPTIONS
     The samstorade command can be customized with the following
     options:

     -r hostname
               Specifies remote SAM-QFS host to query.  The
               default hostname is "localhost".

     -t timeout
               Specifies response timeout in milliseconds.  The
               default timeout value is 5 seconds.

     -s xml_message
               Specifies a valid XML message defined by the
               message DTD /opt/SUNWsamfs/doc/message.dtd that is
               sent to the sam-amld daemon.  An example message
               that can be sent is <message id="devent"
               version="1.0"/>.

     -d        This is the default behavior for the samstorade
               command.  The /opt/SUNWsamfs/doc/message.dtd
               message wrapper is removed returning the
               /opt/SUNWsamfs/doc/samfm.dtd message payload.

FILES
     /opt/SUNWsamfs/sbin/samstorade

     /opt/SUNWsamfs/doc/message.dtd

     /opt/SUNWsamfs/doc/samfm.dtd

     /opt/SUNWsamfs/doc/pkg.mod

     /opt/SUNWsamfs/doc/devent.mod

SEE ALSO
     StorADE, rasagent(1M),
```

samtrace(1M)

```
NAME
     samtrace - Dumps the Sun QFS or Sun SAM-QFS trace buffer

SYNOPSIS
     samtrace [ -d corefile -n namelist ] [ -s ] [ -t ] [ -v ]  [
     -V ] [ -f ]

     samtrace -k suffix [ -s ] [ -v ] [ -V ] [ -f ]

     samtrace -O file

     samtrace -I file [ -f ]

     samtrace -c file [ -b bufs ] [ -p secs ] [ -T ticks ]

     samtrace -i file [ -f ]

AVAILABILITY
     SUNWqfs

     SUNWsamfs

DESCRIPTION
     samtrace dumps the contents of  the  trace  buffer  for  the
     mounted file system.

OPTIONS
     -b bufs
          When used with the -c option, this sets the  number  of
          per-CPU  trace  read  buffers  allocated by samtrace to
          bufs.  The value of bufs must be at least 3,  and  must
          be no more than 64.  The default is 5.

     -c file
          Trace entries are continuously  copied  from  the  live
          kernel into file until the command is killed.  Periodi-
          cally, file is written with the binary contents of  the
          kernel trace buffer; the kernel trace buffer’s contents
          are cleared after each copy is made.   The  entries  in
          file are written in time order, oldest first.

     -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  run-
          ning 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.

     -k suffix
          Indicates that the corefile and namelist have the names
          ’vmcore.suffix’ and ’unix.suffix’, respectively.
```

-i file
: file must be a file created with the -c continuous trace option. samtrace reads file and writes a readable copy of the binary records in file to the standard output.

-I file
: file must be a file created with the -O trace option. samtrace reads file and writes a sorted, readable copy of the binary records in file to the standard output.

-O file
: The system trace buffers are copied to file. This file can later be translated for human interpretation with the -I option.

-p secs
: When used with the -c option, sets an alarm signal for secs seconds after samtrace starts. This allows for automatic termination of continuous samtrace operation.

-s
: Dumps the sam-amld command queue. Includes -v output.

-T ticks
: When used with the -c option, sets the default interval between reads of the kernel trace buffer to ticks scheduler ticks. The contents of the kernel trace buffers are by default copied to a samtrace buffer whenever the trace buffer fills half-way, or 100 ticks (1 second) has passed, whichever occurs first.

-t
: Suppress trace output. When specified alone, displays only table address information. Typically used in conjunction with -v or -V to see verbose information without traces.

-v
: Verbose option, excluding inode free and hash chains.

-V
: Verbose option, including inode free and hash chains. Includes -v output.

-f
: Decodes flag bits in the trace output.

NOTE

samtrace is a utility that is used to provide Sun Microsystems analysts with troubleshooting information. It is not intended for general use at your site.

FILES

/dev/kmem	Special file that is an image of the kernel virtual memory of the computer.
/dev/mem	Special file that is an image of the physical memory of the computer.
/dev/ksyms	Character special file of kernel symbols.

samu(1M)

NAME
     samu - Sun QFS and SAM-QFS operator utility

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

AVAILABILITY
     SUNWqfs
     SUNWsamfs

DESCRIPTION
     samu is a full screen operator interface for Sun QFS and SAM-QFS environments. It has a number of displays that show the status of file systems and devices and allows the operator to control file systems and removable media devices.

OPTIONS

-d c	Specifies the initial display when samu starts execution. See DISPLAYS below.
-r i	Specifies the time interval in seconds for refreshing the display window.
-c string	Specifies an initial command string that should be executed when samu starts execution.
-f cmd-file	Specifies a file from which to read samu commands. Each line in the file is a command.

CONTROL KEYS
     The following "hot" keys are available for all displays:

q	Quit
:	Enter command
space	Refresh display
control-l	Refresh display (clear)
control-r	Enable/disable refresh (default is enabled)

     The following keys perform the listed functions for each of the displays shown:

Key	Function	Display
control-f	Next file system	:a,a,g
	Page forward	c,h,o,p,s,t,u,v,w,A,J,K,M,P
	Next stage request	n
	Next inode	I
	Next sector	S
	Next equipment	T,U
	Next filesystem	N
control-b	Previous file system	:a,a,g
	Page backward	c,h,o,p,s,t,u,v,w,A,J,K,M,P
	Previous stage request	n
	Previous inode	I
	Previous sector	S

```
                    Previous equipment                     T,U
                    Previous filesystem                    N

         control-d  Half-page forward                      c,p,s,u,w,A,J,M
                    Next robot catalog                     v
                    Page forward                           g,h,S
                    Page arcopies forward                  a
                    Page stage queue forward               n
                    Page partitions forward                N

         control-u  Half-page backward                     c,p,s,u,w,A,J,M
                    Previous robot catalog                 v
                    Page backward                          g,h,S
                    Page arcopies backward                 a
                    Page stage queue backward              n
                    Page partitions backward               N

         control-k  Advance display format                 A,I,S
                    Select (manual,robotic,both,priority)  p
                    Advance sort key                       v
                    Toggle path display                    n,u,w

         control-i  Detailed (2-line) display format       v,D
                    Detailed status interpretations        g,n,N

         control-j  Size display unit (base 2 or 10)       D,:a,l,n,m,v

         1-7        Select sort key                        v

         /          Search for VSN                         v

         %          Search for barcode                     v

         $          Search for slot                        v
```

The sort selections for the v display are: 1 slot, 2 count, 3 usage, 4 VSN, 5 access time, 6 barcode, 7 label time.

DISPLAYS

The following displays are available. Those displays marked with '*' are the only ones available for Sun QFS. All others are available in SAM-QFS environments only if samd start has been executed.

```
          a@   Display archiver status
          c    Display configuration            C    Memory
          d*   Display tracing info.            D    Display disk volume dictionary
          f*   Display filesystem info.         F    Optical disk label
          g*   Display client information
          h*   Display help information
          l@   Display usage information        I*   Inode
          m*   Display mass-storage status      J    Preview shared memory
          n@   Display staging activity         K    Kernel statistics
          o    Display optical disk status      L    Shared memory tables
          p    Display mount request preview    M    Shared memory
          r    Display removable media          N*   File system parameters
          s    Display device status summary    P    Active Services
          t    Display tape status              R    SAM-Remote info
          u    Display stage queue              S    Sector data
          v    Display robot VSN catalog        T    SCSI sense data
          w    Display pending stage queue      U    Device table
```

COMMANDS

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

     Archiver commands:
         aridle [ dk | rm | fs.fsname ]         Idle archiving
         arrerun                                Soft restart archiver
         arrestart                              Restart archiver
         arrmarchreq fsname.[* | archreq]       Remove ArchReq
         arrun [ dk | rm | fs.fsname ]          Start archiving
         arscan fsname[.dir | ..inodes][int]    Scan filesystem
         arstop [ dk | rm | fs.fsname ]         Stop archiving
         artrace [fs.fsname]                    Trace archiver

     Display control commands:
         refresh i      Set refresh time
         a filesystem   Select detailed "a" display
         n media        Set n display media selection
         p media        Set p display media selection
         r media        Set r display media selection
         u media        Set u display media selection
         v eq           Set v display robot catalog
         w media        Set w display media selection

     Device commands:
         devlog     eq [option ...]   Set device logging options
         idle       eq                Idle equipment
         off        eq                Off equipment
         on         eq                On equipment
         readonly   eq                Mark equipment read-only
         ro         eq                Mark equipment read-only
         unavail    eq                Mark equipment unavailable
         unload     eq                Unload mounted media/magazine

     File System commands - miscellaneous:
         stripe          eq value      Set stripe width
         suid            eq            Turn on setuid capability
         nosuid          eq            Turn off setuid capability
         sync_meta       eq value      Set sync_meta mode
         atime           eq value      Set access time (atime) update mode
         trace           eq            Turn on file system tracing
         notrace         eq            Turn off file system tracing
         add             eq            Add eq to mounted file system
         remove          eq            Remove eq; copy files to eqs with ON state
         release         eq            Release eq; mark files offline
         alloc           eq            Enable allocation on partition
         noalloc         eq            Disable allocation on partition
         def_retention   eq interval   Set default WORM retention time

     File System commands - SAM-QFS Commands:
         hwm_archive          eq            Turn on hwm archiver start
         nohwm_archive        eq            Turn off hwm archiver start
         maxpartial           eq value      Set maximum partial size in kilobytes
         partial              eq value      Set size to remain online in kilobytes
         partial_stage        eq value      Set where to start staging if partial
         stage_flush_behind   eq value      Set stage flush behind size in kilobytes
         stage_n_window       eq value      Set direct stage size in kilobytes
```

```
    thresh               eq high low   Set high and low release thresholds

File System commands - I/O:
    dio_rd_consec       eq value   Set number of consecutive dio reads
    dio_rd_form_min     eq value   Set size of well-formed dio reads
    dio_rd_ill_min      eq value   Set size of ill-formed dio reads
    dio_wr_consec       eq value   Set number of consecutive dio writes
    dio_wr_form_min     eq value   Set size of well-formed dio writes
    dio_wr_ill_min      eq value   Set size of ill-formed dio writes
    flush_behind        eq value   Set flush behind value in kilobytes
    forcedirectio       eq         Turn on directio mode
    noforcedirectio     eq         Turn off directio mode
    force_nfs_async     eq         Turn on NFS async
    noforce_nfs_async   eq         Turn off NFS async
    readahead           eq value   Set maximum readahead in kilobytes
    writebehind         eq value   Set maximum writebehind in kilobytes
    sw_raid             eq         Turn on software RAID mode
    nosw_raid           eq         Turn off software RAID mode
    wr_throttle         eq value   Set outstanding write size in kilobytes
    abr                 eq         Enable Application Based Recovery
    noabr               eq         Disable Application Based Recovery
    dmr                 eq         Enable Directed Mirror Reads
    nodmr               eq         Disable Directed Mirror Reads
    dio_szero           eq         Turn on dio sparse zeroing
    nodio_szero         eq         Turn off dio sparse zeroing

File System commands - Sun QFS:
    mm_stripe   eq value   Set meta stripe width
    qwrite      eq         Turn on qwrite mode
    noqwrite    eq         Turn off qwrite mode

File System commands - multireader:
    invalid            eq interval   Set multireader invalidate cache delay
    refresh_at_eof     eq            Turn on refresh at eof mode
    norefresh_at_eof   eq            Turn off refresh at eof mode

File System commands - shared fs:
    minallocsz   eq value      Set minimum allocation size
    maxallocsz   eq value      Set maximum allocation size
    meta_timeo   eq interval   Set shared fs meta cache timeout
    mh_write     eq            Turn on multihost read/write
    nomh_write   eq            Turn off multihost read/write
    aplease      eq interval   Set append lease time
    rdlease      eq interval   Set read lease time
    wrlease      eq interval   Set write lease time

Robot commands:
    audit      [-e] eq[:slot[:side]]            Audit slot or library.
               See auditslot(1M) for information on -e.
    import     eq                               Import cartridge from mailbox
    export     eq:slot                          Export cartridge to mailbox
    export     mt.vsn                           Export cartridge to mailbox
    load       eq:slot[:side]                   Load cartridge in drive
    load       mt.vsn                           Load cartridge in drive
    priority   pid                     newpri   Set load priority for process 'pid'

Stager commands:
    stclear   mt.vsn   Clear stage request
    stidle             Idle staging
```

```
        strun              Start staging

    Miscellaneous commands:
        clear           vsn [index]               Clear load request
        diskvols        volume [+flag | -flag]    Set or clear disk volume dictionary flags
        dtrace          daemon[.variable] value   Set daemon trace controls
        fs              fsname                    Set filesystem (N display)
        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).

mcf(4).

samunhold(1M)

NAME

samunhold - Releases SANergy file holds

SYNOPSIS

```
/opt/SUNWsamfs/sbin/samunhold mntpoint
```

AVAILABILITY

SUNWqfs

SUNWsamfs

DESCRIPTION

The samunhold command can be used to release SANergy file holds. These holds can be detected when attempts are made to unmount a file system with the umount(1M) command. If holds are present, the umount(1M) command generates log messages such as the following:

```
Inode XXXX: held by SAN, refcnt = N
```

SANergy File Sharing uses the following two types of leases, both of which require holds:

o Read leases, which typically expire within a few seconds.

o Write leases, which can extend for as long as an hour.

It is preferable to allow SANergy File Sharing to clean up the leases, but in an emergency, or in case of a SANergy File Sharing system failure, the administrator can use the samunhold command to avoid a reboot.

The samunhold command should only be run when SANergy File Sharing has held inodes and is preventing a file system from being unmounted. Prior to executing this command, the administrator should ensure the following:

```
o There are no SANergy applications running on any client,
  possibly including the server itself.

o The file system in question is not fused on any SANergy
  clients.

o The file system is not NFS mounted.
```

OPTIONS

The samunhold command releases all held inodes (files) on the file system whose root directory is the named mntpoint argument. The samunhold command must be run as root.

EXAMPLES

The following example shows the samunhold command:

```
bilbo# samunhold /sam1
bilbo# umount /sam1
```

SEE ALSO

umount(1M).

save_core.sh(1M)

NAME

save_core.sh - SAM-QFS sam-robotsd(1M) exception notification script

SYNOPSIS

```
/etc/opt/SUNWsamfs/scripts/save_core.sh prg_name pid
severity msg_no msg
```

AVAILABILITY

SUNWsamfs

DESCRIPTION

The /etc/opt/SUNWsamfs/scripts/save_core.sh script is executed by sam-robotsd(1M) after it encounters abnormal or exceptional events. A site-specific version of this script can be substituted in the installed location.

This script labels core files and prevents existing core files from being overwritten as more are generated. As released, /etc/opt/SUNWsamfs/scripts/save_core.sh renames the sam-robotsd(1M) child core files to include the program name, process ID, and date.

OPTIONS

The sam-robotsd(1M) daemon executes /etc/opt/SUNWsamfs/scripts/save_core.sh with the following arguments:

prg_name	The name of the program that is calling this script.
pid	The process ID of the program that is calling this

```
                    script.

          severity  A keyword that identifies the severity and the
                    syslog level of the event.  The keywords are as
                    follows:  emerg, alert, crit, err, warning,
                    notice, info, and debug.

          msg_no    The message number as found in the message
                    catalog.

          msg       The text of the translated message string.  This
                    script expects this message to be in a 2-field
                    format.  The first field indicates the program
                    that caused the core dump.  The second field is
                    the process ID of the program that caused the core
                    dump.

     SEE ALSO
          sam-robotsd(1M).
```

scanner(1M)

```
     NAME
          sam-scannerd - SAM-QFS daemon for manually-mounted devices

     SYNOPSIS
          /opt/SUNWsamfs/sbin/sam-scannerd mshmid pshmid

     AVAILABILITY
          SUNWsamfs

     DESCRIPTION
          sam-scannerd monitors the manually-mounted devices.  It will
          periodically check each device for newly inserted media.  If
          sam-scannerd finds media in the device, it will scan it  for
          a  label.   If  a  label is found, it will check the preview
          table to see if there are any requests for this  media.   If
          requests  are found, the SAM-QFS file system is notified and
          the device is assigned to the request.

          sam-scannerd is started automatically by sam-amld  if  there
          are  any  manually-mounted devices defined in the configura-
          tion file.  See mcf(4).

          mshmid is the id of the master shared memory segment created
          by  sam-amld.  pshmid is the id of the preview shared memory
          segment created by sam-amld.

     SEE ALSO
          sam-amld(1M), mcf(4)
```

scsi_trace_decode(1M)

```
NAME
     scsi_trace_decode - Decodes files produced by enabling the
     trace_scsi option

SYNOPSIS
     /opt/SUNWsamfs/tools/scsi_trace_decode -f trace_file [eq_id]

AVAILABILITY
     SUNWsamtp

DESCRIPTION
     The scsi_trace_decode command decodes the raw SCSI trace
     file produced when the debug trace_scsi option in the
     defaults.conf file is enabled.

OPTIONS
     This command accepts the following options:

     -f trace_file
               Specifies the trace file.  Enter a full path name
               for the trace file.  Typically, the trace file is
               /tmp/sam_scsi_trace_ml_eq_id, where ml_eq_id is
               the Equipment Number of the library that contains
               the devices being traced.

               For example, assume that you have two libraries.
               Their Equipment Numbers are 20 and 50.  Each
               library has two tape drives, and their Equipment
               Numbers are 21, 22, 51, and 52.  You could specify
               that trace information for devices 20, 21, and 22
               go to file /tmp/sam_scsi_trace_20 and that trace
               information for devices 50, 51, and 52 go to file
               /tmp/sam_scsi_trace_50.

     eq_id     Shows entries in trace_file for the device
               identified by Equipment Number eq_id only.  The
               default is to show entries for all Equipment
               Numbers in this trace file.

EXAMPLE OUTPUT
       eq41 Issue 09:22:45 8   cdb: 1b 00 00 00  00 00 00 00  00 00 0000
            Load unload

       eq41 Reply 09:22:45 8   cdb: 1b 00 00 00  00 00 00 00  00 00 00 00
            Load unload
            sense: 71 00 04 00  00 00 00 48  00 00 00 00  44 00 f4 03  0e 16 00 41
                    Sense key 04, ASC 44, ASCQ 00
                    Internal target failure

     The various components of the preceding output are as
     follows:

     Output    Meaning

     eq41      The equipment number involved in the command or
               completion.
     Issue     The scsi command has been issued.

     Reply     The command completed.  Completion status is
               shown.

     09:22:45  The time of day when the command was sent or
               received.

     8         The file descriptor upon which the command was
               issued.  Not useful for non-Oracle Corporation
               analysts.

     cdb       The cdb (command descriptor block): the scsi
               command issued.

     sense     The sense data.  If the command erred, then sense
               data is obtained and displayed.  If the command
               did not err, then this field is all zero.

     Sense key ...
               Decoded sense data, showing the sense key,
               additional sense code, and additional sense code
               qualifier.  These values define the error that
               occurred.

     Internal target failure
               The ASCII representation of the ASC, ASCQ
               information.

SEE ALSO
     defaults.conf(4)
```

sefreport(1M)

```
NAME
     sefreport - Displays the content of the System Error
     Facility (SEF) log

SYNOPSIS
     /opt/SUNWsamfs/sbin/sefreport [-v|-t] -d file

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sefreport command reads the content of a SAM-QFS SEF log
     file and writes its output to stdout in a human-readable
     format.  By default, the log file is
     /var/opt/SUNWsamfs/sef/sefdata.  The SEF log file contains
     the data gathered from the log sense pages of peripheral
     tape devices used by SAM-QFS file systems. For more
     information on the SEF log file, including its content and
     format, see the sefdata(4) man page.

     The sefreport command reads the input file specified by the
```

file argument. If no other options are specified, the sefreport command examines the SEF log file and generates the following information for each record contained in file:

o The first header line states the record number, which is its ordinal position in the file.

o The second header line contains the timestamp of the record, the vendor name of the device from which the log sense data was received, the product name of the device, the revision level of the device's firmware, the string VSN, and the Volume Serial Name (VSN) of the volume mounted in the device when the log sense data was generated.

Following the header lines, the log sense data for each page in the record is printed. For each log sense page, a line identifying the page code is printed, followed by a line of column headings. The data is then printed in three columns per line with the following headings: parameter code, control, and parameter value. All data is generated in hexadecimal notation. For the meanings of the parameter codes, control bits, and parameter values, see your vendor documentation for the specific device.

OPTIONS

This command accepts the following options:

-d Includes additional device information. For each record, the command generates a third header line that identifies the equipment number of the device as configured in the mcf file and the path name of the device.

-t Generates log sense output with text descriptions. On each line of log sense data output, an additional string containing the equipment number, page code, VSN, and parameter code description is printed. The -t option is not used when the -v option is specified.

-v Generates verbose output. On each line of log sense data output, an additional string containing the equipment number, page code, and VSN is printed. This string is enclosed in parentheses and the items are colon-separated.

OPERANDS

This command accepts the following operand, which must be specified:

file Specifies the SEF log file. The SEF log file can be read from its default location (/var/opt/SUNWsamfs/sef/sefdata) or it can be redirected to another file for SEF processing.

EXAMPLES

Example 1. Assume that your system is set up to write SEF values to file /var/opt/SUNWsamfs/sef/sefdata.mid. You have entered the following command to write the SEF data using the report formatter:

```
srvr# sefreport /var/opt/SUNWsamfs/sef/sefdata.mid > ~mydir/sef.short
```

The file ~mydir/sef.short is as follows:

```
Record no. 1
Mon Mar 26 11:17:48 2001  STK      9840             1.25 VSN 002981

   PAGE CODE 2
   param code  control   param value
       00h       74h     0x0
       01h       74h     0x0
       02h       74h     0x0
       03h       74h     0x0
       04h       74h     0x0
       05h       74h     0x40050
       06h       74h     0x0

   PAGE CODE 3
   param code  control   param value
       00h       74h     0x0
       01h       74h     0x0
       02h       74h     0x0
       03h       74h     0x0
       04h       74h     0x0
       05h       74h     0x140
       06h       74h     0x0

   PAGE CODE 6
   param code  control   param value
       00h       74h     0x0

Record no. 2
Mon Mar 26 11:30:06 2001  STK      9840             1.25 VSN 002999

   PAGE CODE 2
   param code  control   param value
       00h       74h     0x0
       01h       74h     0x0
       02h       74h     0x0
       03h       74h     0x0
       04h       74h     0x0
       05h       74h     0x1400a0
       06h       74h     0x0

   PAGE CODE 3
   param code  control   param value
       00h       74h     0x0
       01h       74h     0x0
       02h       74h     0x0
       03h       74h     0x0
       04h       74h     0x0
       05h       74h     0x190
       06h       74h     0x0

   PAGE CODE 6
   param code  control   param value
       00h       74h     0x0
```

```
<<<NOTE:  This output has been truncated for inclusion on this
man page.>>>

Example 2:  Assume that you also need to produce a report
with additional data.  You can use the same log file as in
Example 1, but you want this report to contain more
information than sef.short, so you invoke sefreport with the
-d and -v options.  The following command is entered:

srvr# sefreport -d -v /var/opt/SUNWsamfs/sef/sefdata.mid > ~mydir/sef.long
The file ~mydir/sef.long is as follows:

Record no. 1
Mon Mar 26 11:17:48 2001  STK      9840             1.25 VSN 002981
   Eq no. 32   Dev name /dev/rmt/1cbn

 rec  pg cd   param code  control   param value
   1    2         00h       74h     0x0           (32:2:002981)
   1    2         01h       74h     0x0           (32:2:002981)
   1    2         02h       74h     0x0           (32:2:002981)
   1    2         03h       74h     0x0           (32:2:002981)
   1    2         04h       74h     0x0           (32:2:002981)
   1    2         05h       74h     0x40050       (32:2:002981)
   1    2         06h       74h     0x0           (32:2:002981)

 rec  pg cd   param code  control   param value
   1    3         00h       74h     0x0           (32:3:002981)
   1    3         01h       74h     0x0           (32:3:002981)
   1    3         02h       74h     0x0           (32:3:002981)
   1    3         03h       74h     0x0           (32:3:002981)
   1    3         04h       74h     0x0           (32:3:002981)
   1    3         05h       74h     0x140         (32:3:002981)
   1    3         06h       74h     0x0           (32:3:002981)

 rec  pg cd   param code  control   param value
   1    6         00h       74h     0x0           (32:6:002981)

Record no. 2
Mon Mar 26 11:30:06 2001  STK      9840             1.25 VSN 002999
   Eq no. 31   Dev name /dev/rmt/0cbn

 rec  pg cd   param code  control   param value
   2    2         00h       74h     0x0           (31:2:002999)
   2    2         01h       74h     0x0           (31:2:002999)
   2    2         02h       74h     0x0           (31:2:002999)
   2    2         03h       74h     0x0           (31:2:002999)
   2    2         04h       74h     0x0           (31:2:002999)
   2    2         05h       74h     0x1400a0      (31:2:002999)
   2    2         06h       74h     0x0           (31:2:002999)

 rec  pg cd   param code  control   param value
   2    3         00h       74h     0x0           (31:3:002999)
   2    3         01h       74h     0x0           (31:3:002999)
   2    3         02h       74h     0x0           (31:3:002999)
   2    3         03h       74h     0x0           (31:3:002999)
   2    3         04h       74h     0x0           (31:3:002999)
   2    3         05h       74h     0x190         (31:3:002999)
   2    3         06h       74h     0x0           (31:3:002999)
 rec  pg cd   param code  control   param value
   2    6         00h       74h     0x0          (31:6:002999)

<<<NOTE:  This output has been truncated for inclusion on this
man page.>>>
```

FILES

/var/opt/SUNWsamfs/sef/sefdata
    The default system error facility log file for SAM-QFS file systems.

SEE ALSO

mcf(4), sefdata(4), sefsysevent(4).

sendtrap(1M)

NAME

sendtrap - SAM-QFS Simple Network Management Protocol (SNMP) trap notification script

SYNOPSIS

```
/etc/opt/SUNWsamfs/scripts/sendtrap
```

AVAILABILITY

SUNWsamfs

SUNWqfs

DESCRIPTION

The sendtrap script publishes SAM-QFS SNMP trap events. It is executed by the syseventd(1M) daemon when it encounters abnormal or exceptional events including tapealert(1M) events. The SNMP version supported is SNMPv2c.

As released, sendtrap is a script that sends a trap to the local host.

The syseventd(1M) daemon executes sendtrap as follows:

o It is invoked with 7 arguments if it is an archiver, stager, releaser, recycler, or file system alert.

o It is invoked with 13 arguments if it is a tapealert(1M) event.

The arguments used are as follows:

Argument	Meaning
1	A keyword identifying the category of the alert (archiver, stager, releaser, recycler, file system, tapeAlert(1M), and so on).
2	The subcategory or specific type of alert. For example, keywords such as CmdErr to express errors

```
                 in the command files, ReadWarning to express tape
                 drive read problems, and so on.)

       3         The error type.  This identifies the severity and
                 syslog level of the event, as follows:

                 Error Type     Values

                 0              Emergency

                 1              Alert

                 2              Critical

                 3              Error

                 4              Warning

       4         The message number as found in the message
                 catalog.  For tapealert(1M) events, this is a
                 concatenation of the Manual type (SSC2/SMC2) and
                 the parameter code as found in the ANSI SCSI-3
                 SSC2 and SMC2 Manuals at www.t10.org.

       5         The system identifier.  That is, the host name of
                 the machine upon which the event originated.

       6         The text  of  the  translated  message string.

       7         The date and time when the event occurred.

       8         The vendor name of the device.  From SCSI INQUIRY.
                 Used only for tapealert(1M) events.

       9         The product identity of the device.  From SCSI
                 INQUIRY.  Used only for tapealert(1M) events.

       10        The revision number of the device.  From SCSI
                 INQUIRY.  Used only for tapealert(1M) events.

       11        The device name.  For example, /dev/rmt/3cbn.
                 Used only for tapealert(1M) events.

       12        The Volume Serial Name (VSN) of the tape.  Used
                 only for tapealert(1M) events.

       13        The probable cause of the tape alert.  Used only
                 for tapealert(1M) events.

   Configuring SNMP
       To enable SNMP reporting, perform the following steps:

       1. Use vi(1) or another editor to open file
          /etc/opt/SUNWsamfs/defaults.conf.

       2. Edit the file so that the alerts=on directive appears.

       3. Save and close the defaults.conf file.
```

```
     4. Issue the samd(1M) config command to reconfigure the
        sam-fsd(1M) daemon.

  Modifying the Trap Destination Host
     By default, traps are sent to port 161 of the localhost.  To
     change the port number or the hostname of the trap
     destination, modify the TRAP_DESTINATION="hostname:port"
     variable in this script.

     This trap destination hostname must be declared in NIS on
     /etc/hosts.

     You can specify that traps be sent to multiple hosts.
     Separate multiple hostname:port specifications with a space
     character.  For example:

     TRAP_DESTINATION="localhost:161 doodle:163 mgmt_station:1162"

  Modifying the SNMP Community String
     To modify the SNMP community string, modify the value of the
     COMMUNITY variable in this script.  By default, the SNMP
     community string is set to public.

SEE ALSO
     sam-fsd(1M), samd(1M), syseventd(1M), tapealert(1M).
```

set_admin(1M)

```
NAME
     set_admin - Sets administrator privileges for Sun QFS and
     SAM-QFS commands

SYNOPSIS
     set_admin [ sam_admin_group ]

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     set_admin changes the group and permissions of many of the
     Sun QFS and SAM-QFS administrator commands so they can be
     executed by users in a selected administrator group. You
     must be logged in as root to execute this command.

OPTIONS
     This command accepts the following argument:

     sam_admin_group
               Specify the administrator group for the Sun QFS
               and SAM-QFS administrator commands.  If you wish
               to change the administrator group back to the
               default, specify bin as the sam_admin_group.  If
               you do not specify a sam_admin_group, you are
               prompted to enter it.

NOTES
```

If you change the administrator group from the default group, bin, and subsequently run the pkgchk(1M) command on the the Sun QFS and SAM-QFS packages, the pkgchk(1M) command issues ERROR messages for the commands modified by set_admin(1M).

You can ignore these messages. The pkgchk(1M) command issues them because it detects that the group name that is associated with the commands is different from what it was at installation time.

SEE ALSO

pkgchk(1M)

set_state(1M)

NAME

set_state - Set device state

SYNOPSIS

```
/opt/SUNWsamfs/sbin/set_state [ -w ] state eq
```

AVAILABILITY

SUNWsamfs

DESCRIPTION

set_state will change the state of a removable media device eq to state. If -w is specified, the command will wait for the operation to complete before terminating. Note: set_state cannot be used to change a file system partition’s allocation state.

The valid states are:

on
The device is usable by Sun QFS or SAM-QFS file systems. A device moving to the on state will be unloaded if there is media mounted.

idle
The device will not be selected for use by either Sun QFS or SAM-QFS
file systems. Any existing activity will be allowed to complete. Once there is no more activity, the device will be placed in the off state.

unavail
The device is unavailable for use by Sun QFS and SAM-QFS file systems and most Sun QFS and SAM-QFS commands. The only valid commands for a device in this state are load(1M), unload(1M), and set_state(1M). A device moving to the unavail state will be unloaded if there is media mounted.

off
The device is unusable by SSun QFS and SAM-QFS file systems. A device moving to the off state from on, idle or unavail will be unloaded if there is media mounted. The only state a down device may be moved

```
                to is off.

FILES
     mcf                 The configuration file for Sun  QFS  and
                         SAM-QFS environments.
SEE ALSO
     load(1M), unload(1M), mcf(4), sam-robotsd(1M)
```

showqueue(1M)

```
NAME
     showqueue - Display content of an archiver queue files

SYNOPSIS
     /opt/SUNWsamfs/sbin/showqueue [-a] [-b] [-d] [-s] [-v]  [-f]
     [-c] [filesystem[ archreq ...]]

     /opt/SUNWsamfs/sbin/showqueue [-b] [-d] [-c] [-v] -q archreq

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     showqueue reads the archreq files named in the argument list
     and prints the information.

     If there are no names in the argument list, the scanlist and
     all ArchReq files are printed for all mounted filesystems.

     If there is only one name in the argument list, the scanlist
     and all ArchReq files are printed for that filesystem.

     Otherwise, print only the listed ArchReq files.

OPTIONS
     -a   ArchReqs.  Print only ArchReqs.

     -b   Print the space in base 10 units. By default, the space
          is printed in base 2 units.

     -c   Use the current working directory as the base for files
          to  display.   Without  this option, showqueue uses the
          standard    location    for    all    archiver     data
          (/var/opt/SUNWsamfs/archiver/fs_name).

     -d   Debug.   Print  ArchReq  structure   fields   with   no
          interpretation.

     -f   Follow. If not using -q, showqueue will  not  terminate
          after  printing  the  requested  queue information, but
          will enter an endless loop, wherein it sleeps for  five
          seconds and then repeats the command.

     -q   Print the ArchReq file archreq.  archreq is the  actual
          name  of the ArchReq.  This option is provided to allow
          the user to examine an ArchReq that is not  under  con-
          trol  of  the archiver.  For instance, when the ArchReq
          is imported from another system.

     -s   Scanlist.  Print only the scanlist.

     -v   Print information about each file to be archived in the
          ArchReq files.

     Example output for:  showqueue -v samfs3

     showqueue -v samfs3
     Filesystem samfs3:
     Files waiting to start:         10
     Files being scheduled:           0
     Files archiving:                 0
     Events processed:              129
         archive             0
         change              3
         close              60
         create             66
         hwm                 0
         modify              0
         rearchive           0
         rename              0
         remove              0
         unarchive           0
         internal            0
     Exam list: 11 entries
      2007-04-11 14:41:12 Archmax
      2007-04-11 14:41:09 Archmax/dir1
      2007-04-11 14:41:10 Archmax/dir2
      2007-04-11 14:39:10 Archmax/dir2/file0
      2007-04-11 14:39:10 Archmax/dir2/file1
      2007-04-11 14:39:10 Archmax/dir2/file2
      2007-04-11 14:39:10 Archmax/dir2/file3
      2007-04-11 14:39:10 Archmax/dir2/file4
      2007-04-11 14:39:12 Archmax/dir5/file7
      2007-04-11 14:39:12 Archmax/dir5/file8
      2007-04-11 14:39:12 Archmax/dir5/file9

     Scan list  Examine: noscan
       0 2007-04-12 00:00:00 background       ---- inodes
     Archive requests
     samfs1.Archmax.1.0 create 2007-04-11 14:39:09
         files:10 space:  10.005M flags:
         Start archive at 2007-04-11 14:40:09 | 10000 files |   10.0G bytes
         type:f ino:1037 s:0/f:0 space:   1.000M time:1176320229 priority:4000
             Archmax/dir1/file0
         type:f ino:1038 s:0/f:0 space:   1.000M time:1176320229 priority:4000
             Archmax/dir1/file1
         type:f ino:1039 s:0/f:0 space:   1.000M time:1176320229 priority:4000
             Archmax/dir1/file2
         type:f ino:1040 s:0/f:0 space:   1.000M time:1176320229 priority:4000
             Archmax/dir1/file3
         type:f ino:1041 s:0/f:0 space:   1.000M time:1176320229 priority:4000
             Archmax/dir1/file4
         type:f ino:1042 s:0/f:0 space:   1.000M time:1176320229 priority:4000
             Archmax/dir1/file5
         type:f ino:1043 s:0/f:0 space:   1.000M time:1176320229 priority:4000
```

```
            Archmax/dir1/file6
        type:f ino:1044 s:0/f:0 space:   1.000M time:1176320229 priority:4000
            Archmax/dir1/file7
        type:f ino:1045 s:0/f:0 space:   1.000M time:1176320229 priority:4000
            Archmax/dir1/file8
        type:f ino:1046 s:0/f:0 space:   1.000M time:1176320229 priority:4000
            Archmax/dir1/file9

    The scanlist shows the following:
     column
     1    Scanlist entry number
     2-3  Time to scan directory
     4    Archive Set if known
     5    Archive copies expected during scan
     6    Scan depth
     7    Directory to scan
     8    If present, start scan at this subdirectory
```

stageall(1M)

NAME
     sam-stagealld - SAM-QFS associative staging daemon

SYNOPSIS
     /opt/SUNWsamfs/sbin/sam-stagealld

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     sam-stagealld is responsible for the associative staging feature. It is initiated by sam-fsd. 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-fsd(1M)

stageback.sh(1M)

NAME
     stageback.sh - Stages files from SAM-QFS archive tapes

SYNOPSIS
     /opt/SUNWsamfs/examples/stageback.sh output_file

AVAILABILITY
     SUNWsamfs

DESCRIPTION

The stageback.sh script stages files from SAM-QFS archive tapes based on archive_audit(1M) output. You can use this script if an archive volume is partially corrupt and there are no other archive copies available.

OPTIONS

This command accepts the following argument:

output_file
: The name of the output file created by the archive_audit(1M) command.

USAGE

The following steps describe how to use the stageback.sh script.

Step 1. Copy the script from its original location in /opt/SUNWsamfs/examples/stageback.sh to the /tmp directory or to a different alternate location. The script itself contains comments to guide you in tailoring the script for your own use.

Step 2. Modify the variables you need. Generally, only the following variables in the script need to be modified:

MEDIA	The 2-character media type of the volume in question as defined on the mcf(4) man page.
VSN	The volume serial name of the volume in question.

For example:

```
eval /opt/SUNWsamfs/bin/rearch -m lt -v TAPE66 $file
```

Step 3. Remove the pound character (#) from column 1 of the line that defines the variables.

Step 4. Run stageback.sh. As its argument, include the name of the output file created by archive_audit(1M).

EXAMPLES

The following script has been edited to contain site-specific information (only the edited portions of the script are shown):

```
    echo rearch  $file
#
#   Edit the following line for the correct media type and VSN
#
    eval /opt/SUNWsamfs/bin/rearch -m lt -v TAPE66 $file
```

WARNINGS

Improper use of this script can damage user or system data. Please refer to the Sun QFS and SAM-QFS Disaster Recovery

```
     Guide or contact technical support before using this script.

FILES
     The stageback.sh script resides in the following location:

     /opt/SUNWsamfs/examples/stageback.sh

SEE ALSO
     stage(1), release(1).

     archive_audit(1M), rearch(1M).
```

star(1M)

```
NAME
     star - Creates tape archives and adds or extract files

SYNOPSIS
     star [options] ... [file] ...

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     This man(1) page describes the GNU version of the tar(1)
     command as extended by Oracle Corporation.  Oracle Corporation
     has enhanced the tar(1) command to support the Sun QFS and
     SAM-QFS file systems.  The star command saves many files
     together into a single tape or disk archive, and it can be
     used to restore individual files from the archive.

OPTIONS
     This command accepts options in both single-character and
     multicharacter equivalent option formats.

Main Operation Mode Options
     -t, --list                    Lists the content of an
                                   archive.
     -x, -extract, -get            Extracts files from an
                                   archive.
     -c, --create                  Creates a new archive.
     -d, --diff, --compare         Finds differences between
                                   archive and file system.
     -r, --append                  Appends files to the end of an
                                   archive.
     -u, --update                  Only appends files newer than
                                   the copy in archive.
     -A, --catenate, --concatenate Appends tar(1) files to an
                                   archive.
     --delete                      Deletes from the archive (not
                                   on mag tapes!).

Operation Modifier Options
     -W, --verify                  Attempts to verify the archive
                                   after writing it.
     --remove-files                Removes files after adding
```

```
                                 them to the archive.
     -k, --keep-old-files          Does not overwrite existing
                                   files when extracting.
     -U, --unlink-first            Removes each file prior to
                                   extracting over it.
     --recursive-unlink            Empties hierarchies prior to
                                   extracting directory.
     -S, --sparse                  Handles sparse files
                                   efficiently.
     -O, --to-stdout               Extracts files to standard
                                   output.
     -G, --incremental             Handles old GNU-format
                                   incremental backup.
     -g, --listed-incremental      Handles new GNU-format
                                   incremental backup.
     --ignore-failed-read          Does not exit with nonzero on
                                   unreadable files.

File Attribute Handling Options
     --owner=name                  Forces name as the owner for
                                   added files.
     --group=name                  Forces name as the group for
                                   added files.
     --mode=changes                Forces (symbolic) mode changes
                                   for added files.
     --atime-preserve              Does not change access times
                                   on dumped files.
     -m, --modification-time       Does not extract file modified
                                   time.
     --same-owner                  Tries extracting files with
                                   the same ownership.
     --numeric-owner               Specifies to always use
                                   numbers for user/group names.
     -p, --same-permissions, --preserve-permissions
                                   Extracts all protection
                                   information.
     -s, --same-order, --preserve-order
                                   Sorts names to extract to
                                   match archive.
     --preserve                    Same as specifying both -p and
                                   -s.

Device Selection and Switching Options
     -f=archive, --file=archive    Uses archive file or device
                                   archive.  The archive can be
                                   file, host:file or
                                   user@host:file.
     --force-local                 Specifies that archive file is
                                   local even if has a colon.
     --rsh-command=command         Specifies to use remote
                                   command instead of rsh.
     -[0-7][lmh]                   Specifies drive and density.
     -M, --multi-volume            Creates/lists/extracts
                                   multivolume archive.
     -L=num, --tape-length=num     Changes tape after writing num
                                   x 1024 bytes.
     -F=file, --info-script=file, --new-volume-script=file
                                   Runs script in file at the end
                                   of each tape (implies -M).
     --volno-file=file             Uses/updates the volume number
                                   in file.

Device Blocking Options
     -b=blocks, --blocking-factor=blocks
                                   Specifies blocks x 512 bytes
                                   per record.
     --record-size=size            Specifies size bytes per
                                   record, multiple of 512.
     -i, --ignore-zeros            Ignores zeroed blocks in
                                   archive (means EOF).
     -B, --read-full-records       Specifies to reblock as the
                                   file is being read (for 4.2BSD
                                   pipes).

Archive Format Selection Options
     -V=name_or_pattern, --label=name_or_pattern
                                   Creates archive with volume
                                   name name or globbing pattern
                                   pattern at list/extract time.
     -o, --old-archive, --portability
                                   Writes a V7 format archive.
     --posix                       Writes a POSIX-conformant
                                   archive (GNU).  Support for
                                   POSIX is only partially
                                   implemented.  The star command
                                   cannot read, nor can it
                                   produce, --posix archives.  If
                                   the POSIXLY_CORRECT
                                   environment variable is set,
                                   GNU extensions are disallowed
                                   with --posix.
     -z, --gzip, --ungzip          Filters the archive through
                                   gzip(1).
     -Z, --compress, --uncompress  Filters the archive through
                                   compress(1).
     --use-compress-program=prog   Filters through prog (must
                                   accept -d).

Local File Selection Options
     -C=dir, --directory=dir       Changes to directory dir.
     -T=name, --files-from=name    Gets names to extract or
                                   create from file name.
     --null                        Instructs star to expect file
                                   names terminated with NUL
                                   characters so star can work
                                   correctly with file names that
                                   contain newline characters.
                                   Must be specified in
                                   conjunction with the -t or the
                                   -files-from=name option.
                                   Disables the -C option.
     --exclude=pattern             Excludes files, given as a
                                   globbing pattern.
     -X=file, --exclude-from=file  Excludes globbing patterns
                                   listed in file.
     -P, --absolute-names          Does not strip leading slash
                                   characters (/) from file
                                   names.
     -h, --dereference             Dumps instead the files to
                                   which symlinks point.
     --no-recursion                Avoids descending
                                   automatically in directories.
     -l, --one-file-system         Stays in local file system
                                   when creating archive.
     -K=name, --starting-file=name Begins at file name in the
                                   archive.
     -n, --newer_than_existing     Only restores files newer than
                                   the existing copy.
     -N=date, --newer=date, --after-date=date
                                   Only restores files newer than
                                   date.
     --newer-mtime                 Compares date and time when
                                   data changed only.
     --backup[=control]            Backs up before removal,
                                   chooses version control.  You
                                   can use the VERSION_CONTROL
                                   environment variable or the
                                   control argument to specify
                                   version control.  The possible
                                   values for control are as
                                   follows:
                                   control Values Version
                                   t, numbered    Makes numbered
                                                  backups.
                                   nil, existing  Makes numbered
                                                  if numbered
                                                  backups exist,
                                                  simple
                                                  otherwise.
                                   never, simple  Specifies to
                                                  always make
                                                  simple backups.
     --suffix=suffix               Backs up before removal.
                                   Overrides usual suffix.  By
                                   default, the backup suffix is
                                   a tilde character (~).  You
                                   can use this option or the
                                   SIMPLE_BACKUP_SUFFIX
                                   environment variable to
                                   specify an alternative suffix.

Informative Output Message Options
     --help                        Writes help text (which is
                                   this man(1) page), then exits.
     --version                     Writes the tar(1) program
                                   version number, then exits.
     -v, --verbose                 Lists files processed
                                   verbosely.
     --checkpoint                  Writes directory names while
                                   reading the archive.
     --totals                      Writes total bytes written
                                   while creating archive.
     -R, --block-number            Shows block number within
                                   archive with each message.
     -w, --interactive, --confirmation
                                   Prompts for confirmation for
                                   every action.
```

```
Input File Option
     file                          The file can be a file or a
                                   device.

NOTES
     The star(1) command defaults to -f- and -b20.

     Be careful when combining options.  The star(1) command
     supports old-style tar combined options without the leading
     "-", e.g.

     /opt/SUNWsamfs/sbin/star tvbf 128 file

     sets the blocksize to 64K and uses "file" as the archive.
     However,

     /opt/SUNWsamfs/sbin/star -tvbf 128 file

     sets the blocksize to "f" and uses "128" as the archive.  If
     you want to use the leading "-" you should separate the
     options as follows:

     /opt/SUNWsamfs/sbin/star -tv -b 128 -f file

SEE ALSO
     For more information about the star(1) command, enter the
     following command:

     /opt/SUNWsamfs/sbin/star --help

     tar(1)
```

tapealert(1M)

```
NAME
     tapealert - Decodes TapeAlert events

SYNOPSIS
     tapealert -i -f /var/opt/SUNWsamfs/devlog/nn

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The TapeAlert feature displays diagnostic and status
     messages for tape drives and automated library devices.
     These messages can provide network administrators with
     critical diagnostic information, such as for media or drive
     failure, when user intervention is urgent and data is at
     risk.  TapeAlert messages also warn you when media or
     devices need servicing, and the messages also provide
     information regarding media or device status.

     The TapeAlert feature enables a tape drive or automated
     library to convey diagnostic information to network
```

administrators. TapeAlerts interpret log sense page 0x2e. The log sense page contains 64 industry-standard error flags. Robots and tape drives support TapeAlert though their own set of specific error flags.

The SAM-QFS software automatically writes TapeAlert events to the device log file, /var/opt/SUNWsamfs/devlog/nn. TapeAlert events are logged in many situations, for example positioning errors, drive self-test errors, and others. If a TapeAlert event is logged, user action is often required.

The tapealert command reads the events logged in the device log file, interprets them, and writes them to a text file for easier viewing. The TapeAlert events can be used to diagnose hardware and media problems for a particular tape volume. In addition, you can enable real-time TapeAlert output to be sent to you in the form of an email or pager message.

Only unique, discrete, nonzero TapeAlert events are written to the device log (devlog/nn). If repeated identical TapeAlert events are detected, only one is written to the device log. This keeps the device log manageable, accurate, and comprehensive without becoming unwieldy. If a TapeAlert event occurs when a drive is empty, no VSN is recorded in the device log or sent with the sysevent. For more information on the device log file and the information written to it, see the devlog(4) man page.

TapeAlert writes device-specific messages to device-specific files. For each device, whether it is an automated library or a tape drive, TapeAlert writes messages specific to that device in the device's own file. Messages are logged as follows:

- For automated libraries, TapeAlerts are accessed at the following events: SAM-QFS device identification, move media, door lock, door unlock, position element, exchange, and after unrecoverable device errors.

- For tape drives, TapeAlerts are accessed at the following events: SAM-QFS device identification, load, unload, and after unrecoverable device errors.

The tapealert command is not supported for magneto optical or mixed-media libraries. TapeAlert is supported on direct-attached hosts only. TapeAlert is not supported on network-attached hosts.

OPTIONS

The tapealert command requires you to specify one of the following options:

-f /var/opt/SUNWsamfs/devlog/nn
: Specifies the file to be read and interpreted. For nn, enter the Equipment Number of the device. The Equipment Number is the second field in the master configuration file (/etc/opt/SUNWsamfs/mcf). Each device has its own

unique devlog/nn file. The system writes each device's TapeAlert events to its own unique file.

For more information on mcf files, see the mcf(4) man page.

-i Reads standard input for interpretation.

For an example of tapealert command output, see the EXAMPLES section of this man page.

USAGE

You can create a TapeAlert sysevent event handler to record all, or only some, automated library and tape drive TapeAlert flags in real time in a single place. The following sections describe the TapeAlert name-value pairs that are needed to build an event handler and describe how to create various types of event handlers.

TapeAlert Sysevent Class and Name-Value Pairs

To create a custom TapeAlert sysevent event handler, the following information is required:

Field	Value
Class	Device
Subclass	TapeAlert
Vendor	SUNW
Publisher	SUNWsamfs

In addition, you can include all or some of the following TapeAlert sysevent name-value pairs:

Name	Value and Data Type
VENDOR	Inquiry vendor. Data type is string.
PRODUCT	Inquiry product. Data type is string.
REV	Inquiry revision. Data type is string.
USN	Inquiry unit serial number. Data type is string.
TOD	Time of day. Data type is int32.
SET	mcf file Family Set. Data type is string.
FSEQ	mcf file Family Set Equipment Number. Data type is int16.
EQ_ORD	mcf file Equipment Number. Data type is int16.
NAME	Device name. Data type is string.

VERSION	Inquiry version. Data type is byte.
INQ_TYPE	Inquiry peripheral device type. Data type is byte.
VSN	Volume serial name. Data type is string.
FLAGS_LEN	TapeAlert flags number. Data type is int16.
FLAGS	TapeAlert flags 64-1. Data type is uint64.

Creating the Event Handler

Creating the event handler is a two-procedure process. In the first procedure, you create the event handler itself.

In the second procedure, you create a notification mechanism.

The following procedure describes how to create the event handler.

1. Log in as root.

2. Create the notification system.

   After the event handler is created, you need to create a notification system. This can be done through your own user-created script or through a C program event handler. The following procedures describe how to create a C program event handler and how to establish email notification.

   To Create a C Program Notifier:

   The following C program, /var/tmp/event_handler.c, writes TapeAlert events to a temporary file:

```
#include <stdio.h>
#include <time.h>
#include <string.h>
#include <stdlib.h>

int main(int argc, char **argv)
{
        char *vendor, *product, *revision, *name, *vsn;
        time_t tod;
        char *todstr;
        short eq_num;
        uchar_t inq_type;
        int flags_len;
        uint64_t flags;
        FILE *fp;

        vendor = argv[1];
        product = argv [2];
```

```
                revision = argv[3];
                tod = (time_t)strtol(argv[4], NULL, 10);
                todstr = asctime(localtime (&tod));
                *(strchr (todstr, '\n')) = '\0';
                eq_num = atoi(argv[5]);
                name = argv[6];
                inq_type = (uchar_t)strtol(argv[7], NULL, 16);
                vsn = argv[8];
                flags_len = atoi(argv[9]);
                flags = (uint64_t)strtoll(argv[10], NULL, 16);

                if ((fp = fopen ("/var/tmp/tapealert", "a+")) == NULL)
                        return 1;
                fprintf (fp, "%s %-8s %-16s %-4s VSN %s\n", todstr, vendor,
                         product, revision, vsn);
                fprintf (fp, "Eq num. %d Dev name %s\n", eq_num, name);
                fprintf (fp, "TapeAlert %d flags %016llx\n", flags_len, flags);
                fprintf (fp, "\n");
                fclose (fp);
                return 0;
        }
```

After this file is created, you must compile it. After compilation, you can run the following commands to load the event handler into the sysevent daemon:

```
# syseventadm add -c Device -s TapeAlert -v SUNW -p SUNWsamfs
/var/tmp/event_handler \"\$VENDOR\" \"\$PRODUCT\" \"\$REV\" \$TOD
\$EQ_ORD \"\$NAME\" \$INQ_TYPE \"\$VSN\" \$FLAGS_LEN \$FLAGS

# syseventadm restart
```

The following commands show the critical clean drive TapeAlert flag 20 active for drive 81 and 82:

```
# tail -f /var/tmp/tapealert
Mon Jun 16 10:42:45 2003 "EXABYTE " "EXB-89008E030203" "V39e" VSN "000166"
Eq num. 81 Dev name "/dev/rmt/1cbn"
TapeAlert 49 flags 0000000000080000

Mon Jun 16 10:42:51 2003 "EXABYTE " "EXB-89008E030203" "V39e" VSN "000165"
Eq num. 82 Dev name "/dev/rmt/0cbn"
TapeAlert 49 flags 0000000000080000
```

To Create an Email Notifier:

The following procedure describes how to enable email notification.

1. Log in as root.

2. In the script file /var/tmp/email_pager, send yourself or your pager a TapeAlert email by adding a line similar to the following:

   ```
   echo $2 | /usr/ucb/mail -s "TapeAlert $1" admin@support.com
   ```

3. Run commands to load the event handler in the sysevent daemon.

Issue the syseventadm(1M) commands, as follows:

```
# syseventadm add -c Device -s TapeAlert -v SUNW -p SUNWsamfs
/var/tmp/email_pager $EQ_ORD "$VSN"
# syseventadm restart
```

EXAMPLES

Example 1. The following mcf file defines one automated library and two tape drives:

```
# OVERLAND NEO Series
/dev/samst/c2t6u0    80    rb   NEO_Series on
/var/opt/SUNWsamfs/catalog/NEO_Series
/dev/rmt/0cbn        81    tp   NEO_Series on
/dev/rmt/1cbn        82    tp   NEO_Series on

historian            90    hy   -         -
/var/opt/SUNWsamfs/catalog/historian
```

You could decode the TapeAlert flags for these devices using the following tapealert commands:

```
# tapealert -f /var/opt/SUNWsam/devlog/80
# tapealert -f /var/opt/SUNWsam/devlog/81
# tapealert -f /var/opt/SUNWsam/devlog/82
```

Example 2. The following examples show tapealert command output:

```
# tapealert -f /var/opt/SUNWsamfs/devlog/91
2003/11/18 15:05:20 Eq no. 91 Seq no. 7
Code: 0x27
Flag: Diagnostics required
Severity: Warning
Application message:
The tape drive may have a hardware fault. Run extended diagnostics
to verity and diagnose the problem. Check the tape drive users
manual for device specific instructions on running extended
diagnostics tests.
Probable cause:
The drive may have a hardware fault that may be identified by
extended diagnostics (i.e. SEND DIAGNOSTIC command).

Code: 0x32
Flag: Lost statistics
Severity: Warning
Application message:
Media statistics have been lost at some time in the past.
Probable cause:
Drive or library powered down with tape loaded.
```

FILES

/etc/sysevent/config/SUNW,sysevent.conf

/var/opt/SUNWsamfs/devlog/nn

SEE ALSO

samd(1M), syseventadm(1M).

devlog(4), mcf(4), sefsysevent(4).

NOTES

The T10 Technical Committee is responsible for SCSI architecture standards. This tapealert command supports the TapeAlert functionality as defined by T10 in the following papers:

- o SCSI Stream Commands - 2 (SSC-2). For a copy of this paper, see www.t10.org/ftp/t10/drafts/ssc2/ssc2r08g.pdf.
- o SCSI Media Changer Commands - 2 (SMC-2). For a copy of this paper, see www.t10.org/ftp/t10/drafts/smc2/smc2r05b.pdf.

The preceding URLs are supported as of June 2003. If you have difficulty accessing these papers, consult the main T10 Technical Committee webpage at www.t10.org.

Portions of this man page were based on or derived from the following T10 Technical Committe publications:

1. SCSI Stream Commands - 2 (SSC-2), Revision 08d, 9 September 2002.
2. SCSI-3 Media Changer Commands - 2 (SMC-2), Revision 5, July 12, 2002.

TapeAlert is limited to direct attached SCSI automated libraries and tape drives that support Log Sense Page 0x2e.

Sun is not responsible for the availability of third-party Web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be reponsible for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

tarback.sh(1M)

NAME

tarback.sh - Reloads files from SAM-QFS archive tapes

SYNOPSIS

/opt/SUNWsamfs/examples/tarback.sh

AVAILABILITY

SUNWsamfs

DESCRIPTION

The tarback.sh script reloads files from SAM-QFS archive tapes. This script can be used if a file system is lost and there are no usable samfsdump(1M) files or copies of the .inodes files available.

USAGE

The following steps describe how to use the tarback.sh script.

Step 1. Use sammkfs(1M) to recreate or restore the file system.

Step 2. Use samu(1M) to set the drive you are using to unavail.

Step 3. Copy the script from its original location in /opt/SUNWsamfs/examples/tarback.sh to the /tmp directory or to a different alternate location. The script itself contains comments to guide you in tailoring the script for your own use.

Step 4. Modify the variables you need. Generally, only the following variables in the script need to be modified:

Variable Name	Content
EQ="eq"	The Equipment Number of the tape drive as defined in the mcf file.
TAPEDRIVE="path"	The raw path to the device described by EQ=.
BLOCKSIZE="size"	The block size in 512-byte units. For example, specify 256 for a block size of 128 kilobytes.
MEDIATYPE="mt"	The 2-character media type for this tape as defined on the mcf(4) man page.
VSN_LIST="vsn1 vsn2 ..."	The list of VSNs to be read. There is no limit on the number of VSNs that can be specified. Use a space character to separate the VSN names. This list can be continued onto another line by using a backslash character (\\). For example: VSN_LIST="vsn1 vsn2 \\ vsn3"

Step 5. Remove the pound character (#) from column 1 of the line that defines the variables.

```
     Step 6.   Run tarback.sh.  There are no arguments.

EXAMPLES
     The following script has been edited to contain site-
     specific information (only the edited portions of the script
     are shown):

     STAR="/opt/SUNWsamfs/sbin/star"
     LOAD="/opt/SUNWsamfs/sbin/load"
     UNLOAD="/opt/SUNWsamfs/sbin/unload"
     EQ=28
     TAPEDRIVE="/dev/rmt/3cbn"
     BLOCKSIZE=256
     MEDIATYPE="lt"

     VSN_LIST="VSNA VSNB VSNC \
        VSNZ"

WARNINGS
     Improper use of this script can damage user or system data.
     Please refer to the Sun QFS and SAM-QFS Disaster Recovery
     Guide or contact technical support before using this script.

FILES
     The tarback.sh script resides in the following location:

     /opt/SUNWsamfs/examples/tarback.sh

SEE ALSO
     samload(1M), samu(1M), star(1M), unload(1M).
```

tplabel(1M)

```
NAME
     tplabel - Label tape

SYNOPSIS
     tplabel -vsn vvvvvv -[new | old vv...]  [-b   blksize]  [-w]
     [-V] [-erase] eq

     tplabel -vsn vvvvvv -[new | old vv...]  [-b   blksize]  [-w]
     [-V] [-erase] eq:slot

DESCRIPTION
     tplabel labels the tape volume specified by eq:slot.  eq  is
     the  equipment number.  If eq is a library, slot is the slot
     in the library containing the tape cartridge.

     The following sequence of labels is written:

          VOL1
          HDR1
          HDR2
          tapemark
          EOF1
          tapemark
          tapemark

     The labels conform to ANSI  X3.27-1987  File  Structure  and
     Labeling of Magnetic Tapes for Information Interchange.

     -vsn vvvvvv specifies the volume serial name  (VSN)  of  the
     tape  being  labeled.  The VSN must be one to six characters
     in length.  All characters in the VSN must be selected  from
     the  26 upper-case letters, the 10 digits, and the following
     special characters: !"%&'()*+,-./:;<=>?_.

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

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

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 nor-
               mally not necessary.  Complete media erasure  will
               take  a long time to perform since all data in the
               media is erased.

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

tpverify(1M)

```
NAME
     tpverify - Tape Verify

SYNOPSIS
     /opt/SUNWsamfs/sbin/tpverify    [    -a    ]    [    -w    ]
     eq:slot[:partition] [ deq ]

     /opt/SUNWsamfs/sbin/tpverify [ -a ] [ -w ]  mediatype.vsn  [
     deq ]
     /opt/SUNWsamfs/sbin/tpverify    [    -c    ]    [    -w    ]
     eq:slot[:partition] [ deq ]

     /opt/SUNWsamfs/sbin/tpverify [ -c ] [ -w ]  mediatype.vsn  [
     deq ]
```

AVAILABILITY
SUNWsamfs

DESCRIPTION
tpverify requests that the volume specified by eq:slot[:partition] or mediatype.vsn be loaded into device deq and verified. The device specified by deq must be a tape drive. If deq is not specified, then the volume is loaded into an available drive in the media changer eq. The SAM-QFS file system chooses the drive into which the volume is loaded. If tpverify is canceled then the last position verified is saved in the robot catalog and is used as the starting position of the next tpverify command. The "all" option starts a verify operation from the beginning of tape. A verify operation run in a tape drive in the "on" state can be canceled by the SAM-QFS archiver or stager if the vsn is needed. A verify operation run in a tape drive in the "una- vail" state can not be canceled by SAM. The itemize command with the -2 option is used to display a tape’s last verified time and last verified position.

OPTIONS

-a Override the last verified position saved in the robot catalog to start the verify operation from the first archive file on media.

-c Cancels a running tpverify command. Use the same arguments used to run the tpverify command along with the -c option.

-w Wait for the operation to complete before ter- minating.

RETURN VALUES
One of the following values is returned by the program:

0 Successful verify.

255 Verify failed.

254 User canceled verify.

253 SAM-FS canceled verify.

252 Drive needs cleaning.

251 Verify DIV error set bad media.

250 Verify set the drive to down.

249 Verify media error.

FILES

mcf	The configuration file for SAM-QFS environments

SEE ALSO
itemize(1M), sam-robotsd(1M), mcf(4).

trace_rotate(1M)

NAME
trace_rotate - Rotates trace files

SYNOPSIS
trace_rotate trace_file

AVAILABILITY
SUNWsamfs

DESCRIPTION
The trace_rotate script rotates trace files generated by Sun QFS or SAM-QFS daemons. It is executed by sam-fsd when a daemon trace file has aged or grown beyond parameters specified in the defaults.conf file.

The process of rotating trace files assumes that you want to keep no more than seven generations of a trace file in your directories at one time. When the trace files are rotated, the newest trace file is renamed trace_file.1, the next-newest trace file is renamed trace_file.2, and so on. The oldest trace file in the directory is deleted as new ones are added, so the oldest trace file in the directory at any time is always called trace_file.7. This process provides two benefits:

o A given trace file never becomes so large that it is unwieldy to copy or view.

o Entries are expired after a period of time. This prevents file systems from filling up due to the volume of trace entries.

OPTIONS
This command accepts the following arguments:

trace_file
The full path name of the trace file.

EXAMPLES
By default, trace files are rotated by the sam-fsd daemon according to parameters specified in the /etc/opt/SUNWsamfs/defaults.conf file. If no parameters are specified, the trace files are rotated when their size reaches 10 megabytes. The current parameters can be displayed by executing the /opt/SUNWsamfs/sbin/sam-fsd command.

You could use the following command line to invoke the

```
     script manually:

     # trace_rotate /var/opt/SUNWsamfs/trace/sam-archiverd

     You can enable this script's trace file rotation mechanism
     automatically in one of the following ways:

     Method 1. By using the daemon_name.age=age or the
               daemon_name.size=size directive in the
               defaults.conf(4) file.  For more information, see
               the defaults.conf(4) man page.

     Method 2. By setting up a crontab(1) entry to run the
               trace_rotate script.  The following crontab(1)
               entry maintains eight back-up files, sam-
               archiverd.0 through sam-archiverd.7, plus the
               original:

     10 3 * * 0  /opt/SUNWsamfs/sbin/trace_rotate /var/opt/SUNWsamfs/trace/sam-archiverd

SEE ALSO
     crontab(1).

     sam-fsd(1M).

     defaults.conf(4).
```

umount_samfs(1M)

```
NAME
     umount_samfs - Unmounts a Sun QFS or SAM-QFS file system

SYNOPSIS
     umount -F samfs [-f] [generic_options] [-o await_clients=n]
     special | mount_point

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The umount command unmounts a currently mounted file system
     from the file system hierarchy.  The file system may be
     specified by either its mount point or its special (also
     known as its family set name).

     For more information on the mount(1M) command, see the
     mount(1M) man page and the mount_samfs(1M) man page.

     For more information on the umount command, see the
     umount(1M) man page.

OPTIONS
     -F samfs  Specifies that the file system being unmounted is
               of type samfs.  Both Sun QFS and SAM-QFS file
               systems are of type samfs.
     -f        Forcibly unmount the file system, i.e., unmount
               the file system even if it is busy.  This may fail
               or hang in some situations, particularly on
               clients if the metadata server does not have the
               FS mounted.

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

     -o await_clients=n
               If the mounted file system is a Sun QFS or SAM-QFS
               shared file system and the current host is the
               metadata server for that file system, the umount
               command will wait for the specified period (n
               seconds) for any mounted clients to first unmount.
               The unmount command proceeds after either the last
               client host unmounts the file system, or the
               waiting period expires.

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

     mount_point
               The path name or directory at which the file
               system is mounted.  If the mount_point had any
               contents prior to the mount operation, these
               become accessible after the umount command
               successfully completes.

EXAMPLES
     # umount samfs1

     Unmount the file system whose family set name is samfs1.  If
     the file system is in use, the command will fail.

     # umount -f -o await_clients=30 /qfs1

     Forcibly unmount the file system mounted on /qfs1.  If the
     file system is a shared file system, and the local host is
     the metadata server for that file system, then umount will
     wait up to 30 seconds for the clients to unmount before
     issuing the unmount.  If the file system is not shared, or
     has no mounted clients, or the local host is not the
     metadata server, the await_clients option has no effect.
     The file system is forcibly unmounted.

FILES
     /etc/mnttab         Table of mounted file systems.

SEE ALSO
     release(1).

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

     mount(2).  umount(2).
     mcf(4), mnttab(4),
```

unarchive(1M)

```
NAME
     unarchive - Deletes archive entries

SYNOPSIS
     unarchive -c copy_no [-f] [-m media_type [-v vsn]] [-M] [-o]
     filename . . .

     unarchive [-c copy_no] [-f] -m media_type [-v vsn] [-M] [-o]
     filename . . .

     unarchive -c copy_no [-f] [-m media_type [-v vsn]] [-M] [-o]
     -r dirname [filename] . . .

     unarchive [-c copy_no] [-f] -m media_type [-v vsn] [-M] [-o]
     -r dirname [filename] . . .

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The unarchive command deletes archive entries for one or
     more files or directories.  The specifications for the
     archive copy (-c copy_no) and/or the media type and VSN
     (-m media_type [-v vsn]) determine which archive copy is
     deleted.

     There are several ways to specify one or more archive
     entries to be unarchived.  These ways are as follows:

     o  By copy number

     o  By copy number, media type, and VSN

     o  By copy number and media type

     o  By media type

     o  By media type and VSN

OPTIONS
     This command accepts the following options:

     -c copy_no
               Deletes the specified archive copy number.  If one
               or more -c options are are specified, only those
               archive copies (copies 1, 2, 3, or 4) are deleted.
               Specify 1, 2, 3, or 4 for copy_no.  Either a -c or
               a -m option must be specified.

     -f        Suppresses errors.
     -m media_type
               Deletes all archive copies from the specified
               media_type.  For the list of possible media_type
               specifications, see the mcf(4) man page.  Either a
               -c or a -m option must be specified.  If you
               specify a -m option, you can also specify a -v
               option.

     -M        Unarchives metadata only.  This includes
               directories, the segment index, and removable
               media files.  Regular files and symbolic links are
               not unarchived.  If you are unarchiving a
               directory, you must specify the -M option.

     -o        Specifies that the file must be online before its
               archive entry is deleted.  If the file is offline,
               the unarchive command stages the file to disk
               before deleting any entries.

     -r dirname
               Recursively deletes the archive entries of dirname
               and its subdirectories.  The archive entries of
               files in the directories and subdirectories are
               deleted.

     -v vsn    Deletes the archive copies on vsn.  For vsn,
               specify a volume serial name (VSN).  If you
               specify a -v option, you must also specify a -m
               option.

     filename  Deletes the archive entries for the specified
               filename.

NOTES
     If the last (undamaged) copy of a file would be unarchived,
     the unarchive command reports Last undamaged offline copy
     and does not unarchive that copy.

SEE ALSO
     mcf(4).
```

undamage(1M)

```
NAME
     undamage - Marks archive entries as undamaged and unstaled

SYNOPSIS
     /opt/SUNWsamfs/sbin/undamage -c copy_no [-f] [-m media_type
     [-v vsn]] [-M] filename ...

     /opt/SUNWsamfs/sbin/undamage [-c copy_no] [-f] -m media_type
     [-v vsn] [-M] filename ...

     /opt/SUNWsamfs/sbin/undamage -c copy_no [-f] [-m media_type
     [-v vsn]] [-M] -r dirname ...  filename ...
     /opt/SUNWsamfs/sbin/undamage [-c copy_no] [-f] -m media_type
     [-v vsn] [-M] -r dirname ...  filename ...

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The undamage command marks archive entries for one or more
     files or directories as undamaged and not stale based on the
     archive copy number and/or the media type and VSN specified.
     The undamage command also marks the file(s) themselves as
     undamaged.

     There are several ways to mark one or more archive entries
     as undamaged.  These ways are as follows:

     o  By copy number

     o  By copy number, media type, and VSN

     o  By copy number and media type

     o  By media type

     o  By media type and VSN

OPTIONS
     This command accepts the following options:

     -c copy_no
          Marks the specified archive copy number as undamaged.
          If one or more -c options are specified, only those
          archive copies (copies 1, 2, 3, or 4) are marked as
          undamaged.  Specify 1, 2, 3, or 4 for copy_no.  Either
          a -c or a -m option must be specified.

     -f   Suppresses errors.

     -m media_type
          Marks all copies from the specified media_type as
          undamaged.  For the list of possible media_type
          specifications, see the mcf(4) man page.  Either a -c
          or a -m option must be specified.  If you specify a -m
          option, you can also specify a -v option.

     -M   Marks only metadata as undamaged.  This includes
          directories, the segment index, and removable-media
          files.  Regular files are not marked as undamaged.  If
          you are marking a directory as undamaged, you must
          specify the -M option.

     -r dirname ...
          Recursively marks one or more specified dirnames and
          subdirectories as undamaged.  The archive entries of
          files in the directories and subdirectories are marked
          as undamaged.

     -v vsn
          Marks the archive copies on vsn as undamaged.  For vsn,
          specify a volume serial name (VSN).  If you specify a
          -v option, you must also specify a -m option.

     filename ...
          Marks the archive entries for one or more specified
          filename arguments as undamaged.

EXAMPLE
     The following command marks all archive copies of myfile as
     undamaged:

     # undamage -c1 -c2 -c3 -c4 myfile

SEE ALSO
     mcf(4).
```

unload(1M)

```
NAME
     unload - Unload media from a device

SYNOPSIS
     /opt/SUNWsamfs/sbin/unload [ -w ] eq

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     Unload the media mounted on device eq.  The device specified
     by eq must be a removable media device or a media changer.

     If eq is a removable media  device  controlled  by  a  media
     changer, the medium will be moved into storage. This command
     is used when a shutdown of a SAM-QFS file system is required
     and a tape is still in a drive. This command is also used in
     situations where the system administrator wishes to remove a
     tape from a drive that is currently in the unavail state.

     If eq is a media changer, unload moves catalog entries  from
     the  media changer’s catalog to the historian’s catalog. The
     device state for device eq is set to off.  When  the  device
     state  for  the  media  changer  is  set to on and the media
     changer has bar codes, then the catalog information for that
     media  changer is retrieved from the historian. If the media
     changer does not have bar codes, an  audit  invoked  by  the
     administrator  will  recover the historian information. This
     command is useful for moving tapes in to and  out  of  media
     changers  which  do  not have import/export capabilities, or
     sense capability for open door. By first issuing the  unload
     command,  the  system administrator can safely open the door
     to the media changer, add or remove tapes, close  the  door,
     and re-audit the media changer.

     If -w is specified, the command will wait for the  operation
     to complete before terminating.
```

```
FILES
     mcf                 The  configuration  file   for   SAM-QFS
                         environments

SEE ALSO
     auditslot(1M),   historian(7),   load(1M),    set_state(1M),
     mcf(4), sam-robotsd(1M)
```

unrearch(1M)

```
NAME
     unrearch - Removes a specification to rearchive a file

SYNOPSIS
     unrearch -c copy_no [-f] [-m media_type [-v vsn]] [-M]
     filename . . .

     unrearch [-c copy_no] [-f] -m media_type [-v vsn] [-M]
     filename . . .

     unrearch -c copy_no [-f] [-m media_type [-v vsn]] [-M]
     -r dirname [filename] . . .

     unrearch [-c copy_no] [-f] -m media_type [-v vsn] [-M]
     -r dirname [filename] . . .

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The unrearch command lets you remove a request to rearchive
     a file or a directory.  For example, if you have used the
     rearch(1M) command to request that a file be rearchived, you
     can use the unrearch command to clear the bit that the
     rearch(1M) command had set.  The specifications for the
     archive copy (-c copy_no) and/or the media type and VSN
     (-m media_type [-v vsn]) determine which archive copy is
     affected.

     There are several ways to remove the request to rearchive
     from one or more archive entries.  These ways are as
     follows:

     o  By copy number

     o  By copy number, media type, and VSN

     o  By copy number and media type

     o  By media type

     o  By media type and VSN

OPTIONS
     This command accepts the following options:
```

```
     -c copy_no
               Removes the rearchive request for copy_no.
               Specify 1, 2, 3, or 4 for copy_no.  If one or more
               -c options are are specified, the command removes
               the rearchive request from only those archive
               copies (copies 1, 2, 3, or 4).  Either a -c or a
               -m option must be specified.

     -f        Suppresses errors.

     -m media_type
               Removes rearchive requests from all archive copies
               on the specified media_type.  For the list of
               possible media_type specifications, see the mcf(4)
               man page.  Either a -c or a -m option must be
               specified.  If you specify a -m option, you can
               also specify a -v option.

     -M        Removes rearchive requests for metadata only.
               This includes directories, the segment index, and
               removable media files.  Regular files and symbolic
               links are not unrearchived.  If you are
               unarchiving a directory, you must specify the -M
               option.

     -r dirname
               Recursively removes the rearchive requests for the
               entries of dirname and its subdirectories.
               Removes the archive requests of files in the
               directories and subdirectories.

     -v vsn    Removes the rearchive requests for the archive
               copies on vsn.  For vsn, specify a volume serial
               name (VSN).  If you specify a -v option, you must
               also specify a -m option.

     filename  Removes the rearchive requests for the specified
               filename.

SEE ALSO
     mcf(4), rearch(1M).
```

unreserve(1M)

```
NAME
     unreserve - Unreserve a volume for archiving.

SYNOPSIS
     /opt/SUNWsamfs/sbin/unreserve mediatype.vsn
     /opt/SUNWsamfs/sbin/unreserve eq:slot[:partition]

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     unreserve removes the assignment of the volume for  archival
```

of specific files.

Normally, relabeling a volume will remove the reservation of a volumes. This command is provided to unreserve a volume without re-labeling.

The volume is determined by the specifier mediatype.vsn , or eq:slot[:partition]

SEE ALSO

archiver(1M), archiver.cmd(1M), reserve(1M)

CHAPTER 3

Library Functions (Man Pages Section 3)

This chapter provides the section 3 man pages for Sun QFS and Sun Storage Archive Manager.

intro_libsam(3)

```
NAME
     intro_libsam, intro_libsamrpc - Introduces the Sun QFS and
     and SAM-QFS Application Programmer Interface (API) routines

AVAILABILITY
     SUNWqfs

     SUNWsamfs

DESCRIPTION
     The Sun QFS and SAM-QFS API allows a Sun QFS or SAM-QFS file
     to be requested from within an application program.  The
     aplication program can reside either on the machine upon
     which the Sun QFS or SAM-QFS file system is running or on
     another machine on the network.  This man page provides an
     introduction to the API routines.

     The following topics are presented:

     o API overview

     o API library routines

     o Using libsam

     o Using libsamrpc

API OVERVIEW
     When a request is made, the process or program making the
     request is the client process or program, running on the
     client machine.  The requests are received and processed by
     the server, running on the server, or host, machine.  For
     the API routines, the server machine is always the machine
     upon which the Sun QFS or SAM-QFS file system is running.
```

In the simplest case, the client and server machines are the same, and no network communication is necessary. In other cases, however, the application programmer needs to allow for the client program to run on a machine where the Sun QFS or SAM-QFS file system is not running. In this case, networked library calls from libsamrpc must be used.

The two API libraries available with the Sun QFS and SAM-QFS file systems are as follows:

o libsam. The library calls in libsam do not perform network communication. They only make local requests. In this case, each library call makes a system call, and the server is the local operating system.

o libsamrpc. The library calls in libsamrpc use Remote Procedure Calls (RPCs) to communicate with a special server process, sam-rpcd. Because of the RPC mechanism, the client and server can exist on the same machine or on different machines in the network. The server process always runs on the machine upon which the Sun QFS or SAM-QFS file system is running.

Both libsam and libsamrpc are released in shared object (.so) and archive (.a) format for Solaris platforms. libsam.so and libsam.a are installed in /opt/SUNWsamfs/lib. libsamrpc.so and libsamrpc.a are installed in /opt/SUNWsamfs/client/lib, with symbolic links to them in /opt/SUNWsamfs/lib.

API LIBRARY ROUTINES

The library calls for the Sun QFS and SAM-QFS software are supported in libsam, and a subset is supported in libsamrpc.

Table 1 lists the API library routines and indicates the environments in which they are supported. In addition, table 1 indicates the libraries in which they are included:

Table 1. Library routine availability

Routine	Description
sam_advise	Sets file attributes. Availability: Sun QFS and SAM-QFS environments. Libraries: libsam.
sam_archive	Sets archive attributes on a file. Availability: SAM-QFS environments. Libraries: libsam and libsamrpc.
sam_rearchive	Sets rearchive attributes on a file. Availability: SAM-QFS environments. Libraries: libsam.
sam_exarchive	Exchanges archive copies of a file or directory.

```
                  Availability:  SAM-QFS environments.
                  Libraries:  libsam.

   sam_unarchive  Removes archive copies for a file or
                  directory.
                  Availability:  SAM-QFS environments.
                  Libraries:  libsam.

   sam_unrearch   Removes rearchive attributes on a file or
                  directory.
                  Availability:  SAM-QFS environments.
                  Libraries:  libsam.

   sam_damage     Sets damaged attribute on a file or
                  directory.
                  Availability:  SAM-QFS environments.
                  Libraries:  libsam.

   sam_undamage   Clears damaged and stale status of a file or
                  directory.
                  Availability:  SAM-QFS environments.
                  Libraries:  libsam.

   sam_cancelstage
                  Cancels a pending or in-progress stage on a
                  file.
                  Availability:  SAM-QFS environments.
                  Libraries:  libsam.

   sam_closecat   Ends access to the catalog for an automated
                  library.
                  Availability:  SAM-QFS environments.
                  Libraries:  libsam.

   sam_closerpc   Closes down the RPC connection.
                  Availability:  SAM-QFS environments.
                  Libraries:  libsamrpc.

   sam_devstat, sam_ndevstat
                  Gets device status.  sam_ndevstat accepts a
                  longer device name.
                  Availability:  SAM-QFS environments.
                  Libraries:  libsam.

   sam_devstr     Translates numeric device status into a
                  character string.
                  Availability:  SAM-QFS environments.
                  Libraries:  libsam.

   sam_getcatalog Obtains a range of entries from the catalog
                  for an automated library.
                  Availability:  SAM-QFS environments.
                  Libraries:  libsam.

   sam_initrpc    Initializes the RPC connection.
                  Availability:  SAM-QFS environments.
                  Libraries:  libsamrpc.

   sam_opencat    Accesses the VSN catalog for an automated
```

```
                              library.
                              Availability:  SAM-QFS environments.
                              Libraries:  libsam.

               sam_readrminfo Gets information for a removable media file.
                              Availability:  SAM-QFS environments.

                              Libraries:  libsam.

               sam_release    Releases and sets release attributes on a
                              file.
                              Availability:  SAM-QFS environments.
                              Libraries:  libsam and libsamrpc.

               sam_request    Creates a removable media file.
                              Availability:  SAM-QFS environments.
                              Libraries:  libsam.

               sam_restore_copy
                              Creates an archive copy for a file.
                              Availability:  SAM-QFS environments.
                              Libraries:  libsam.

               sam_restore_file
                              Creates an offline file.
                              Availability:  SAM-QFS environments.
                              Libraries:  libsam.

               sam_segment    Sets segment attributes on a file or
                              directory.
                              Availability:  SAM-QFS environments.
                              Libraries:  libsam and libsamrpc.

               sam_segment_stat
                              Obtains file information and follows symbolic
                              links to a segmented file.
                              Availability:  SAM-QFS environments.
                              Libraries:  libsam.

               sam_setfa      Sets file attributes.
                              Availability:  Sun QFS and SAM-QFS
                              environments.
                              Libraries:  libsam and libsamrpc.

               sam_ssum       Sets checksum attributes on a file.
                              Availability:  SAM-QFS environments.
                              Libraries:  libsam.

               sam_stage      Stages and sets stage attributes on a file.
                              Availability:  SAM-QFS environments.
                              Libraries:  libsam and libsamrpc.

               sam_stat, sam_lstat
                              sam_stat obtains file information and follows
                              symbolic links to the file.  sam_lstat
                              obtains file information, and if that file is
                              a link, it returns information about the
                              link.
                              Availability:  Sun QFS and SAM-QFS
```

```
                   environments.
                   Libraries:  libsam and libsamrpc.

     sam_vsn_stat, sam_segment_vsn_stat
                   Obtain VSN status for a file or a file's data
                   segment that overflows VSNs.
                   Availability:  SAM-QFS environments.
                   Libraries:  libsam.

     All APIs in libsam, except for sam_closecat, sam_getcatalog,
     and sam_opencat, are available for use with 64-bit programs.
     Oracle Corporation, Inc. does not support a 64-bit version of
     libsamrpc.

     For more details about each library routine, see the
     individual corresponding man page for that routine.  Library
     routines contained in libsam are found in section 3 of the
     online man pages.  Library routines contained in libsamrpc
     are found in section 3X of the online man pages.

USING libsam
     No special initialization or configuration is required prior
     to using the API library routines in libsam.  The
     application program must be linked with libsam, however.
     For information on the routines, see the individual libsam
     man pages, all of which are listed in the SEE ALSO section
     of this man page.

USING libsamrpc
     The source code for libsamrpc is included in the release for
     customers who wish to write and run application programs on
     platforms that do not run the Solaris operating system.  In
     these cases, the library must be ported to the client
     machine.  The source code is located in
     /opt/SUNWsamfs/client/src.  Example application programs are
     located in /opt/SUNWsamfs/client/examples.

  Specifying the Server Machine
     A call to sam_initrpc is required before any other RPC
     client API calls can be executed successfully.  Only one
     sam_initrpc call is required, followed by any number of
     other client API calls (other than sam_closerpc).  The
     sam_initrpc call accepts one argument:  a pointer to a
     character string that specifies the name of the server
     machine.  If this pointer is NULL, sam_initrpc checks for an
     environment variable named SAMHOST.  If this environment
     variable is set, that name is used for the server machine.
     If there is no SAMHOST environment variable, the default
     server name samhost is used.

     In summary, the name of the server machine can be specified
     in any of three ways, which are checked by sam_initrpc in

     the following order:

     1. As an argument to the sam_initrpc call.

     2. As the environment variable SAMHOST.
```

3. By accepting the default server name, samhost.

RPC Server Process

The RPC API server process receives and processes requests from the client. This server process, /opt/SUNWsamfs/sbin/sam-rpcd, must be run on the same machine as the file system. The sam-rpcd daemon must be running for client requests to execute successfully.

The sam-rpcd daemon is started automatically by sam-amld if the appropriate entry is made in the defaults.conf file. For information on editing the defaults.conf file, see Configuring the API later in this man page.

The sam-rpcd daemon can also be started manually. It should be run as superuser. The sam-rpcd command accepts no arguments.

The sam-rpcd daemon services the requests it receives by making the appropriate system call on the server machine and then returning the output or result to the client. For more information on this daemon, see the sam-rpcd(1M) man page.

Configuring the API

The following steps describe setting up the API server and clients. These steps assume that your software is properly configured and running.

Step 1: Configure the API Server

For the server portion of the API to run successfully, the following conditions must be present:

o The RPC program name and number pair must be known on the server machine

o The RPC program name and number pair must be the same as the pair used on the API client machines.

Make an entry for the RPC program name and number. The RPC program number is a number chosen by you. The RPC program name is samfs. The name and number pair must be the same on the server and all clients. The /etc/nsswitch.conf file determines where you should specify the RPC program name and number pair. For more information on this, see the nsswitch.conf(4) man page.

In /etc/rpc (or the NIS database), add the following line:

```
samfs          150005
```

In /etc/services (or the NIS database), add the following line:

```
samfs          5012/tcp  # SAM-QFS API
```

The API server is started automatically by the sam-amld daemon if the following entry is made in the defaults.conf file (note that changes to the defaults.conf file do not take effect until the next time the sam-amld daemon is initialized):

```
samrpc = on
```

The sam-rpcd daemon is not automatically started if no entry for it appears in the defaults.conf file or if the following entry appears in the file:

```
samrpc = off
```

For more information about the defaults.conf file, see the defaults.conf(4) man page.

Step 2: Configure the API Client Machines

The following two configuration components must be present on the client machine for the RPC communication to be successful:

o The name of the server machine.

o The RPC program name and number pair.

Make an entry for the RPC program name and number on all client machines, as you did on the API server machine previously. Again, the RPC program name must be samfs. The RPC program number is a number chosen by you, but it must be the same on the server and client machines.

In /etc/rpc (or the NIS database), add the following line:

```
samfs          150005
```

The host name of the server machine must be known on the client machine. For default cases, the host name samhost must be listed as an alias for the SAM-QFS file system server machine. For more information, see the sam_initrpc(3X) man page.

Authentication and libsamrpc

Authentication information is generated at the time of the sam-initrpc call. This information consists of the user identification (uid) and group identification (gid) of the calling process. It is associated with the connection made to the RPC server process.

Subsequent libsamrpc calls have this information associated. When the request is received by the RPC server process on the server machine, the uid and gid information is used. File access and operations are granted or denied based on this information.

It is important that the server machine have a common uid and gid space with the client machines.

SEE ALSO

sam_advise(3), sam_archive(3), sam_rearch(3),

sam_exarchive(3), sam_unarchive(3), sam_unrearch(3),
sam_damage(3), sam_undamage(3), sam_cancelstage(3),
sam_closecat(3), sam_devstat(3), sam_devstr(3),
sam_getcatalog(3), sam_lstat(3), sam_ndevstat(3),
sam_opencat(3), sam_readrminfo(3), sam_release(3),
sam_request(3), sam_restore_copy(3), sam_restore_file(3),
sam_segment(3), sam_setfa(3), sam_ssum(3), sam_stage(3),
sam_stat(3).

sam_archive(3X), sam_closerpc(3X), sam_initrpc(3X),
sam_lstat(3X), sam_release(3X), sam_stage(3X), sam_stat(3X).

intro_libsamrpc(3)

NAME
intro_libsam, intro_libsamrpc - Introduces the Sun QFS and and SAM-QFS Application Programmer Interface (API) routines

AVAILABILITY
SUNWqfs

SUNWsamfs

DESCRIPTION
The Sun QFS and SAM-QFS API allows a Sun QFS or SAM-QFS file to be requested from within an application program. The aplication program can reside either on the machine upon which the Sun QFS or SAM-QFS file system is running or on another machine on the network. This man page provides an introduction to the API routines.

The following topics are presented:

o API overview

o API library routines

o Using libsam

o Using libsamrpc

API OVERVIEW
When a request is made, the process or program making the request is the client process or program, running on the client machine. The requests are received and processed by the server, running on the server, or host, machine. For the API routines, the server machine is always the machine upon which the Sun QFS or SAM-QFS file system is running.

In the simplest case, the client and server machines are the same, and no network communication is necessary. In other cases, however, the application programmer needs to allow for the client program to run on a machine where the Sun QFS or SAM-QFS file system is not running. In this case, networked library calls from libsamrpc must be used.

The two API libraries available with the Sun QFS and SAM-QFS file systems are as follows:

- o libsam. The library calls in libsam do not perform network communication. They only make local requests. In this case, each library call makes a system call, and the server is the local operating system.

- o libsamrpc. The library calls in libsamrpc use Remote Procedure Calls (RPCs) to communicate with a special

  server process, sam-rpcd. Because of the RPC mechanism, the client and server can exist on the same machine or on different machines in the network. The server process always runs on the machine upon which the Sun QFS or SAM-QFS file system is running.

Both libsam and libsamrpc are released in shared object (.so) and archive (.a) format for Solaris platforms. libsam.so and libsam.a are installed in /opt/SUNWsamfs/lib. libsamrpc.so and libsamrpc.a are installed in /opt/SUNWsamfs/client/lib, with symbolic links to them in /opt/SUNWsamfs/lib.

API LIBRARY ROUTINES

The library calls for the Sun QFS and SAM-QFS software are supported in libsam, and a subset is supported in libsamrpc.

Table 1 lists the API library routines and indicates the environments in which they are supported. In addition, table 1 indicates the libraries in which they are included:

Table 1. Library routine availability

Routine	Description
sam_advise	Sets file attributes. Availability: Sun QFS and SAM-QFS environments. Libraries: libsam.
sam_archive	Sets archive attributes on a file. Availability: SAM-QFS environments. Libraries: libsam and libsamrpc.
sam_rearchive	Sets rearchive attributes on a file. Availability: SAM-QFS environments. Libraries: libsam.
sam_exarchive	Exchanges archive copies of a file or directory. Availability: SAM-QFS environments. Libraries: libsam.
sam_unarchive	Removes archive copies for a file or directory. Availability: SAM-QFS environments. Libraries: libsam.

```
sam_unrearch   Removes rearchive attributes on a file or
               directory.
               Availability:  SAM-QFS environments.
               Libraries:  libsam.

sam_damage     Sets damaged attribute on a file or
               directory.
               Availability:  SAM-QFS environments.
               Libraries:  libsam.

sam_undamage   Clears damaged and stale status of a file or
               directory.
               Availability:  SAM-QFS environments.
               Libraries:  libsam.

sam_cancelstage
               Cancels a pending or in-progress stage on a
               file.
               Availability:  SAM-QFS environments.
               Libraries:  libsam.

sam_closecat   Ends access to the catalog for an automated
               library.
               Availability:  SAM-QFS environments.
               Libraries:  libsam.

sam_closerpc   Closes down the RPC connection.
               Availability:  SAM-QFS environments.
               Libraries:  libsamrpc.

sam_devstat, sam_ndevstat
               Gets device status.  sam_ndevstat accepts a
               longer device name.
               Availability:  SAM-QFS environments.
               Libraries:  libsam.

sam_devstr     Translates numeric device status into a
               character string.
               Availability:  SAM-QFS environments.
               Libraries:  libsam.

sam_getcatalog Obtains a range of entries from the catalog
               for an automated library.
               Availability:  SAM-QFS environments.
               Libraries:  libsam.

sam_initrpc    Initializes the RPC connection.
               Availability:  SAM-QFS environments.
               Libraries:  libsamrpc.

sam_opencat    Accesses the VSN catalog for an automated
               library.
               Availability:  SAM-QFS environments.
               Libraries:  libsam.

sam_readrminfo Gets information for a removable media file.
               Availability:  SAM-QFS environments.

               Libraries:  libsam.
```

sam_release	Releases and sets release attributes on a file. Availability: SAM-QFS environments. Libraries: libsam and libsamrpc.
sam_request	Creates a removable media file. Availability: SAM-QFS environments. Libraries: libsam.
sam_restore_copy	Creates an archive copy for a file. Availability: SAM-QFS environments. Libraries: libsam.
sam_restore_file	Creates an offline file. Availability: SAM-QFS environments. Libraries: libsam.
sam_segment	Sets segment attributes on a file or directory. Availability: SAM-QFS environments. Libraries: libsam and libsamrpc.
sam_segment_stat	Obtains file information and follows symbolic links to a segmented file. Availability: SAM-QFS environments. Libraries: libsam.
sam_setfa	Sets file attributes. Availability: Sun QFS and SAM-QFS environments. Libraries: libsam and libsamrpc.
sam_ssum	Sets checksum attributes on a file. Availability: SAM-QFS environments. Libraries: libsam.
sam_stage	Stages and sets stage attributes on a file. Availability: SAM-QFS environments. Libraries: libsam and libsamrpc.
sam_stat, sam_lstat	sam_stat obtains file information and follows symbolic links to the file. sam_lstat obtains file information, and if that file is a link, it returns information about the link. Availability: Sun QFS and SAM-QFS environments. Libraries: libsam and libsamrpc.
sam_vsn_stat, sam_segment_vsn_stat	Obtain VSN status for a file or a file's data segment that overflows VSNs. Availability: SAM-QFS environments.

Libraries: libsam.

All APIs in libsam, except for sam_closecat, sam_getcatalog, and sam_opencat, are available for use with 64-bit programs. Oracle Corporation, Inc. does not support a 64-bit version of libsamrpc.

For more details about each library routine, see the individual corresponding man page for that routine. Library routines contained in libsam are found in section 3 of the online man pages. Library routines contained in libsamrpc are found in section 3X of the online man pages.

USING libsam

No special initialization or configuration is required prior to using the API library routines in libsam. The application program must be linked with libsam, however. For information on the routines, see the individual libsam man pages, all of which are listed in the SEE ALSO section of this man page.

USING libsamrpc

The source code for libsamrpc is included in the release for customers who wish to write and run application programs on platforms that do not run the Solaris operating system. In these cases, the library must be ported to the client machine. The source code is located in /opt/SUNWsamfs/client/src. Example application programs are located in /opt/SUNWsamfs/client/examples.

Specifying the Server Machine

A call to sam_initrpc is required before any other RPC client API calls can be executed successfully. Only one sam_initrpc call is required, followed by any number of other client API calls (other than sam_closerpc). The sam_initrpc call accepts one argument: a pointer to a character string that specifies the name of the server machine. If this pointer is NULL, sam_initrpc checks for an environment variable named SAMHOST. If this environment variable is set, that name is used for the server machine. If there is no SAMHOST environment variable, the default server name samhost is used.

In summary, the name of the server machine can be specified in any of three ways, which are checked by sam_initrpc in

the following order:

1. As an argument to the sam_initrpc call.

2. As the environment variable SAMHOST.

3. By accepting the default server name, samhost.

RPC Server Process

The RPC API server process receives and processes requests from the client. This server process, /opt/SUNWsamfs/sbin/sam-rpcd, must be run on the same machine as the file system. The sam-rpcd daemon must be running for client requests to execute successfully.

The sam-rpcd daemon is started automatically by sam-amld if the appropriate entry is made in the defaults.conf file. For information on editing the defaults.conf file, see Configuring the API later in this man page.

The sam-rpcd daemon can also be started manually. It should be run as superuser. The sam-rpcd command accepts no arguments.

The sam-rpcd daemon services the requests it receives by making the appropriate system call on the server machine and then returning the output or result to the client. For more information on this daemon, see the sam-rpcd(1M) man page.

Configuring the API

The following steps describe setting up the API server and clients. These steps assume that your software is properly configured and running.

Step 1: Configure the API Server

For the server portion of the API to run successfully, the following conditions must be present:

- o The RPC program name and number pair must be known on the server machine
- o The RPC program name and number pair must be the same as the pair used on the API client machines.

Make an entry for the RPC program name and number. The RPC program number is a number chosen by you. The RPC program name is samfs. The name and number pair must be the same on the server and all clients. The /etc/nsswitch.conf file determines where you should specify the RPC program name and number pair. For more information on this, see the nsswitch.conf(4) man page.

In /etc/rpc (or the NIS database), add the following line:

```
samfs          150005
```

In /etc/services (or the NIS database), add the following line:

```
samfs          5012/tcp  # SAM-QFS API
```

The API server is started automatically by the sam-amld daemon if the following entry is made in the defaults.conf file (note that changes to the defaults.conf file do not take effect until the next time the sam-amld daemon is initialized):

```
samrpc = on
```

The sam-rpcd daemon is not automatically started if no entry for it appears in the defaults.conf file or if the following entry appears in the file:

```
samrpc = off
```

For more information about the defaults.conf file, see the defaults.conf(4) man page.

Step 2: Configure the API Client Machines

The following two configuration components must be present on the client machine for the RPC communication to be successful:

o The name of the server machine.

o The RPC program name and number pair.

Make an entry for the RPC program name and number on all client machines, as you did on the API server machine previously. Again, the RPC program name must be samfs. The RPC program number is a number chosen by you, but it must be the same on the server and client machines.

In /etc/rpc (or the NIS database), add the following line:

```
samfs          150005
```

The host name of the server machine must be known on the client machine. For default cases, the host name samhost must be listed as an alias for the SAM-QFS file system server machine. For more information, see the sam_initrpc(3X) man page.

Authentication and libsamrpc

Authentication information is generated at the time of the sam-initrpc call. This information consists of the user identification (uid) and group identification (gid) of the calling process. It is associated with the connection made to the RPC server process.

Subsequent libsamrpc calls have this information associated. When the request is received by the RPC server process on the server machine, the uid and gid information is used. File access and operations are granted or denied based on this information.

It is important that the server machine have a common uid and gid space with the client machines.

SEE ALSO

sam_advise(3), sam_archive(3), sam_rearch(3), sam_exarchive(3), sam_unarchive(3), sam_unrearch(3), sam_damage(3), sam_undamage(3), sam_cancelstage(3), sam_closecat(3), sam_devstat(3), sam_devstr(3), sam_getcatalog(3), sam_lstat(3), sam_ndevstat(3), sam_opencat(3), sam_readrminfo(3), sam_release(3), sam_request(3), sam_restore_copy(3), sam_restore_file(3), sam_segment(3), sam_setfa(3), sam_ssum(3), sam_stage(3), sam_stat(3).

```
     sam_archive(3X), sam_closerpc(3X), sam_initrpc(3X),
     sam_lstat(3X), sam_release(3X), sam_stage(3X), sam_stat(3X).
```

qfs_listio(3)

```
NAME
     qfs_lio_read, qfs_lio_write, qfs_lio_poll, qfs_lio_wait -
     Issues list I/O or waits for listio.

SYNOPSIS
     cc [flag ...] file ...  -L/opt/SUNWsamfs/lib
     -R/opt/SUNWsamfs/lib -lsam [library ...]

     #include "/opt/SUNWsamfs/include/listio.h"

     int qfs_lio_init(qfs_lio_handle_t *hdl);

     int qfs_lio_read(int fd, int mem_list_count, void
     **mem_addr, size_t *mem_count, int file_list_count, offset_t
     *file_off, offset_t *file_len, qfs_lio_handle_t *hdl);

     int qfs_lio_write(int fd, int mem_list_count, void
     **mem_addr, size_t *mem_count, int file_list_count, offset_t
     *file_off, offset_t *file_len, qfs_lio_handle_t *hdl);

     int qfs_lio_wait(qfs_lio_handle_t *hdl);

AVAILABILITY
     SUNWqfs
     SUNWsamfs

DESCRIPTION
     The qfs_lio_read() function issues a listio read for an open
     file descriptor.

     The qfs_lio_write() function issues a listio write for an
     open file descriptor.

     The qfs_lio_init() must be used to initialize a handle
     object before passing it to one of the other interfaces.

     The qfs_lio_wait() can be issued to wait until all I/O in
     the listio call has completed.

ARGUMENTS
     These functions accept the following arguments:

     fd        issues I/O for a file using a Sun QFS or SAM-QFS
               ioctl call.

     mem_list_count
               is the number of elements in the mem_addr and
               mem_count arrays.

     mem_addr, mem_count
```

```
                    are arrays describing a list of memory regions.

          file_list_count
                    is the number of elements in the file_off and
                    file_len arrays.

          file_off, file_len
                    are arrays describing a list of file regions.

          hdl       points to an opaque value that is used to indicate
                    the status of an asynchronous list I/O request.
                    If hdl is non-null, the function returns when all
                    I/O has issued.  If hdl is NULL, the function
                    returns when all I/O has been completed.

     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
          The qfs_lio_read(), qfs_lio_write(), and qfs_lio_wait()
          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 superuser.

          EFAULT              mem_addr, mem_count, file_off, or
                              file_len points to an illegal address.

          EINTR               A signal was caught during the qfs_lio()
                              function.

     SEE ALSO
          setfa(1), sam_setfa(3), directio(3C),
```

sam_advise(3)

```
     NAME
          sam_advise - Provides advice to the file system

     SYNOPSIS
          cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam [library ... ]

          #include "/opt/SUNWsamfs/include/lib.h"

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

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

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

OPTIONS

b Advises the system to use buffered (paged) I/O. The default I/O mode is buffered (uses the page cache). At the last close, the type of I/O is set back to paged or direct based on the mount option forcedirectio or the directio attribute set by the setfa command.

d Return the advice on the file to the default, i.e. the qwrite is reset to the mount setting. When this option is specified, the advice is reset to the default. If it is used, it should be the first character in the string.

p Obsolete. Now does nothing, but remains for compatibility.

r Advises the system to use direct (raw) I/O (see directio(3C) for Solaris 2.6 and above). The default I/O mode is buffered (uses the page cache). At the last close, the type of I/O is set back to paged or direct based on the mount option forcedirectio or the directio attribute set by the setfa command.

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 file system. (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 superuser.
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 com-
                         ponent    exceeds    {NAME_MAX}    while
                         {_POSIX_NO_TRUNC} is in effect.

     ENOENT              The named file does not exist or is  the
                         null pathname.

     ENOLINK             path points to a remote machine and  the
                         link   to  that  machine  is  no  longer
                         active.

     ENOTDIR             A component of the path prefix is not  a
                         directory.

SEE ALSO
     setfa(1),     sam_setfa(3),     directio(3C),     mlock(3C),
     mount_samfs(1M), mcf(4)
```

sam_archive(3)

```
NAME
     sam_archive - Sets archive attributes on a file or directory

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam [library ... ]

     #include "/opt/SUNWsamfs/include/lib.h"

     int sam_archive(const char *path, const char *ops);

DESCRIPTION
     sam_archive() sets archive attributes on a file or directory
     using a Sun QFS or SAM-QFS system call.  path is the file on
     which to set the attributes.  ops is the character string of
     options,   for   example:   "dn".   Individual  options  are
     described below.

OPTIONS
     C    Specifies concurrent archiving for this file. This file
          can  be  archived even if opened for write. The archive
          time is regulated by the modification time.  Note,  nfs
          files  are  not  opened and are by default concurrently
          archived. Concurrent archiving is useful for databases,
          however  caution  is  advised since archiving can occur
          while the file is being modified and this can result in
          wasted  media.  The  default  is  to disallow archiving
          while the file is opened for write.

     I    Support inconsistent archive copies. This means that an
          archive  copy  can be created even if the file is modi-
          fied while it is being copied to the media. By default,
          the  archive  copy  is disallowed if the file is incon-
          sistent, that is, if the file is modified while it  was
          being  copied  to  the media.  Note, the file cannot be
          staged if the copy  is  marked  inconsistent;  however,
```

after a samfsrestore, the inconsistent flag is removed from the archive copy and the file can be staged.

Inconsistent archiving is useful for databases, however caution is advised because it a file can be staged from an inconsistent copy after the file is restored using samfsrestore.

d   Return the archive attributes on the file to the default, i.e. archive the file according to the archiver rules. When this option is specified, the attributes are reset to the default. If it is used, it should be the first character in the string.

i   Specifies that the file be immediately archived if it is not already archived.

w   Wait for the file to have at least 1 archive copy before completing. Not valid with d or n.

Note that it may take a long time for the file to be archived.

W   Wait for the file to have all its required archive copies before completing. Not valid with d or n.

Note that it may take a long time for the file to be archived.

n   Specifies that this file never be archived. Not valid with either of the checksum g (generate) or u (use) attributes. (See ssum(1) or sam_ssum(3)).

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of -1 is returned and errno is set to indicate the error.

ERRORS

sam_archive() fails if one or more of the following are true:

EINVAL	An invalid option was specified, or the file is neither a regular file nor a directory.
EPERM	Not the owner or superuser.
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 com-

```
                    ponent    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)

sam_audit(3)

NAME
     sam_audit - Audits media

SYNOPSIS

```
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib  -lsamapi  [library ... ]

     #include "/opt/SUNWsamfs/include/samapi.h"

     int   sam_audit(ushort_t   eq_number,   uint_t    ea,    int
     wait_response);
```

DESCRIPTION
     sam_audit() performs a catalog audit on either a single element address in a robot or all the element addresses in a robot, for either optical or tape robotic devices. Specifying ea as the number of the robot element address will perform a catalog audit of that element address. If the ea is set to the global ROBOT_NO_SLOT value, an audit of the entire robot will be performed. The call will return immediately after issuing the command if zero is specified for wait_response value. Other values for wait_response will give undefined results.

RETURN VALUES
     Upon succesful completion a value of 0 is returned. Otherwise, a value of -1 is returned and errno is set to indicate the error.

ERRORS
     sam_audit() fails if one or more of the following are true:

```
     ER_FIFO_PATH_LENGTH The path and filename for the FIFO  pipe
                         is too long.

     ER_NO_EQUIP_ORDINAL eq_number is  not  a  defined  equipment
                         number in this configuration.

     ER_NO_MASTER_SHM    No Sun  QFS  or  SAM-QFS  master  shared
```

```
                        memory  segment defined.  Check that the
                        Sun QFS  or  SAM-QFS  file  systems  are
                        mounted.

     ER_NO_MASTER_SHM_ATT
                        No Sun  QFS  or  SAM-QFS  master  shared
                        memory  segment  found.   Check that the
                        Sun QFS  or  SAM-QFS  file  systems  are
                        mounted.

     ER_NOT_VALID_SLOT_NUMBER
                        ea is not a valid element address in the
                        robot at equipment number eq_number

     ER_NO_RESPONSE_FIFO Unable to create the response FIFO pipe.

     ER_NO_STAT_ROBOT_CATALOG
                        The  robot  media  changer  catalog  for
                        equipment  number  eq_number  cannot  be
                        accessed for status.

     ER_OPERATOR_NOT_PRIV
                        Operator does  not  have  permission  to
                        perform a full audit

     ER_ROBOT_CATALOG_MISSING
                        The  robot  media  changer  catalog  for
                        equipment  number  eq_number  is missing
                        and a full audit is required.

     ER_SLOT_NOT_OCCUPIED
                        No media exists at the  element  address
                        number ea specified

     ER_UNABLE_TO_MAP_CATALOG
                        The  catalog  for  the  removable  media
                        changer at equipment number eq_number is
                        unable to be mapped into memory.

FILES
     mcf                The configuration file for  Sun  QFS  or
                        SAM-QFS

SEE ALSO
     auditslot(1M),  export(1M),   import(1M),   move(1M),   sam-
     robotsd(1M).

     mcf(4).
```

sam_cancelstage(3)

```
NAME
     sam_cancelstage - Cancels a file stage

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

     #include "/opt/SUNWsamfs/include/lib.h"

     int sam_cancelstage(const char *path)

DESCRIPTION
     sam_cancelstage() cancels the stage of the file  pointed  to
     by  path.  Only the file owner or superuser can perform this
     operation on the file.

RETURN VALUES
     Upon successful completion a value of 0 is returned.  Other-
     wise, 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 com-
                         ponent    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).
```

sam_chmed(3)

```
NAME
     sam_chmed - Changes library catalog flags

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib  -lsamapi  [library ... ]

     #include "/opt/SUNWsamfs/include/samapi.h"

     int sam_chmed(ushort_t eq_number, uint_t ea, int  partition,
     char   *media,   char  *vsn,  int  flags,  int  on_off,  int
     wait_response);

DESCRIPTION
     sam_chmed() sets or clears library catalog flags for a  par-
     ticular  VSN  vsn  of  media type or a robotic media changer
     equipment number eq_number and ea and partition number.   If
     the on_off is set to one (1) or zero (0), the flag positions
     represented by the flags will be  set  or  cleared,  respec-
     tively.   The call will return immediately after issuing the
     command if zero is specified for wait_response value.  Other
     values for wait_response will give undefined results.

RETURN VALUES
     Upon succesful completion a value of 0 is returned.   Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.

ERRORS
     sam_chmed() fails if one or more of the following are true:

     ER_FIFO_PATH_LENGTH The path and filename for the FIFO  pipe
                         is too long.

     ER_INVALID_FLAG_SET One of the flags specified to be set  or
                         cleared is not defined.

     ER_INVALID_MEDIA_TYPE
                         media specified is  not  a  valid  media
                         type.  See  mcf(4) for valid media types
                         to be specified.

     ER_INVALID_VSN_LENGTH
                         vsn specified is not the correct length.

     ER_NO_EQUIP_ORDINAL eq_number is  not  a  defined  equipment
                         number in this configuration.

     ER_NO_MASTER_SHM    No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment defined.  Check that the
                         Sun QFS  or  SAM-QFS  file  systems  are
                         mounted.

     ER_NO_MASTER_SHM_ATT
                         No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment  found.   Check that the
                         Sun QFS  or  SAM-QFS  file  systems  are
```

```
                        mounted.

     ER_NO_RESPONSE_FIFO Unable to create the response FIFO pipe.

     ER_NO_STAT_ROBOT_CATALOG
                        The  robot  media  changer  catalog  for
                        equipment  number  eq_number  cannot  be
                        accessed for status.

     ER_NOT_VALID_SLOT_NUMBER
                        ea is not a valid element address in the
                        robot at equipment number eq_number

     ER_ON_OFF_BAD_VALUE on_off value  specified  is  not  valid;
                        must be one (1) or zero (0).

     ER_OPERATOR_NOT_PRIV
                        Operator does  not  have  permission  to
                        change media catalog flags

     ER_ROBOT_CATALOG_MISSING
                        The  robot  media  changer  catalog  for
                        equipment  number  eq_number  is missing
                        and a full audit is required.

     ER_ROBOT_DEVICE_REQUIRED
                        eq_number is not a robotic device

     ER_SLOT_NOT_OCCUPIED
                        No media exists at the  element  address
                        ea specified

     ER_SLOT_OR_VSN_REQUIRED
                        Either  a  element  address  or   volume
                        serial  number  must  be specified; both
                        cannot be specified.

     ER_UNABLE_TO_MAP_CATALOG
                        The  catalog  for  the  removable  media
                        changer at equipment number eq_number is
                        unable to be mapped into memory.

FILES
     mcf                The configuration file for  Sun  QFS  or
                        SAM-QFS file systems.

SEE ALSO
     chmed(1M), sam-recycler(1M), samu(1M).

     mcf(4).
```

sam_clear_request(3)

```
NAME
     sam_clear_request - Clears entry from removable media  mount
     requests

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib  -lsamapi  [library ... ]

     #include "/opt/SUNWsamfs/include/samapi.h"

     int sam_clear_request(uint_t slot, int wait_response);

DESCRIPTION
     sam_clear_request() clears removable media entry from  mount
     request  list.  The call will return immediately after issu-
     ing the command  if  zero  is  specified  for  wait_response
     value.   Other  values for wait_response will give undefined
     results.

RETURN VALUES
     Upon succesful completion a value of 0 is returned.   Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.

ERRORS
     sam_clear_request() fails if one or more  of  the  following
     are true:

     ER_FIFO_PATH_LENGTH The path and filename for the FIFO  pipe
                         is too long.

     ER_NO_MASTER_SHM    No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment defined.  Check that the
                         Sun QFS  or  SAM-QFS  file  systems  are
                         mounted.

     ER_NO_MASTER_SHM_ATT
                         No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment  found.   Check that the
                         Sun QFS  or  SAM-QFS  file  systems  are
                         mounted.

     ER_NO_RESPONSE_FIFO Unable to create the response FIFO pipe.

     ER_OPERATOR_NOT_PRIV
                         Operator does  not  have  permission  to
                         perform a full audit

SEE ALSO
     samu(1M).
```

sam_closecat(3)

```
NAME
     sam_closecat - Closes a catalog handle

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

     #include "/opt/SUNWsamfs/include/catalog.h"

     int sam_closecat(int cat_handle);

AVAILABILITY
     32-bit programs only

DESCRIPTION
     sam_closecat() deallocates the catalog handle  indicated  by
     cat_handle.   cat_handle is a catalog "handle" obtained from
     a previous call to sam_opencat(). Deallocation of the  cata-
     log handle ends access to the automated library catalog that
     it referred to, and makes the catalog handle  available  for
     return by subsequent calls to sam_opencat().

RETURN VALUES
     Upon successful completion, a value of 0 is returned.   Oth-
     erwise,  a value of -1 is returned and errno is set to indi-
     cate the error.

ERRORS
     sam_closecat() fails if one or more of the  following  error
     conditions are true:

     EBADFILE            cat_handle is invalid.

SEE ALSO
     sam_getcatalog(3), sam_opencat(3).
```

sam_damage(3)

```
NAME
     sam_damage - Sets damaged attribute on a file or directory

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

     #include "/opt/SUNWsamfs/include/lib.h"

     int sam_damage(const char *path, int num_opts, ... );

DESCRIPTION
     sam_damage() lets you mark archive copies of  a  file  or  a
     directory  as  damaged,  using a Sun Storage Archive Manager
     system call.  path is the file on which to  set  the  attri-
     butes,  followed  by a sequence of num_opts input characters
     or options.  Individual options are described below.
```

The function marks copies of a file or directory as damaged based on the archive copy number and/or the media type and VSN specified. There are several ways to mark one or more copies as damaged. These ways are as follows:

o By copy number

o By copy number, media type, and VSN

o By copy number and media type

o By media type

o By media type and VSN

If a fatal error is detected when staging an archive copy, that archive copy is marked as damaged. An archive copy that is damaged is not selected for staging.

OPTIONS

a
: Rearchives the damaged copy.

c copy_no
: Marks the specified archive copy number as damaged. If one or more ’c’ options are specified, only those archive copies (1, 2, 3, or 4) are marked as damaged. Specify 1, 2, 3, or 4 for copy_no. Either a "c copy_no" or a "m media" option must be specified.

M
: Marks only metadata as damaged. This includes directories, the segment index, and removable-media files. Regular files are not marked as damaged. If you are marking a directory as damaged, you must specify the "M" option.

m media_type
: Marks all copies from the specified media_type as damaged. For the list of possible media_type specifications, see the mcf(4) man page. Either a "c copy_no" or a "m media" option must be specified. If you specify a "m media" option, you can also specify a "v vsn" option.

o
: Specifies that the file must be online before it is marked as damaged. If the file is offline, the sam_damage function stages the file to disk before deleting any entries.

v vsn
: Marks the archive copies on vsn as damaged. For vsn, specify a volume serial name (VSN). If you specify a "v vsn" option, you must also specify a "m media" 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_damage() 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 superuser.

     EFAULT              Argument points to an illegal address.

     EINTR               A   signal   was   caught   during   the
                         sam_damage() 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 com-
                         ponent    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
     damage(1m), mcf(4)
```

sam_devstat(3)

```
NAME
     sam_devstat, sam_ndevstat, sam_devstatl - Gets device status

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

     #include "/opt/SUNWsamfs/include/devstat.h"

     int sam_devstat(ushort_t eq, struct sam_devstat *buf, size_t
     bufsize);

     int  sam_ndevstat(ushort_t  eq,  struct  sam_ndevstat  *buf,
     size_t bufsize);

     int  sam_devstatl(ushort_t  eq,  struct  sam_devstatl  *buf,
     size_t bufsize);

DESCRIPTION
     sam_devstat()   and   sam_ndevstat()   are   obsolete,   use
     sam_devstatl().

     sam_devstatl() obtains information about the device  identified
by the equipment number, eq.

     buf is a pointer to  a  sam_devstatl  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_devstatl).

     The contents of the structure pointed to by buf include  the
     following members for sam_devstatl:

          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 */
          uint64_t space;       /* Space left on device */
          uint64_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 char-
               acters.

     state     The  state  of  the  device.   This  field  is  an
               enumerated type defined in devstat.h.

     status    The status of the device.  Status bits are defined
               in  the file devstat.h.  Also, the library routine
               sam_devstr(3)  is  available  to   translate   the
               numeric  status  field  into a character string as
               displayed in the Sun QFS or SAM-QFS graphical user
               interfaces  and in the Sun QFS or SAM-QFS adminis-
               trative tool samu(1M).

     space     The space left on the device,  in  the  number  of
               1024 blocks.

     capacity  The capacity of the device, in the number of  1024
               blocks.

RETURN VALUES
     Upon successful completion a value of 0 is returned.  Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.

ERRORS
     sam_devstatl() fails if one or more  of  the  following  are
     true:

     ENODEV              The equipment number supplied is  not  a
```

```
                        valid number, or no device is configured
                        with that equipment number.

     EACCES             The program does not have permission  to
                        access  the  Sun  QFS  or SAM-QFS shared
                        memory segment.

     EINVAL             The size  of  the  Sun  QFS  or  SAM-QFS
                        shared memory segment is incorrect.  You
                        may need to recompile your program  with
                        the  current  version  of the Sun QFS or
                        SAM-QFS software.

     ENOENT             Access to the Sun QFS or SAM-QFS  shared
                        memory  segment has failed; possibly Sun
                        QFS or SAM-QFS is not running.

     EMFILE             The Sun QFS  or  SAM-QFS  shared  memory
                        segment  could  not  be accessed because
                        the number  of  shared  memory  segments
                        attached  to  the  calling process would
                        exceed the system-imposed limit.

     ENOMEM             The available data space  is  not  large
                        enough  to accommodate access to the Sun

                        QFS or SAM-QFS shared memory segment.

     E2BIG              For  sam_devstat()  and   sam_ndevstat()
                        only.   The  capacity or space is larger
                        than UINT_MAX or the name field  of  the
                        device  was  too long to fit in the name
                        field.  Use sam_devstatl().

SEE ALSO
     samu(1M).

     sam_devstr(3).
```

sam_devstr(3)

```
NAME
     sam_devstr - Translates numeric device status into character
     string

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

     #include "/opt/SUNWsamfs/include/lib.h"

     char *sam_devstr(uint_t status)

DESCRIPTION
     sam_devstr() translates an unsigned integer, status, into  a
     character  status  string  based  on the bits set in status,
     returning a pointer to the character string.
```

The meanings of the characters in the string returned are as follows:

s---------	Volume is being scanned.
m---------	If a file system, the file system is mounted.
M---------	The device is in maintenance mode.
-E--------	Device received an unrecoverable error.
-a--------	Device is auditing. If a file system, 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.
----C-----	Device needs cleaning.
-----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).
--------V-	The device is verifying media (tape only).
--------Q-	The device verify media command was canceled (tape only).
---------r	Device is ready. If a file system, its disk space is being released.
---------R	Device is ready and the volume is read-only.
---------W	Device is ready and the cartridge write-protect tab is set.
---------p	Device is present.

RETURN VALUES

A pointer to the character status string is returned.

SEE ALSO

sam_devstat(3).

sam_errno(3)

NAME

sam_errno - Interprets error number to create text string

SYNOPSIS

```
cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib  -lsamapi  [library ... ]

#include "/opt/SUNWsamfs/include/samapi.h"

char *sam_errno(int err);
```

DESCRIPTION

sam_errno() validates the error number and, if valid, obtains the text string for the error number err value.

RETURN VALUES

An appropriate character string for the error number err is returned. If the error number err is invalid, the message "Unknown error number" is returned.

SEE ALSO

errno(1M), strerror(1M).

sam_exarchive(3)

```
NAME
     sam_exarchive - Exchanges archive copies of a file or direc-
     tory

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

     #include "/opt/SUNWsamfs/include/lib.h"

     int sam_exarchive(const char *path, int num_opts, ... );

DESCRIPTION
     sam_exarchive() lets you exchange archive copies of  a  file
     or  a  directory  using a Sun Storage Archive Manager system
     call.  path is the file whose specified archive  copies  are
     to  be  exchanged,  followed by a sequence of num_opts input
     characters or options.   Individual  options  are  described
     below.

OPTIONS
     c copy_m

     c copy_n
          Specifies the copies to be exchanged.   The  copy_m  is
          exchanged with copy_n.  Exactly two 'c' options must be
          specified.  The first copy (copy_m) must have  a  valid
          archive entry.

     M    Exarchives metadata only.  This  includes  directories,
          the  segment index, and removable media files.  Regular
          files are not exarchived.  If  you  are  exarchiving  a
          directory, you must specify the "M" option.

RETURN VALUES
     Upon successful completion a value of 0 is returned.  Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.

ERRORS
     sam_exarchive() 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 superuser.

     EFAULT              Argument points to an illegal address.

     EINTR               A   signal   was   caught   during   the

                         sam_exarchive() 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
exarchive(1m), mcf(4)

sam_export(3)

NAME
sam_export - Exports media from the removable media robotic device

SYNOPSIS
```
cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib  -lsamapi  [library ... ]

#include "/opt/SUNWsamfs/include/samapi.h"

int sam_export(ushort_t eq_number, char *vsn, uint_t ea, int
wait_response, int one_step);
```

DESCRIPTION
sam_export() requests the robotic media changer to place the selected media that belongs in ea or is labeled with vsn into the mail-slot of the media changer. Either the vsn or the ea must be specified. The vsn should be set to a NULL pointer if ea is specified. The ea should be set to ROBOT_NO_SLOT if the vsn is specified.

The call will return immediately after issuing the command if zero is specified for wait_response value. Other values for wait_response will give undefined results.

For network-controlled media changers such as the GRAU using the GRAU ACT interface, IBM 3494, or Sony libraries using PSC, this interface 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 Sony. For STK network-controlled media changers using ACSLS, this interface removes the entry from the catalog only if zero is specified for one_step value. Other values for one_step will cause the physical removal of media and also remove the media from the catalog.

```
RETURN VALUES
     Upon succesful completion a value of 0 is returned.   Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.

ERRORS
     sam_export() fails if one or more of the following are true:

     ER_FIFO_PATH_LENGTH The path and filename for the FIFO  pipe
                         is too long.

     ER_NO_EQUIP_ORDINAL eq_number is  not  a  defined  equipment
                         number in this configuration.

     ER_NO_MASTER_SHM    No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment defined.  Check that the

                         Sun QFS  or  SAM-QFS  file  systems  are
                         mounted.

     ER_NO_MASTER_SHM_ATT
                         No Sun QFS or SAM-QFS shared memory seg-
                         ment  found.   Check that the Sun QFS or
                         SAM-QFS file systems are mounted.

     ER_NO_RESPONSE_FIFO Unable to create the response FIFO pipe.

     ER_NO_STAT_ROBOT_CATALOG
                         The  robot  media  changer  catalog  for
                         equipment  number  eq_number  cannot  be
                         accessed for status.

     ER_NOT_VALID_SLOT_NUMBER
                         ea is not a valid element address in the
                         robot at equipment number eq_number

     ER_OPERATOR_NOT_PRIV
                         Operator does  not  have  permission  to
                         export removable media.

     ER_ROBOT_CATALOG_MISSING
                         The  robot  media  changer  catalog  for
                         equipment  number  eq_number  is missing
                         and a full audit is required.

     ER_ROBOT_DEVICE_REQUIRED
                         Equipment  number   eq_number   is   not
                         defined as a robotic device.

     ER_SLOT_NOT_OCCUPIED
                         ea does not contain any removable media.

     ER_SLOT_OR_VSN_REQUIRED
                         Either ea or vsn must be specified

     ER_UNABLE_TO_MAP_CATALOG
                         The  catalog  for  the  removable  media
                         changer at equipment number eq_number is
                         unable to be mapped into memory.
     ER_VSN_NOT_FOUND_IN_ROBOT
                         vsn specified cannot  be  found  in  the
                         specified   robot  at  equipment  number
                         eq_number

FILES
     mcf                 The configuration file for Sun  QFS  and
                         SAM-QFS file systems.

SEE ALSO
     build_cat(1M), dump_cat(1M),  export(1M),  import(1M),  sam-
     robotsd(1M).

     sam_import(3).

     mcf(4).
```

sam_getcatalog(3)

```
NAME
     sam_getcatalog,  sam_getcatalogl,  sam_getcataloglv  -  Gets
     catalog entries

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

     #include "/opt/SUNWsamfs/include/catalog.h"

     int sam_getcatalog(int cat_handle,  uint  start_entry,  uint
     end_entry, struct sam_cat_ent *buf, size_t entbufsize);

     int sam_getcatalogl(int cat_handle, uint  start_entry,  uint
     end_entry, struct sam_cat_entl *buf, size_t entbufsize);

     int sam_getcataloglv(int cat_handle, uint start_entry,  uint
     end_entry, struct sam_cat_entlv *buf, size_t entbufsize);

DESCRIPTION
     sam_getcatalog() and  sam_getcatalogl()  are  obsolete,  use
     sam_getcataloglv().

     sam_getcataloglv() obtains a range of entries from the cata-
     log  of  an automated library or the historian.  The catalog
     from  which  entries  will  be  obtained  is  indicated   by
     cat_handle.  cat_handle is similar to a file descriptor, and
     is returned from a previous call to sam_opencat(). The range
     of  entries  is  indicated  by  start_entry  and  end_entry.
     start_entry must be less than or  equal  to  end_entry,  and
     must be in the range of valid slot numbers for the automated
     library (or historian).  buf is a pointer  to  an  array  of
     sam_cat_entlv  structures,  into  which  the  catalog  entry
     information is placed.  This array should be large enough to
     hold  the  number  of  entries requested.  entbufsize is the
     size of a single sam_cat_entlv structure, usually  indicated
     by sizeof(struct sam_cat_entlv).
```

The contents of a sam_cat_entlv structure include the following members:

```
/* catalog table entry */
uint_t   status;      /* Catalog entry status */
char     media[4];    /* Media type */
char     vsn[32];     /* VSN */
int      access;      /* Count of accesses */
uint64_t capacity;    /* Capacity of volume */
uint64_t space;       /* Space left on volume */
uint64_t 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) */
time_t   lvtime;      /* Last verified time */
uint64_t lvpos;       /* Last verified position */
```

The last verified time and position are valid if CS_DIV(status) is non-zero.

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_getcataloglv() fails if one or more of the following are true:

EBADF	The catalog handle provided is invalid.
EFAULT	buf is an invalid address.
EOVERFLOW	The catalog library software returned more information than was requested.
ENOENT	This is no longer an active catalog.
EINVAL	The buffer size provided is invalid, or start_entry or end_entry is invalid. (Either start_entry is less than zero, end_entry is greater than the number of entries in the catalog, or start_entry is greater than end_entry.)
E2BIG	For sam_getcatalog() only. The capacity or space is larger than UINT_MAX. Use sam_getcataloglv().

sam_closecat(3), sam_opencat(3), tpverify(1M).

sam_getfsdata(3)

```
NAME
     sam_getfsdata - Obtains size of family  set  and  number  of
     disks used

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib  -lsamapi  [library ... ]

     #include "/opt/SUNWsamfs/include/samapi.h"

     int sam_getfsdata(int eq, long long *space, long long *capa-
     city, int *disk_count);

DESCRIPTION
     sam_getfsdata() obtains the space and capacity of the speci-
     fied  family  set  and  the  number of disk and/or partition
     members in the specified family set.

RETURN VALUES
     Upon succesful completion a value of 0 is returned.   Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.

ERRORS
     sam_getfsdata() fails if one or more of  the  following  are
     true:

     ENODEV              No disks or partitions found for  family
                         set equipment number eq_number specified

     ER_DEVICE_NOT_CORRECT_TYPE
                         Device referenced  by  equipment  number
                         eq_number  is  not of family set or disk
                         set type.

     ER_NO_MASTER_SHM    No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment defined.  Check that the
                         Sun QFS  or  SAM-QFS  file  systems  are
                         mounted.

     ER_NO_MASTER_SHM_ATT
                         No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment  found.   Check that the
                         Sun QFS  or  SAM-QFS  file  systems  are
                         mounted.

FILES
     mcf                 The configuration file for Sun  QFS  and
                         SAM-QFS file systems.
```

sam_getfsdisks(3)

```
NAME
     sam_getfsdisks - Obtains list of disks in family set

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib  -lsamapi  [library ... ]

     #include "/opt/SUNWsamfs/include/samapi.h"

     int sam_getfsdisks(int eq, struct sam_fs_disk *disks, size_t
     disks_size, uint start_pos, int disk_count);

DESCRIPTION
     sam_getfsdisks() obtains the  data  for  each  disk  in  the
     specified  family set, based on the starting position number
     start_pos up to disk_count disks.

RETURN VALUES
     Upon succesful completion a value of 0 is returned.   Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.

ERRORS
     sam_getfsdisks() fails if one or more of the  following  are
     true:

     EFAULT              No address specified to store disk data.

     EINVAL              Size of storage area for  disk  data  is
                         too small for number of disks requested.

     ER_DEVICE_NOT_CORRECT_TYPE
                         Device referenced  by  equipment  number
                         eq_number  is  not of family set or disk
                         set type.

     ER_NO_DEVICE_FOUND  No disks or partitions found for  family
                         set equipment number eq

     ER_NO_MASTER_SHM    No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment defined.  Check that the
                         Sun QFS  or  SAM-QFS  file  systems  are
                         mounted.

     ER_NO_MASTER_SHM_ATT
                         No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment  found.   Check that the
                         Sun QFS  or  SAM-QFS  file  systems  are
                         mounted.

FILES
     mcf                 The configuration file for Sun  QFS  and
                         SAM-QFS file systems.
```

sam_import(3)

```
NAME
     sam_import - Imports media to the  removable  media  robotic
     device

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib  -lsamapi  [library ... ]

     #include "/opt/SUNWsamfs/include/samapi.h"

     int   sam_import(ushort_t   eq_number,   char   *vsn,   char
     *media_nm, int audit_eod, int wait_response);

DESCRIPTION
     sam_import() requests the robotic media  changer  to  import
     the  selected  media into the robot.  The media is placed in
     the first available element address  of  the  catalog.   For
     GRAU,  STK,  or  IBM  libraries,  the vsn must be specified.
     Physical import of media within the GRAU and  STK  are  per-
     formed by utilites supplied by the vendor.  Both the vsn and
     the media_nm need to be specified to import  into  the  His-
     torian.

     The call will return immediately after issuing  the  command
     if  zero is specified for wait_response value.  Other values
     for wait_response will give undefined results.

RETURN VALUES
     Upon succesful completion a value of 0 is returned.   Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.

ERRORS
     sam_import() fails if one or more of the following are true:

     ER_AUDIT_EOD_NOT_HISTORIAN
                         A non-zero audit_eod flag, requesting an
                         audit  to  end-of-data, cannot be speci-
                         fied for the Historian.

     ER_DEVICE_NOT_READY The specified eq_number  device  is  not
                         ready.

     ER_FIFO_PATH_LENGTH The path and filename for the FIFO  pipe
                         is too long.

     ER_HISTORIAN_MEDIA_ONLY
                         media_nm can only be specified  for  the
                         Historian.

     ER_INVALID_MEDIA_TYPE
                         The specified media_nm is  not  a  valid

                         media type.

     ER_MEDIA_FOR_HISTORIAN
                         The  type  of  media  media_nm  must  be
```

```
                        specified for the Historian.

     ER_NO_EQUIP_ORDINAL eq_number is  not  a  defined  equipment
                        number in this configuration.

     ER_NO_MASTER_SHM    No Sun  QFS  or  SAM-QFS  master  shared
                        memory  segment defined.  Check that the
                        Sun QFS  or  SAM-QFS  file  systems  are
                        mounted.

     ER_NO_RESPONSE_FIFO Unable to create the response FIFO pipe.

     ER_NO_MASTER_SHM_ATT
                        No Sun  QFS  or  SAM-QFS  master  shared
                        memory  segment  found.   Check that the
                        Sun QFS  or  SAM-QFS  file  systems  are
                        mounted.

     ER_OPERATOR_NOT_PRIV
                        Operator does  not  have  permission  to
                        import removable media.

     ER_ROBOT_DEVICE_REQUIRED
                        Equipment  number   eq_number   is   not
                        defined as a robotic device.

     ER_VSN_BARCODE_REQUIRED
                        vsn must be specified for the GRAU, STK,
                        and  IBM  media  changers  and  the His-
                        torian.

FILES
     mcf                The configuration file for Sun  QFS  and
                        SAM-QFS file systems.

SEE ALSO
     export(1M), import(1M), sam-robotsd(1M).

     sam_export(3).

     mcf(4).
```

sam_load(3)

```
NAME
     sam_load - Loads media on the removable media device

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib  -lsamapi  [library ... ]

     #include "/opt/SUNWsamfs/include/samapi.h"

     int sam_load(ushort_t eq_number,  char  *vsn,  char  *media,
     uint_t slot, int partition, int wait_response);

DESCRIPTION
```

sam_load() requests that the media be loaded from slot:partition or media.vsn into device eq_number. The device must be in the unavailable state (see set_state(1M)) and controlled by a media changer. If device eq_number already has media loaded, it will be unloaded and the media put away before the new media is loaded.

The call will return immediately after issuing the command if zero is specified for wait_response value. Other values for wait_response will give undefined results.

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

RETURN VALUES

Upon succesful completion a value of 0 is returned. Otherwise, a value of -1 is returned and errno is set to indicate the error.

ERRORS

sam_load() fails if one or more of the following are true:

ER_DEVICE_NOT_READY The specified eq_number device is not ready.

ER_DEVICE_NOT_UNAVAILABLE
The specified eq_number device must be in the "unavailable" state (see set_state(1M))

ER_DEVICE_OFF_OR_DOWN
The specified eq_number device is "off" or "down" and must be in the "unavailable" state (see set_state(1M))

ER_DEVICE_USE_BY_ANOTHER
The specified eq_number device is busy

and is being used by another process.

ER_FIFO_PATH_LENGTH The path and filename for the FIFO pipe is too long.

ER_NO_EQUIP_ORDINAL eq_number is not a defined equipment number in this configuration.

ER_NO_MASTER_SHM No Sun QFS or SAM-QFS master shared memory segment defined. Check that the Sun QFS or SAM-QFS file systems are mounted.

ER_NO_MASTER_SHM_ATT
No Sun QFS or SAM-QFS master shared memory segment found. Check that the Sun QFS or SAM-QFS file systems are mounted.

```
     ER_NO_RESPONSE_FIFO Unable to create the response FIFO pipe.

     ER_NO_STAT_ROBOT_CATALOG
                         The  robot  media  changer  catalog  for
                         equipment  number  eq_number  cannot  be
                         accessed for status.

     ER_NOT_VALID_SLOT_NUMBER
                         The  specified  ea  is  not  valid   for
                         eq_number device.

     ER_NOT_REMOV_MEDIA_DEVICE
                         The specified eq_number device is not  a
                         removable media device.

     ER_OPERATOR_NOT_PRIV
                         Operator does  not  have  permission  to
                         load removable media.

     ER_ROBOT_CATALOG_MISSING
                         The  robot  media  changer  catalog  for
                         equipment  number  eq_number  is missing
                         and a full audit is required.

     ER_ROBOT_DEVICE_REQUIRED
                         Equipment  number   eq_number   is   not
                         defined as a robotic device.

     ER_SLOT_IS_CLEAN_CARTRIDGE
                         ea contains a cleaning cartridge.

     ER_SLOT_NOT_OCCUPIED
                         ea does not contain any removable media.

     ER_SLOT_OR_VSN_REQUIRED
                         Either a ea in the media  changer  or  a
                         vsn must be specified.

     ER_UNABLE_TO_MAP_CATALOG
                         The  catalog  for  the  removable  media
                         changer at equipment number eq_number is
                         unable to be mapped into memory.

     ER_VSN_NOT_FOUND_IN_ROBOT
                         The specified vsn cannot be found in the
                         robotic   media   changer  at  equipment
                         number eq_number.

FILES
     mcf                 The configuration file for Sun  QFS  and
                         SAM-QFS file systems.

SEE ALSO
     load(1M), set_state(1M), sam-robotsd(1M), unload(1M).

     sam_unload(3).

     mcf(4).
```

sam_lstat(3)

```
NAME
     sam_stat, sam_lstat, sam_segment_stat - Gets file or segment
     status

SYNOPSIS
     cc [flag ...] file ...  -L/opt/SUNWsamfs/lib
     -R/opt/SUNWsamfs/lib -lsam [library ...]

     #include "/opt/SUNWsamfs/include/stat.h"

     int sam_stat(const char *path, struct sam_stat *buf, size_t
     bufsize);

     int sam_lstat(const char *path, struct sam_stat *buf, size_t
     bufsize);

     int sam_segment_stat(const char *path, struct sam_stat *buf,
     size_t bufsize);

AVAILABILITY
     SUNWqfs
     SUNWsamfs

DESCRIPTION
     The sam_stat() function returns file system attributes for
     the file to which path points.  The sam_segment_stat()
     function works with segmented files.  It returns attributes
     for the file segments to which path points.

     The sam_lstat() function returns file attributes similar to
     sam_stat().  The difference is that if file is a symbolic
     link, sam_lstat() returns information about the link, while
     sam_stat() returns information about the file or the file's
     segments that the link references.

     If these functions succeed, they write file attributes to
     the structure, or to the array of structures, to which buf
     points.  If they are returning information about a segmented
     file, they write information about the first file segment to
     the first structure in the array of structures.  They write
     information about the second file segment to the second
     structure in the array of structures, etc.

     Note that when sam_stat() and sam_lstat() are executed on a
     segmented file, the functions return information about the
     index inode.

     The sam_stat and sam_lstat functions are supported in Sun
     QFS and SAM-QFS environments.  The sam_segment_stat function
     is supported in Sun QFS and SAM-QFS environments.

OPTIONS
     These functions accept the following arguments:

     path      Specifies the path to the file.  This is the file
               or segmented file for which the file status is to
```

be obtained. Read, write, or execute permission of the named file is not required, but all directories listed in the path leading to the file must be searchable.

buf — Specifies a pointer to a structure into which information is placed concerning the file. The functions use one sam_stat structure from this argument for each single file or file segment. The length of buf, in bytes, must be sized as follows:

```
bytes =
number_of_segments * sizeof(struct sam_stat)
```

The number_of_segments is 1 for a nonsegmented file (used by sam_stat and sam_lstat). The number_of_segments is greater than 1 for a segmented file (used by sam_segment_stat).

For an unsegmented file, buf must be a sam_struct structure.

For a segmented file, buf must be an array of sam_struct structures.

bufsize — Specifies the length of the user's buffer, in bytes, to which buf points.

STRUCTURE CONTENTS

Table 1 and Table 2 show the content of the structure pointed to by buf.

TABLE 1. Members of struct sam_stat That Contain POSIX Standard File Attributes

Data Type	Field Name	Description
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	Numeric user ID of the file's owner
ulong_t	st_gid	Numeric 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

The following list describes Table 1's fields in more detail.

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 can be returned only 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 numeric user ID of the file's owner.
st_gid	The numeric 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.

TABLE 2. Members of struct sam_stat That Contain Sun QFS and SAM-QFS File Attributes

Data Type	Field Name	Description
uint_t	old_attr	Backward compatible, see attr
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]	Array of archive copy information
uchar_t	cs_algo	Checksum algorithm indicator
uchar_t	flags	Flags: staging, stage err, etc.
uchar_t	stripe_width	Stripe width set by setfa -s or -h
uchar_t	stripe_group	Stripe group set by setfa -g or -o
ulong_t	gen	Inode generation number
ulong_t	partial_size	Partial size in kilobytes
dev_t	rdev	ID of device if S_IFBLK or S_IFCHR
u_longlong_t	st_blocks	Block count in 512 byte blocks
ulong_t	segment_size	Segment size in megabytes
ulong_t	segment_number	Number of this segment
uint_t	stage_ahead	Number of segment to stage ahead
uint_t	admin_id	admin ID; inherited from directory
uint_t	allocahead	Allocate ahead set by setfa -A
uint_t	obj_depth	Stripe depth (KB) set by setfa -v
u_longlong_t	csum_val[2]	128 checksum value
time_t	rperiod_start_time	Time WORM retention period started
uint_t	rperiod_duration	WORM retention period duration
u_longlong_t	attr	File attributes

The following list describes Table 2's fields in more detail.

```
     attr      Attributes assigned to the file by Sun QFS and
               SAM-QFS functions and operations.

     attribute_time
               Time when the Sun QFS and SAM-QFS attributes last
               changed.  Changed by the following functions:
               sam_archive, sam_release, and sam_stage.  Also
               changed by the 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 the algorithm that is used when
               calculating the data verification value (checksum)
               for the file.  For more information, see ssum(1).

     flags     Flags containing miscellaneous additional
               information about the file.  Includes a bit that
               indicates that a stage is pending or is in
               progress on the file.  Also includes a bit that
               indicates 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
     The sam_stat() and sam_lstat() functions fail if one or more
     of the following are true:

     EACCES              Search permission is denied for a
                         component of the path prefix.

     EFAULT              Either buf or path points to an illegal
                         address.

     EINTR               A signal was caught during sam_stat() or
                         sam_lstat() function processing.

     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 path
                         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 to which buf points.
```

EXAMPLES

This example uses sam_segment_stat to obtain the status of a segmented file.

```
     struct sam_stat file_info;
     struct sam_stat *data_seg_info_ptr;
     int number_of_data_segments;
     int result;

     /*
      * Initialize file_info to be all zero bits:
      */
     memset((void *) "file_info, 0, sizeof(struct sam_stat));

     /*
      * Stat the file using the regular sam_stat function:
      */
     result = sam_stat(path, "file_info, sizeof(struct sam_stat));

     if (result != 0) {
         fprintf(stderr, "Error failed to sam stat the file, %s.\n", path);
         exit -70;
     }

     if (SS_ISSEGMENT_F(file_info.attr)) {
         /*
          * File is segmented, how many data segments does it have?
          */

         /*
          * Determine how many complete (full) segments it has:
          */
         number_of_data_segments = file_info.st_size /
                                   (file_info.segment_size * 1048576);

         /*
          * Determine if it has one data segment that isn't "full":
          */
         if (file_info.st_size "gt;
             number_of_data_segments * file_info.segment_size * 1048576) {
             number_of_data_segments++;
         }
     } else {
         /*
          * File isn't segmented
          */
         number_of_data_segments = 1;
     }

     /*
      * Allocate enough memory to hold all of the stat information for each
      * data segment:
      */
     data_seg_info_ptr = (struct sam_stat *) malloc(number_of_data_segments *
                                                   sizeof(struct sam_stat));

     if (data_seg_info_ptr == NULL) {
         fprintf(stderr, "Error failed to allocate memory for data segment stat operation.\n");
         exit -80;
     }

     /*
      * Initialize file_info to be all zero bits:
      */
     memset((void *) data_seg_info_ptr, 0, number_of_data_segments *
                                          sizeof(struct sam_stat));

     if (SS_ISSEGMENT_F(file_info.attr)) {
         /*
          * Use sam_segment_stat to get the stat information for all of the
          * data segments of the file.
          */
         result = sam_segment_stat(path, data_seg_info_ptr,
                                                   number_of_data_segments *
                                                   sizeof(struct sam_stat));
     } else {
         /*
          * File is not segmented, just use the stat information from the
          * sam_stat call
          */
         memcpy((void *) data_seg_info_ptr, (void *)file_info, sizeof(struct sam_stat));
     }

     if (!SS_ISSEGMENT_F(file_info.attr)) {
         number_of_data_segments = 1;
         data_seg_info_ptr = "file_info_ptr;
     }

     /*
      * data_seg_info_ptr now points to an array of sam_stat structures.
      * There is one sam_stat structure for each data segment and they are
      * indexed 0 through  number_of_data_segments - 1.
      *
      * Do not forget to deallocate the memory buffer pointed to by
      * data_seg_info_ptr using free.
      */
```

SEE ALSO
     ssum(1).

     mknod(2), stat(2).

sam_mig_create_file(3)

```
NAME
     sam_mig_create_file - Creates an offline SAM-QFS  file  from
     foreign media

SYNOPSIS
     cc [ flag  ... ] file  ...  -L/opt/SUNWsamfs/lib  -lsamut  [library ... ]

     #include "/opt/SUNWsamfs/include/mig.h"
     #include "/opt/SUNWsamfs/include/stat.h"

     int sam_mig_create_file(char *path, struct sam_stat *buf);

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     sam_mig_create_file() creates an offline Sun QFS or  SAM-QFS
     file  from  a  foreign  (non-  Sun  QFS  or  SAM-QFS) media.
     sam_mig_create_file() creates an offline file using informa-
     tion  supplied  by  a  foreign  data migration program.  The
     information used to identify the location  of  the  file  is
     stored in the file inode in the archive record.

     Note that the program calling this function  is  responsible
     for  creating all directories in the path before calling the
     function.

     path is the pathname to the file to be created.  It  may  be
     an  absolute or relative pathname but must be no longer than
     PATH_MAX (see the /usr/include/limits.h file).

     buf is a sam_stat structure (see sam_stat(3)).

     The following members in  the  sam_stat  structure  must  be
     filled in.  All other fields are ignored.

     ulong_t       st_mode      /* File mode (see mknod(2)) */
     ulong_t       st_uid       /* User ID of the file's owner */
     ulong_t       st_gid       /* Group ID of the file's owner */
     u_longlong_t  st_size      /* File size in bytes */
     ulong_t       st_atime     /* Time of last access      */
     ulong_t       st_ctime     /* Time of last file status change */
     ulong_t       st_mtime     /* Time of last data modification  */

     These members in the sam_copy_s structure  for  the  desired
     copy  (copy[] part of the sam_stat structure) must be filled
     in:
     u_longlong_t  position;       /* Any 8 bytes */
     time_t        creation_time;  /* The time the archive file is created */
     uint_t        offset;         /* Any 4 bytes */
     char          vsn[32];        /* Any 31 characters */
     char          media[2];       /* 2nd character of media type (must be 'z') */
     char          media[3];       /* 3rd character of media type */
                                   /* (must be a digit or lowercase alpha)*/

     position  Any 8 bytes  that  the  3rd  party  media  program
```

```
                    requires.

          creation_time
                    This is the time that the archive  was  made.   If
                    creation_time is zero, it will be set to the value
                    of time().

          offset    Any 4 bytes  that  the  3rd  party  media  program
                    requires.

          vsn       This is any 31  characters.   The  32nd  character
                    must  be a zero byte.  Other utilities may require
                    this to be a valid VSN.

          media     The second character of the  two  character  media
                    type.   If this field is zero, then this copy does
                    not contain any archive information  and  will  be
                    ignored.  At least one of the entries must contain
                    information.  Upon succesful creation of a file  a
                    value  of 0 is returned.  Otherwise, a value of -1
                    is returned and  errno  is  set  to  indicate  the
                    error.

     FILES
          /opt/SUNWsamfs/migkit/mig_build_cd.c
                              The example Migration Toolkit program.

          /etc/opt/SUNWsamfs/mcf
                              The configuration file for Sun  QFS  and
                              SAM-QFS file systems.

     SEE ALSO
          sam_stat(3).
```

sam_mig_mount_media(3)

```
     NAME
          sam_mig_mount_media - Queues mount request for media

     SYNOPSIS
          cc [ flag  ... ] file  ...  -L/opt/SUNWsamfs/lib  -lsamut  [library ... ]

          #include "/opt/SUNWsamfs/include/mig.h"

          char *sam_mig_mount_media(char *vsn, char *media);

     AVAILABILITY
          SUNWsamfs

     DESCRIPTION
          sam_mig_mount_media() queues a mount request for  the  media
          with  the volume serial number or barcode vsn and media type
          media.  sam_mig_mount_media() returns a pointer to the path-
          name  of  the  device  where  the  media is mounted.  A null
          pointer will be returned if the  media  cannot  be  mounted.
          This pointer will actually be a symbolic link to the device.
```

The symbolic link will be deleted when the reservation for the device has expired or is released with sam_mig_release_device. The daemon will wait for the device to close before releasing the device.

RETURN VALUES

Upon succesful completion a pointer to the pathname of the device is returned. Otherwise, a value of 0 is returned and errno is set to indicate the error.

FILES

/etc/opt/SUNWsamfs/mcf
The configuration file for Sun QFS or SAM-QFS

NOTE

Note that the media type passed to sam_mig_mount_media must be the media type as shown in the catalog entry for the VSN. It must not begin with "z".

SEE ALSO

mcf(4)

sam_mig_rearchive(3)

NAME

sam_mig_rearchive - Sets rearchive flag on files residing on foreign media

SYNOPSIS

```
cc [ flag  ... ] file  ...  -L/opt/SUNWsamfs/lib  -lsamut  [library ... ]

#include "/opt/SUNWsamfs/include/mig.h"

int sam_mig_rearchive(char *mount_point, char  **vsns,  char
*media);
```

AVAILABILITY

SUNWsamfs

DESCRIPTION

sam_mig_rearchive() is a routine that traverses a file system and marks archive files residing on a foreign medium as needing to be rearchived. This allows a site the ability to migrate files from the foreign media to Sun QFS or SAM-QFS media in a controlled fashion.

mount_point is the mount point of the Sun QFS or SAM-QFS file system to scan. vsns is a NULL terminated list pointing to the VSNs to be searched. media is the two character string containing the foreign media type.

RETURN VALUES

Upon succesful initialization a value of 0 is returned. Otherwise, a value of 1 is returned and errno is set to indicate the error.

```
FILES
     /opt/SUNWsamfs/migkit/mig_rearch.c
                         The example Migration Toolkit program.

     /etc/opt/SUNWsamfs/mcf
                         The configuration file for Sun  QFS  and
                         SAM-QFS file systems.

SEE ALSO
     mcf(4).
```

sam_mig_release_device(3)

```
NAME
     sam_mig_release_device - Releases a device

SYNOPSIS
     cc [ flag  ... ] file  ...  -L/opt/SUNWsamfs/lib  -lsamut  [library ... ]

     #include "/opt/SUNWsamfs/include/mig.h"

     int sam_mig_release_device(char *device);

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     sam_mig_release_device() releases the  reservation  for  the
     device  returned  from  sam_mig_mount_media.  The drive will
     enter the pool of available drives and  the  media  will  be
     unloaded  when  the  drive is needed or the idle_unload time
     has expired (see defaults.conf(4)).

RETURN VALUES
     Upon succesful completion a value of 0 is returned.   Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.

FILES
     /etc/opt/SUNWsamfs/mcf
                         The configuration file for Sun  QFS  and
                         SAM-QFS file systems.

SEE ALSO
     defaults.conf(4).
```

sam_mig_stage_end(3)

```
NAME
     sam_mig_stage_end - Completes staging function  for  foreign
     data migration program

SYNOPSIS
     cc [ flag  ... ] file  ...  -L/opt/SUNWsamfs/lib  -lsamut  [library ... ]

     #include "/opt/SUNWsamfs/include/mig.h"

     int sam_mig_stage_end(tp_stage_t *stage_req, int error);

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     sam_mig_stage_end() is called when the foreign  data  migra-
     tion  program  has  finished  the stage or detected an error
     after sam_mig_stage_file  has  been  called  without  error.
     stage_req is the number of this stage request.  error is the
     number of the error to pass back to the file  system.   A  0
     indicates no error.

RETURN VALUES
     Upon succesful completion a value of 0 is returned.

FILES
     /opt/SUNWsamfs/migkit/mig_cd.c
                         The example Migration Toolkit program.

     /etc/opt/SUNWsamfs/mcf
                         The configuration file for Sun  QFS  and
                         SAM-QFS file systems.

SEE ALSO
     sam_mig_stage_file(3).
```

sam_mig_stage_error(3)

```
NAME
     sam_mig_stage_error - Passes errors to the file system

SYNOPSIS
     cc [ flag  ... ] file  ...  -L/opt/SUNWsamfs/lib  -lsamut  [
     library ... ]

     #include "/opt/SUNWsamfs/include/mig.h"

     int sam_mig_stage_error(tp_stage_t *stage_req, int error);

AVAILABILITY
     SUNWsamfs

DESCRIPTION
```

```
     This function is used to pass error to the file  system  for
     the  stage  request associated with stage_req. This ends all
     activity for this stage request.

RETURN VALUES
     Upon succesful completion a value of 0 is returned.

FILES
     /opt/SUNWsamfs/migkit/mig_cd.c
                         The example Migration Toolkit program.

     /etc/opt/SUNWsamfs/mcf
                         The configuration file for Sun  QFS  and
                         SAM-QFS file systems.

SEE ALSO
     mcf(4).
```

sam_mig_stage_file(3)

```
NAME
     sam_mig_stage_file  -  Stages  function  from  foreign  data
     migration program

SYNOPSIS
     cc [ flag  ... ] file  ...  -L/opt/SUNWsamfs/lib  -lsamut  [
     library ... ]

     #include "/opt/SUNWsamfs/include/mig.h"

     int sam_mig_stage_file(tp_stage_t *stage_req);

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     sam_mig_stage_file() is called when the foreign data  migra-
     tion  program  is  ready  to  start staging the data for the
     stage request associated with stage_req.

RETURN VALUES
     sam_mig_stage_file() returns a 0 if the file system is ready
     for  usage.   Otherwise,  a value of -1 is returned and ends
     all activity for this stage.  errno is set to  indicate  the
     error.

FILES
     /opt/SUNWsamfs/migkit/mig_cd.c
                         The example Migration Toolkit program.

     /etc/opt/SUNWsamfs/mcf
                         The configuration file for Sun  QFS  and
                         SAM-QFS file systems.

SEE ALSO
     mcf(4).
```

sam_mig_stage_write(3)

```
NAME
     sam_mig_stage_write - Stages data from foreign  data  migra-
     tion program

SYNOPSIS
     cc [ flag  ... ] file  ...  -L/opt/SUNWsamfs/lib  -lsamut  [
     library ... ]

     #include "/opt/SUNWsamfs/include/mig.h"

     int sam_mig_stage_write(tp_stage_t *stage_req, char *buffer,
     int len, offset_t offset);

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     sam_mig_stage_write() passes  data  from  the  foreign  data
     migration  program  to the file system for the stage associ-
     ated with stage_req (see sam_mig_stage_file(3)).   stage_req
     is the number of this stage request.  buffer is a pointer to
     the data that needs to be transferred.  len is the number of
     bytes  of  data  to transfer.  offset is the offset from the
     beginning of this stage request.  This  is  not  the  offset
     from the beginning of the file (keep in mind stage_never).

RETURN VALUES
     sam_mig_stage_write returns the actual number of bytes writ-
     ten.   Otherwise, a value of -1 is returned.  If an error is
     returned, sam_mig_stage_end should  still  be  called.   The
     only  function  allowed  on  stage_req  after  an  error  is
     sam_mig_stage_end.

FILES
     /opt/SUNWsamfs/migkit/mig_cd.c
                         The example Migration Toolkit program

     /etc/opt/SUNWsamfs/mcf
                         The configuration file for  Sun  QFS  or
                         SAM-QFS

SEE ALSO
     sam_mig_stage_end(3), sam_mig_stage_file(3)
```

sam_move(3)

```
NAME
     sam_move - Move media in a robotic media changer

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib  -lsamapi  [
     library ... ]
```

```
#include "/opt/SUNWsamfs/include/samapi.h"

int  sam_move(ushort_t  eq_number,  uint_t  src_ea,   uint_t
dest_ea, int wait_response);
```

DESCRIPTION

sam_move() requests that the media in the source element address src_ea be moved to the destination element address dest_ea in the robotic media changer at equipment number eq_number. The source element address src_ea must be in use and occupied (that is, the media is not mounted). The destination element address dest_ea must not be occupied or in use. Some robotic media changers do not support moving media between storage element addresses.

The call will return immediately after issuing the command if zero is specified for wait_response value. Other values for wait_response will give undefined results.

RETURN VALUES

Upon succesful completion a value of 0 is returned. Otherwise, a value of -1 is returned and errno is set to indicate the error.

ERRORS

sam_move() fails if one or more of the following are true:

ER_DEVICE_NOT_READY	The specified eq_number device is not ready.
ER_DEVICE_USE_BY_ANOTHER	The specified eq_number device is busy and is being used by another process.
ER_FIFO_PATH_LENGTH	The path and filename for the FIFO pipe is too long.
ER_NO_EQUIP_ORDINAL	eq_number is not a defined equipment number in this configuration.
ER_NO_MASTER_SHM	No Sun QFS or SAM-QFS master shared memory segment defined. Check that the Sun QFS or SAM-QFS file systems are mounted.
ER_NO_MASTER_SHM_ATT	No Sun QFS or SAM-QFS master shared memory segment found. Check that the Sun QFS or SAM-QFS file systems are mounted.
ER_NO_RESPONSE_FIFO	Unable to create the response FIFO pipe.
ER_NO_STAT_ROBOT_CATALOG	The robot media changer catalog for equipment number eq_number cannot be accessed for status.
ER_NOT_VALID_DEST_SLOT_NO	

```
                        The specified destination element
                        address dest_ea is not valid for
                        eq_number device.

     ER_NOT_VALID_SLOT_NUMBER
                        The specified source element address
                        src_ea is not valid for eq_number dev-
                        ice.

     ER_OPERATOR_NOT_PRIV
                        Operator does not have permission to
                        move removable media.

     ER_ROBOT_CATALOG_MISSING
                        The robot media changer catalog for
                        equipment number eq_number is missing
                        and a full audit is required.

     ER_ROBOT_DEVICE_REQUIRED
                        Equipment number eq_number is not
                        defined as a robotic device.

     ER_ROBOT_NO_MOVE_SUPPORT
                        Robotic media changer at equipment
                        number eq_number does not support move-
                        ment of media between element addresses.

     ER_DST_SLOT_IS_OCCUPIED
                        dest_ea already contains removable
                        media.

     ER_SLOT_NOT_OCCUPIED
                        ea does not contain any removable media.

     ER_DST_SLOT_NOT_AVAIL_MOVE
                        dest_ea is not available for the move.

     ER_SRC_SLOT_NOT_AVAIL_MOVE

                        src_ea is not available for the move.

     ER_UNABLE_TO_MAP_CATALOG
                        The catalog for the removable media
                        changer at equipment number eq_number is
                        unable to be mapped into memory.

FILES
     mcf                The configuration file for Sun QFS and
                        SAM-QFS

SEE ALSO
     export(1M), import(1M), sam-robotsd(1M).

     sam_export(3), sam_import(3).

     mcf(4).
```

sam_odlabel(3)

```
NAME
     sam_odlabel - Label an optical disk on the specified device

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib  -lsamapi  [
     library ... ]

     #include "/opt/SUNWsamfs/include/samapi.h"

     int  sam_odlabel(ushort_t  eq_number,  char  *new_vsn,  char
     *old_vsn,  uint_t  ea,  int  modifier,  char  *use_info, int
     erase, int wait_response);

DESCRIPTION
     sam_odlabel() labels an optical disk on the specified device
     with  equipment  number  eq_number  and,  if the device is a
     robotic media changer, a ea must be specified.   If  old_vsn
     is specified as a NULL pointer, the media will be assumed to
     be not labeled and a new label will be written.   A  new_vsn
     must  be specified.  A VSN must be one to thirty-one charac-
     ters in length.  A VOL (volume) and a PAR (partition)  label
     are written.  These labels conform to ISO standard IEC13346.
     The data portion follow ISO standard TC97SC23.

     If erase is specified as nonzero, the  media  is  completely
     erased before a label is written.

     The call will return immediately after issuing  the  command
     if  zero is specified for wait_response value.  Other values
     for wait_response will give undefined results.

RETURN VALUES
     Upon succesful completion a value of 0 is returned.   Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.

ERRORS
     sam_odlabel() fails if one or  more  of  the  following  are
     true:

     ER_BLOCK_SIZE_TOO_LARGE
                         The specified block_size is greater than
                         the maximum block size allowed.

     ER_DEVICE_NOT_LABELED
                         The specified eq_number device is not  a
                         labeled device.

     ER_DEVICE_NOT_MANUAL_LOAD
                         The specified eq_number device is not  a
                         manual load type device.

     ER_DEVICE_NOT_THIS_TYPE
                         The specified eq_number  device  is  not
                         the correct media type.
     ER_FIFO_PATH_LENGTH The path and filename for the FIFO  pipe
                         is too long.

     ER_INVALID_MEDIA_TYPE
                         Invalid  media  type  specified  to   be
                         labeled.

     ER_INVALID_U_INFO_LENGTH
                         use_info must be less than  128  charac-
                         ters in length.

     ER_INVALID_VSN_LENGTH
                         The  specified  new_vsn  or  old_vsn  is
                         greater than 31 characters in length.

     ER_MEDIA_VSN_NOT_OLD_VSN
                         The old_vsn does not match  the  current
                         VSN on the media.

     ER_NO_EQUIP_ORDINAL eq_number is  not  a  defined  equipment
                         number in this configuration.

     ER_NO_MASTER_SHM    No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment defined.  Check that the
                         Sun QFS  or  SAM-QFS  file  systems  are
                         mounted.

     ER_NO_RESPONSE_FIFO Unable to create the response FIFO pipe.

     ER_NO_MASTER_SHM_ATT
                         No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment  found.   Check that the
                         Sun QFS  or  SAM-QFS  file  systems  are
                         mounted.

     ER_NO_STAT_ROBOT_CATALOG
                         The  robot  media  changer  catalog  for
                         equipment  number  eq_number  cannot  be
                         accessed for status.

     ER_NOT_VALID_SLOT_NUMBER
                         ea specified  is  not  a  valid  element
                         address  number  of  the  robotic  media
                         changer.

     ER_OLD_VSN_NOT_UNK_MEDIA
                         old_vsn not matching unknown media VSN.

     ER_OPERATOR_NOT_PRIV
                         Operator does  not  have  permission  to
                         label removable media.

     ER_ROBOT_CATALOG_MISSING
                         No robot catalog was found for equipment
                         number  eq_number  which is defined as a
                         robotic media changer.

     ER_ROBOT_DEVICE_REQUIRED
                         No devices were found to be defined  for
                         equipment   number  eq_number  which  is
                         defined as a robotic media changer.

     ER_SLOT_NOT_OCCUPIED
                         No media was found to occupy the element
                         address  in  the media changer at equip-
                         ment number eq_number

     ER_VSN_BARCODE_REQUIRED
                         new_vsn must be specified.

     ER_UNABLE_TO_MAP_CATALOG
                         The  catalog  for  the  removable  media
                         changer at equipment number eq_number is
                         unable to be mapped into memory.

FILES
     mcf                 The configuration file for  Sun  QFS  or
                         SAM-QFS environments.

SEE ALSO
     odlabel(1M), tplabel(1M).

     sam_tplabel(3).
```

sam_opencat(3)

```
NAME
     sam_opencat - Accesses an  automated  library’s  catalog  to
     read entries

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [
     library ... ]

     #include "/opt/SUNWsamfs/include/catalog.h"

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

AVAILABILITY
     32-bit programs only

DESCRIPTION
     sam_opencat() initiates  access  to  the  automated  library
     catalog pointed to by path.  The string which path points to
     is limited to 127  characters.   It  returns  a  sam_cat_tbl
     structure  in  the  area  pointed to by buf . bufsize is the
     length of the user’s  buffer  to  which  buf  points.   This
     should   be   equal   to   or   greater  than  sizeof(struct
     sam_cat_tbl).

     The user may have access to at most MAX_CAT catalogs at  any
     one time.

     The contents of a sam_cat_tbl structure include the  follow-
     ing members:

          /* catalog table */
          time_t   audit_time;  /* Audit time */
          int      version;     /* Catalog version number */
          int      count;       /* Number of slots */
          char     media[4];    /* Media type, if entire catalog is one */

     Following the call to sam_opencat(), entries in the  library
     catalog are obtained using sam_getcatalog().

RETURN VALUES
     Upon successful completion, a catalog "handle" is  returned,
     which is an integer equal to or greater than zero.

     This   "handle"   is   used   on   subsequent    calls    to
     sam_getcatalog()  to  specify  the catalog to access, and is
     also used by sam_closecat() to deallocate the  "handle"  and
     end access to the catalog.

     If the call  to  sam_opencat()  fails,  a  value  of  -1  is
     returned and errno is set to indicate the error.

ERRORS
     sam_opencat() fails if one or more of  the  following  error
     conditions are true:

     EMFILE              The user already has access  to  MAX_CAT
                         catalogs  ,  or the process has too many
                         open files.

     EINVAL              bufsize is set to an invalid  value,  or
                         either path or buf is a null pointer.

     ER_UNABLE_TO_INIT_CATALOG
                         This process was  unable  to  initialize
                         the catalog data.

     ENOENT              There is no active catalog file with the
                         name given.

SEE ALSO
     sam_closecat(3), sam_getcatalog(3)
```

sam_readrminfo(3)

```
NAME
     sam_readrminfo - Gets removable media file status

SYNOPSIS
     cc  [  flag    ...  ]   file      ...   -L/opt/SUNWsamfs/lib
     -R/opt/SUNWsamfs/lib -lsam [ library ... ]

     #include "/opt/SUNWsamfs/include/rminfo.h"

     int sam_readrminfo(const char *path, struct sam_rminfo *buf,
          size_t bufsize);
```

DESCRIPTION

sam_readrminfo() returns information about a removable media file. The removable media file is pointed to by path.

buf is a pointer to a sam_rminfo() structure into which information is placed concerning the file.

bufsize is the length of the user's buffer to which buf points. This should be equal to or greater than sizeof(struct sam_rminfo). The maximum number of overflow VSNs is 256. The following macro can be used to calculate the size of the sam_rminfo structure for n VSNs.

```
#define SAM_RMINFO_SIZE(n) (sizeof(struct sam_rminfo) + ((n)
- 1) * sizeof(struct sam_section))
```

The contents of the structure pointed to by buf is documented in sam_request(3).

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of -1 is returned and errno is set to indicate the error.

ERRORS

sam_readrminfo() fails if one or more of the following are true:

EACCES	Search permission is denied for a component of the path prefix.
EFAULT	buf or path points to an illegal address.
EINTR	A signal was caught during the sam_readrminfo() function.
ELOOP	Too many symbolic links were encountered in translating path.
EMULTIHOP	Components of path require hopping to multiple remote machines and the file system does not allow it.
ENAMETOOLONG	The length of the path argument exceeds {PATH_MAX}, or the length of a path component exceeds {NAME_MAX} while {_POSIX_NO_TRUNC} is in effect.
ENOENT	The named file does not exist or is the null pathname.
ENOLINK	path points to a remote machine and the link to that machine is no longer active.
ENOTDIR	A component of the path prefix is not a

```
                         directory.

     EOVERFLOW           A component is too large to store in the
                         structure pointed to by buf.

SEE ALSO
     sam_request(3)
```

sam_rearch(3)

```
NAME
     sam_rearch - Sets rearchive attributes on a file  or  direc-
     tory

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

     #include "/opt/SUNWsamfs/include/lib.h"

     int sam_rearch(const char *path, int num_opts, ... );

DESCRIPTION
     sam_rearch() sets rearchive attributes on a file  or  direc-
     tory  using a Sun Storage Archive Manager system call.  path
     is the file on which to set the attributes,  followed  by  a
     sequence  of  num_opts input characters or options.  Indivi-
     dual options are described below.

OPTIONS
     c copy_no
          Specifies the archive copy number.  If one or more  'c'
          options are specified, only those archive copies (1, 2,
          3 or 4) are marked. If not specified,  the  default  is
          all copies only in the case that media type and VSN are
          specified, using the  "m  media"  option  and  "v  vsn"
          option.

     M    Rearchives metadata only.  This  includes  directories,
          the  segment  index, and removable media files. Regular
          files and symbolic links are not rearchived.

     m media
          Specifies the media type.  If specified, archive copies
          on the specified media are marked.  This option must be
          specified in conjunction with the "v vsn" option.   For
          more  information  on  media  types, see the mcf(4) man
          page.

     o    Requires the file to be online before its archive entry
          is  rearchived.   If  the file is offline, the function
          stages  the  file  onto  disk  before  rearchiving  any
          entries.

     v vsn
          Marks archive copies on VSN vsn for rearchiving.   This
          option  must  be  specified  in conjunction with the "m
          media" option.

RETURN VALUES
     Upon successful completion a value of 0 is returned.  Other-
     wise, a value of -1 is returned and errno is set to indicate

     the error.

ERRORS
     sam_rearch() 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 superuser.

     EFAULT              Argument points to an illegal address.

     EINTR               A   signal   was   caught   during   the
                         sam_rearch() 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 com-
                         ponent    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
     rearch(1m), mcf(4)
```

sam_release(3)

```
NAME
     sam_rearch - Sets rearchive attributes on a file  or  direc-
     tory

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

     #include "/opt/SUNWsamfs/include/lib.h"

     int sam_rearch(const char *path, int num_opts, ... );
```

```
DESCRIPTION
     sam_rearch() sets rearchive attributes on a file  or  direc-
     tory  using a Sun Storage Archive Manager system call.  path
     is the file on which to set the attributes,  followed  by  a
     sequence  of  num_opts input characters or options.  Indivi-
     dual options are described below.

OPTIONS
     c copy_no
          Specifies the archive copy number.  If one or more  'c'
          options are specified, only those archive copies (1, 2,
          3 or 4) are marked. If not specified,  the  default  is
          all copies only in the case that media type and VSN are
          specified, using the  "m  media"  option  and  "v  vsn"
          option.

     M    Rearchives metadata only.  This  includes  directories,
          the  segment  index, and removable media files. Regular
          files and symbolic links are not rearchived.

     m media
          Specifies the media type.  If specified, archive copies
          on the specified media are marked.  This option must be
          specified in conjunction with the "v vsn" option.   For
          more  information  on  media  types, see the mcf(4) man
          page.

     o    Requires the file to be online before its archive entry
          is  rearchived.   If  the file is offline, the function
          stages  the  file  onto  disk  before  rearchiving  any
          entries.

     v vsn
          Marks archive copies on VSN vsn for rearchiving.   This
          option  must  be  specified  in conjunction with the "m
          media" option.

RETURN VALUES
     Upon successful completion a value of 0 is returned.  Other-
     wise, a value of -1 is returned and errno is set to indicate

     the error.

ERRORS
     sam_rearch() 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 superuser.

     EFAULT              Argument points to an illegal address.

     EINTR               A   signal   was   caught   during   the
                         sam_rearch() 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 com-
                         ponent    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
     rearch(1m), mcf(4)
```

sam_request(3)

```
NAME
     sam_request - Creates a removable media file

SYNOPSIS
     cc  [  flag    ...  ]   file      ...   -L/opt/SUNWsamfs/lib
     -R/opt/SUNWsamfs/lib -lsam [ library ... ]

     #include "/opt/SUNWsamfs/include/rminfo.h"

     int sam_request(const char *path,  struct  sam_rminfo  *buf,
     size_t bufsize);

DESCRIPTION
     sam_request() creates a removable media file allowing access
     to  either tape or optical disk. The removable media file is
     pointed to by path.

     buf is a pointer to  a  sam_rminfo()  structure  into  which
     information is placed concerning the file.

     bufsize is the length of the  user's  buffer  to  which  buf
     points.    This    should   be   equal  to  or  greater  than
     sizeof(struct sam_rminfo).  The maximum number  of  overflow
     VSNs  is  256.  The following macro can be used to calculate
     the size of the sam_rminfo structure for n VSNs.

     #define SAM_RMINFO_SIZE(n) (sizeof(struct sam_rminfo) + ((n)
     - 1) * sizeof(struct sam_section))

     The contents of the structure pointed to by buf include  the
     following members:

                                    /* POSIX rminfo structure. */
     ushort_t       flags;          /* File mode (see mknod(2)) */
     char           media[4];       /* Media type */
```

```
ulong_t        creation_time;   /* Creation time of removable media file */
uint_t         block_size;      /* Block size of file in bytes */
U_longlong_t   position;        /* Position of file on the removable media */
U_longlong_t   required_size;   /* Required size for optical only */

                                /* optical information only. */
char           file_id[32];     /* File identifier */
int            version;         /* Version number */
char           owner_id[32];    /* Owner identifier */
char           group_id[32];    /* Group identifier */
char           info[160];       /* Information */

                                /* all media information. */
short          n_vsns;          /* Number of vsns containing file */
short          c_vsn;           /* Current vsn ordinal -- returned */
struct sam_section  section[1];   /* VSN array - n_vsns entries */
```

```
flags     The access flags for the file.

          RI_blockio uses block I/O for data transfers. Each
          request buffer is a block on the device.

          RI_bufio uses buffered I/O for data transfers. The
          block size is defined by block_size.

          RI_foreign uses block I/O for data transfers.  The
          tape is not written by SAM-QFS, is barcoded, write
          protected, and is opened for read access only.

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  opti-
          cal file label.

group_id  The file’s group ID. It is written into the  opti-
          cal file label.
```

```
     info      The file's  optional  information  field.   It  is
               written into the optical file label by .

     n_vsns    Specified the number of removable media cartridges
               containing the file.

     c_vsn     Specifies the current removable media ordinal.

     sam_section

               Specifies the array of removable media cartridges.
               The contents of the sam_section structure includes
               the following members:

                              /* POSIX sam_section structure. */
     char          vsn[32];   /* Volume serial name */
     U_longlong_t  length;    /* Length of this section in bytes */
     U_longlong_t  position;  /* Position of this section */
     U_longlong_t  offset;    /* Byte offset of this section */

RETURN VALUES
     Upon successful completion a value of 0 is returned.  Other-
     wise, 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  com-
                         ponent 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 com-
                         ponent    exceeds    {NAME_MAX}    while
                         {_POSIX_NO_TRUNC} is in effect.

     ENOENT              The named file does not exist or is  the
                         null pathname.

     ENOLINK             path points to a remote machine and  the
                         link  to  that  machine  is  no  longer
                         active.

     ENOTDIR             A component of the path prefix is not  a
                         directory.
     EOVERFLOW           A component is too large to store in the

                         structure pointed to by buf.

     ENOTSUP             License does not support foreign tapes.

SEE ALSO
     request(1).

     mcf(4).
```

sam_restore_copy(3)

```
NAME
     sam_restore_copy - Creates an archive copy for a Sun QFS or
     SAM-QFS file

SYNOPSIS
     cc [ flag ... ] file ...  -L/opt/SUNWsamfs/lib
     -lsam [ library ... ]

     #include "/opt/SUNWsamfs/include/stat.h"

     int sam_restore_copy(const char *path, int copy, struct
     sam_stat *buf, size_t bufsize, struct sam_section *vbuf,
     size_t vbufsize);

DESCRIPTION
     The sam_restore_copy() library routine creates an archive
     copy for an existing Sun QFS or SAM-QFS file.  The file must
     already exist and the archive copy must not exist.  The
     sam_restore_copy() library routine creates an archive copy
     using information supplied by the user and obtained from a
     source such as the archiver.log.  This library routine
     accepts the following arguments:

     path      The path name to the file where the archive copy
               is being created.  It may be an absolute or
               relative path name, but it must be no longer than
               PATH_MAX (see the /usr/include/limits.h file).

     copy      The copy number (0, 1, 2, or 3) of the archive
               copy that is being created.

     buf       A sam_stat structure.  See sam_stat(3).

     bufsize   The size of the sam_stat structure.  See
               sam_stat(3).

     vbuf      A sam_section structure for the array of VSNs if
               the archive copy overflowed volumes, that is,
               n_vsns > 1.  If n_vsns = 1, vbuf should be set to
               NULL.  See sam_stat(3).

     vbufsize  The size of the sam_section structure.  If n_vsns
```

```
                    = 1, vbufsize should be set to 0.  See
                    sam_stat(3).

          The following members in the sam_stat structure must exist.
          All other fields are ignored.

          ulong_t      st_mode      /* File mode (see mknod(2)) */
          ulong_t      st_uid       /* User ID of the file's owner */
          ulong_t      st_gid       /* Group ID of the file's owner */
          u_longlong_t st_size      /* File size in bytes */

          ulong_t      st_atime     /* Time of last access      */
          ulong_t      st_ctime     /* Time of last file status change */
          ulong_t      st_mtime     /* Time of last data modification  */

          The following members in the sam_copy_s structure must exist
          for the copy.  All other fields are ignored.

          u_longlong_t position;    /* Position of the file on the media. */
          time_t       creation_time; /* Time the archive copy was created */
          uint_t       offset;      /* Location of the copy in the archive file */
          short        n_vsns;      /* Number of volumes used by the archive */
          char         media[4];    /* Two character media type. */
          char         vsn[32];     /* Volume serial name of the media volume */

          The preceding fields have the following meaning:

          position  The position of the file recorded on the media.

          creation_time
                    This is the time that the archive was made.  If
                    creation_time is zero, it is set to the value of
                    time().

          offset    The location of the copy in the archive file in
                    units of 512 bytes.

          n_vsns    The number of volumes that this copy spans.

          vsn       The volume serial name of the cartridge where the
                    file resides.

          media     The two-character media type.  For example, the
                    media type for DLT tape is lt.  See mcf(4).

     RETURN VALUES
          Upon succesful creation of a file, a value of 0 is returned.
          Otherwise, a negative value is returned and errno is set to
          indicate the error.  The possible return values are:
          -1    user is not root
          -2    invalid copy number
          -3    invalid VSN
          -4    file does not exist
          -5    file open failed
          -6    uid and gid do not match those for the existing file
          -7    invalid VSN for this copy
          -8    multiple copies but vbufsize incorrect
          -9    media type invalid
          -10   copy restore failed for some other reason
```

```
FILES
     /etc/opt/SUNWsamfs/mcf
                         The configuration file for Sun QFS or

                         SAM-QFS file systems.

SEE ALSO
     sam_restore_file(3), sam_stat(3).

     mcf(4).
```

sam_restore_file(3)

```
NAME
     sam_restore_file - Creates an offline Sun QFS file.

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

     #include "/opt/SUNWsamfs/include/stat.h"

     int sam_restore_file(const char *path, struct sam_stat *buf,
     size_t bufsize);

DESCRIPTION
     sam_restore_file() creates an offline file in a Sun  QFS  or
     SAM-QFS  file system.  sam_restore_file() creates an offline
     file using information supplied by  the  user  and  obtained
     from  a source such as the archiver.log file.  The file must
     not exist.

     Note that the program calling this function  is  responsible
     for  creating all directories in the path before calling the
     function.

     path is the pathname to the file to be created.  It  may  be
     an  absolute or relative pathname but must be no longer than
     PATH_MAX (see the /usr/include/limits.h file).

     buf is a sam_stat(3) structure (see sam_stat(3)).

     bufsize is  the  size  of  the  sam_stat(3)  structure  (see
     sam_stat(3)).

     The following members  in  the  sam_stat(3)  structure  must
     exist.  All other fields are ignored.

          ulong_t      st_mode      /* File mode (see mknod(2)) */
          ulong_t      st_uid       /* User ID of the file's owner */
          ulong_t      st_gid       /* Group ID of the file's owner */
          u_longlong_t st_size      /* File size in bytes */
          ulong_t      st_atime     /* Time of last access      */
          ulong_t      st_ctime     /* Time of last file status change */
          ulong_t      st_mtime     /* Time of last data modification  */
     The following members in the sam_copy_s structure must exist
     for all copies, if any. All other fields are ignored.

          char         media[4];    /* Two character media type. */
          u_longlong_t position;    /* Position of the file on the media. */
          uint_t        offset;   /* Loc of copy in archive file in 512 bytes */
          time_t       creation_time; /* Time the archive copy was created */
          char         vsn[32];     /* Volume serial name of the media */

     position  The position of the file recorded on the media.

     offset    The location of this copy in the archive  file  in
               512 bytes.

     creation_time
               This is the time that the archive  was  made.   If
               creation_time is zero, it will be set to the value
               of time().

     vsn       The volume serial name of the cartridge where  the
               file resides.

     media     The two character  media  type.  See  mcf(4).  For
               example, the media type for DLT tape is lt.

RETURN VALUES
     Upon succesful creation of a file a value of 0 is  returned.
     Otherwise,  a negative value is returned and errno is set to
     indicate the error.  The possible return values are:
     -1   user is not root
     -2   invalid media type
     -3   invalid VSN
     -5   file does not exist
     -6   restore failed for some other reason

FILES
     sam_stat(3).
```

sam_segment(3)

```
NAME
     sam_segment - Sets segment attributes on a file or directory

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

     #include "/opt/SUNWsamfs/include/lib.h"

     int sam_segment(const char *path, const char *ops);

DESCRIPTION
     sam_segment() sets segment attributes on a file or directory
     using  a  Sun QFS or SAM-QFS system call.  If a file is seg-
     mented, it is archived and staged in  segment  size  chunks.
     sam_segment() is not supported on a SAM-QFS shared file sys-
     tem.  path is the file on which to set the attributes.   ops
```

is the character string of options, for example: "dl104857600". Individual options are described below.

OPTIONS

d   Return the segment file attributes on the file to the default, i.e. reset to the file access instead of segment access. It not possible to reset a file that has already been segmented. When this option is specified, the attributes are reset to the default. If it is used, it should be the first character in the string.

l n Specifies the segment size in units of bytes. The segment_size must be greater than or equal to one megabyte. This segment size is the size at which the file will be segmented for purposes of archiving and staging. An error is returned if the file is greater than the segment size.

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

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of -1 is returned and errno is set to indicate the error.

ERRORS

sam_segment() fails if one or more of the following are true:

EINVAL	An invalid option was specified, or the file is neither a regular file nor a directory. The file exceeds the specified segment size.
EPERM	Not the owner or superuser.
EFAULT	path or ops points to an illegal address.
EINTR	A signal was caught during the sam_segment() function.
ELOOP	Too many symbolic links were encountered in translating path.
EMULTIHOP	Components of path require hopping to multiple remote machines and the file system does not allow it.
ENAMETOOLONG	The length of the path argument exceeds {PATH_MAX}, or the length of a path component exceeds {NAME_MAX} while {_POSIX_NO_TRUNC} is in effect.

ENOENT	The named file does not exist or is the null pathname.
ENOLINK	path points to a remote machine and the link to that machine is no longer active.
ENOTDIR	A component of the path prefix is not a directory.
ENOTSUP	License does not support segment.

SEE ALSO
     segment(1)

sam_segment_stat(3)

NAME
     sam_stat, sam_lstat, sam_segment_stat - Gets file or segment status

SYNOPSIS
```
     cc [flag ...] file ...  -L/opt/SUNWsamfs/lib
     -R/opt/SUNWsamfs/lib -lsam [library ...]

     #include "/opt/SUNWsamfs/include/stat.h"

     int sam_stat(const char *path, struct sam_stat *buf, size_t
     bufsize);

     int sam_lstat(const char *path, struct sam_stat *buf, size_t
     bufsize);

     int sam_segment_stat(const char *path, struct sam_stat *buf,
     size_t bufsize);
```

AVAILABILITY
     SUNWqfs
     SUNWsamfs

DESCRIPTION
     The sam_stat() function returns file system attributes for the file to which path points. The sam_segment_stat() function works with segmented files. It returns attributes for the file segments to which path points.

     The sam_lstat() function returns file attributes similar to sam_stat(). The difference is that if file is a symbolic link, sam_lstat() returns information about the link, while sam_stat() returns information about the file or the file's segments that the link references.

     If these functions succeed, they write file attributes to the structure, or to the array of structures, to which buf points. If they are returning information about a segmented file, they write information about the first file segment to the first structure in the array of structures. They write information about the second file segment to the second structure in the array of structures, etc.

Note that when sam_stat() and sam_lstat() are executed on a segmented file, the functions return information about the index inode.

The sam_stat and sam_lstat functions are supported in Sun QFS and SAM-QFS environments. The sam_segment_stat function is supported in Sun QFS and SAM-QFS environments.

OPTIONS

These functions accept the following arguments:

path	Specifies the path to the file. This is the file or segmented file for which the file status is to be obtained. Read, write, or execute permission of the named file is not required, but all directories listed in the path leading to the file must be searchable.
buf	Specifies a pointer to a structure into which information is placed concerning the file. The functions use one sam_stat structure from this argument for each single file or file segment. The length of buf, in bytes, must be sized as follows:

```
bytes =
number_of_segments * sizeof(struct sam_stat)
```

The number_of_segments is 1 for a nonsegmented file (used by sam_stat and sam_lstat). The number_of_segments is greater than 1 for a segmented file (used by sam_segment_stat).

For an unsegmented file, buf must be a sam_struct structure.

For a segmented file, buf must be an array of sam_struct structures.

bufsize	Specifies the length of the user's buffer, in bytes, to which buf points.

STRUCTURE CONTENTS

Table 1 and Table 2 show the content of the structure pointed to by buf.

TABLE 1. Members of struct sam_stat That Contain POSIX Standard File Attributes

Data Type	Field Name	Description
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	Numeric user ID of the file’s owner
ulong_t	st_gid	Numeric 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

The following list describes Table 1’s fields in more detail.

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 can be returned only 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 numeric user ID of the file’s owner.
st_gid	The numeric 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.

TABLE 2. Members of struct sam_stat That Contain Sun QFS and SAM-QFS File Attributes

Data Type	Field Name	Description
uint_t	old_attr	Backward compatible, see attr
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]	Array of archive copy information
uchar_t	cs_algo	Checksum algorithm indicator
uchar_t	flags	Flags: staging, stage err, etc.
uchar_t	stripe_width	Stripe width set by setfa -s or -h
uchar_t	stripe_group	Stripe group set by setfa -g or -o
ulong_t	gen	Inode generation number

ulong_t	partial_size	Partial size in kilobytes
dev_t	rdev	ID of device if S_IFBLK or S_IFCHR
u_longlong_t	st_blocks	Block count in 512 byte blocks
ulong_t	segment_size	Segment size in megabytes
ulong_t	segment_number	Number of this segment
uint_t	stage_ahead	Number of segment to stage ahead
uint_t	admin_id	admin ID; inherited from directory
uint_t	allocahead	Allocate ahead set by setfa -A
uint_t	obj_depth	Stripe depth (KB) set by setfa -v
u_longlong_t	csum_val[2]	128 checksum value
time_t	rperiod_start_time	Time WORM retention period started
uint_t	rperiod_duration	WORM retention period duration
u_longlong_t	attr	File attributes

The following list describes Table 2's fields in more detail.

attr — Attributes assigned to the file by Sun QFS and SAM-QFS functions and operations.

attribute_time
Time when the Sun QFS and SAM-QFS attributes last changed. Changed by the following functions: sam_archive, sam_release, and sam_stage. Also changed by the 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 the algorithm that is used when calculating the data verification value (checksum) for the file. For more information, see ssum(1).

flags — Flags containing miscellaneous additional information about the file. Includes a bit that indicates that a stage is pending or is in progress on the file. Also includes a bit that indicates 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

The sam_stat() and sam_lstat() functions fail if one or more of the following are true:

EACCES	Search permission is denied for a component of the path prefix.

EFAULT	Either buf or path points to an illegal address.
EINTR	A signal was caught during sam_stat() or sam_lstat() function processing.
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 path 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 to which buf points.

EXAMPLES

This example uses sam_segment_stat to obtain the status of a segmented file.

```
struct sam_stat file_info;
struct sam_stat *data_seg_info_ptr;
int number_of_data_segments;
int result;

/*
 * Initialize file_info to be all zero bits:
 */
memset((void *) "file_info, 0, sizeof(struct sam_stat));

/*
 * Stat the file using the regular sam_stat function:
 */
result = sam_stat(path, "file_info, sizeof(struct sam_stat));

if (result != 0) {
    fprintf(stderr, "Error failed to sam stat the file, %s.\n", path);
    exit -70;
}

if (SS_ISSEGMENT_F(file_info.attr)) {
    /*
     * File is segmented, how many data segments does it have?
```

```
         */

        /*
         * Determine how many complete (full) segments it has:
         */
        number_of_data_segments = file_info.st_size /
                                  (file_info.segment_size * 1048576);

        /*
         * Determine if it has one data segment that isn’t "full":
         */
        if (file_info.st_size >
            number_of_data_segments * file_info.segment_size * 1048576) {
            number_of_data_segments++;
        }
    } else {
        /*
         * File isn’t segmented
         */
        number_of_data_segments = 1;
    }

    /*
     * Allocate enough memory to hold all of the stat information for each
     * data segment:
     */
    data_seg_info_ptr = (struct sam_stat *) malloc(number_of_data_segments *
                                               sizeof(struct sam_stat));

    if (data_seg_info_ptr == NULL) {
        fprintf(stderr, "Error failed to allocate memory for data segment stat operation.\n");
        exit -80;
    }

    /*
     * Initialize file_info to be all zero bits:
     */
    memset((void *) data_seg_info_ptr, 0, number_of_data_segments *
                                      sizeof(struct sam_stat));

    if (SS_ISSEGMENT_F(file_info.attr)) {
        /*
         * Use sam_segment_stat to get the stat information for all of the
         * data segments of the file.
         */
        result = sam_segment_stat(path, data_seg_info_ptr,
                                               number_of_data_segments *
                                               sizeof(struct sam_stat));
    } else {
        /*
         * File is not segmented, just use the stat information from the
         * sam_stat call
         */
        memcpy((void *) data_seg_info_ptr, (void *)file_info, sizeof(struct sam_stat));
    }

    if (!SS_ISSEGMENT_F(file_info.attr)) {
        number_of_data_segments = 1;
        data_seg_info_ptr = "file_info_ptr;
    }

    /*
     * data_seg_info_ptr now points to an array of sam_stat structures.
     * There is one sam_stat structure for each data segment and they are
     * indexed 0 through  number_of_data_segments - 1.
     *
     * Do not forget to deallocate the memory buffer pointed to by
     * data_seg_info_ptr using free.
     */
```

SEE ALSO
     ssum(1).

     mknod(2), stat(2).

sam_segment_vsn_stat(3)

NAME

sam_vsn_stat, sam_segment_vsn_stat - Gets VSN status for an archive copy that overflows VSNs

SYNOPSIS

```
cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

#include </opt/SUNWsamfs/include/stat.h>

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

int sam_segment_vsn_stat(const char  *path,  int  copy,  int
segment_ord, struct sam_section *buf, size_t bufsize);
```

DESCRIPTION

The sam_vsn_stat() function obtains information about the VSNs for the archive copy indicated by copy of path, where path points to a non-segmented file.

If sam_vsn_stat() is called and path points to a segmented file, then VSN information about the archive copy copy of the segmented file's index inode is returned.

The sam_segment_vsn_stat() function obtains information about the VSNs for the archive copy indicated by copy of the data segment indicated by segment_ord of the segmented file pointed to by path.

sam_vsn_stat() and sam_segment_vsn_stat() obtain information about the VSNs for the indicated archive copy when the indicated archive copy uses multiple VSNs.

sam_vsn_stat() and sam_segment_vsn_stat() fail if called to obtain VSN stat information for an archive copy that only uses one VSN. Use the sam_stat() or sam_segment_stat() subroutines to determine the number of VSNs used by a given archive copy and to get VSN information for archive copies that only use one VSN.

sam_vsn_stat() places VSN information for all of the sections that comprise the overflowed archive copy into buf.

Read, write, or execute permission of the named file is not required, but all directories listed in the path name leading to the file must be searchable.

copy is the archive copy number (0, 1, 2 or 3).

segment_ord is the data segment number (0, ..., n_segs - 1) where n_segs is the current number of data segments that

comprise the file pointed to by path.

buf is a pointer to a sam_section structure into which VSN information is placed concerning the file's archive copy.

bufsize is the length of the user's buffer to which buf points. sam_vsn_stat and sam_segment_vsn_stat place VSN information for each overflowed section that comprises the archive copy into buf. Hence, bufsize should be at least sizeof(struct sam_vsn_stat) * n_vsns bytes, where n_vsns is the number of VSNs used by the archived copy.

The contents of the structure pointed to by buf include the following struct sam_section members:

```
     char         vsn[32];
     u_longlong_t length;
     u_longlong_t position;
     u_longlong_t offset;
```

vsn	The VSN of the section. This is a null-terminated string with a maximum of 31 characters.
length	The length of the section on the volume.
position	The position of the start of the archive file that contains this section.
offset	The offset of this file on the archive file.

RETURN VALUES

Upon successful completion, a value of 0 is returned. Otherwise, a value of -1 is returned and errno is set to indicate the error.

ERRORS

sam_vsn_stat() and sam_segment_vsn_stat() 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_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 com-
                        ponent    exceeds    {NAME_MAX}    while
                        {_POSIX_NO_TRUNC} is in effect.

     ENOENT             The named file does not exist or is  the
                        null pathname.

     ENOLINK            path points to a remote machine and  the
                        link  to  that  machine  is  no  longer
                        active.

     ENOTDIR            A component of the path prefix is not  a
                        directory.

     EOVERFLOW          A component is too large to store in the
                        structure pointed to by buf.

USAGE
     sam_vsn_stat         Call sam_stat to  get  the  number  of
                          VSNs  used  for the archive copy.  The
                          call to sam_stat will write the number
                          of  VSNs  used  by the archive copy in
                          your struct  sam_stat  buffer  in  the
                          member   copy[copy].n_vsns.    If  the
                          archive copy uses only  one  VSN  (the
                          number  of  VSNs is 1), then your pro-
                          gram or script must retrieve  the  VSN
                          information  for the archive copy from
                          the copy member of the sam_stat struc-
                          ture that was filled in when your pro-
                          gram or script called  sam_stat.   The
                          copy  member of the sam_stat structure
                          is of type struct sam_copy_s.

     sam_segment_vsn_stat  Call sam_stat to determine whether the
                          file pointed to by path is segmented.

                          If the file pointed to by path is  not
                          segmented,  then  use  sam_vsn_stat to
                          obtain  VSN  information  as  detailed
                          above.

                          If the file pointed to by path is seg-
                          mented,  then call sam_segment_stat to
                          get the number of VSNs  used  for  the
                          archive  copy indicated by copy of the
                          data segment indicated by segment_ord.
```

```
                              The  call  to  sam_segment_stat   will
                              write  the  number of VSNs used by the
                              archive copy  of  the  indicated  data
                              segment  in  your  array  of  sam_stat
                              structures in the  member  located  in
                              sam_stat_buff_array[segment_ord].copy[copy].n_vsns.

                              If the archive copy uses only one  VSN
                              (the  number  of VSNs is 1), then your
                              program or script  must  retrieve  the
                              VSN  information  for the archive copy
                              from the copy member of the element in
                              the  array of sam_stat structures that
                              was filled in  when  your  program  or
                              script  called  sam_segment_stat.  The
                              copy member of the sam_stat  structure
                              is  of  type  struct sam_copy_s and is
                              found in the array of sam_stat  struc-
                              tures under the index segment_ord.

     A struct sam_copy_s structure has the following members:

          u_longlong_t position;
          time_t       creation_time;
          uint_t       offset;
          ushort_t     flags;
          short        n_vsns;
          char         media[4];
          char         vsn[32];

     position       Location of the archive file

     creation_time  Time that the archive copy was created

     offset         Location of the copy in the archive file

     flags          Sun  QFS  and  SAM-QFS  archive  copy  status
                    flags.   These  indicate  whether the archive
                    copy has been made,  is  stale,  is  damaged,
                    etc.   See  /opt/SUNWsamfs/include/stat.h for
                    bit masks which can be applied to these flags
                    to  resolve  the  state  and  status  of  the
                    archive copy.

     n_vsns         Number of VSNs used  by  the  archived  copy.
                    Will  be  1  in  case of no overflow, will be
                    greater than one if the  archive  copy  over-
                    flows volumes.

     media          Media type.  This is a null-terminated string
                    with a maximum of 3 characters.

     vsn            The VSN  of  the  copy.   This  is   a  null-
                    terminated  string with a maximum of 31 char-
                    acters.

     If the archive copy uses more than one VSN  (the  number  of
     VSNs  is  greater  than 1), then your program or script must
     call sam_vsn_stat or sam_segment_vsn_stat  to  retrieve  the
```

VSN information for all of the sections that comprise the archive copy.

Do not call sam_vsn_stat or sam_segment_vsn_stat if the archive copy uses only one VSN (does not overflow).

SEE ALSO

sam_stat(3)

NOTES

The Sun QFS and SAM-QFS file systems permit a maximum of MAX_VOLUMES sections per archive copy. Hence, instead of dynamically allocating a buffer of structures, a more efficient method is to to declare a static array with MAX_VOLUMES number of elements.

The constant MAX_VOLUMES is declared in the following include file: /opt/SUNWsamfs/include/rminfo.h .

sam_set_fs_contig(3)

NAME

sam_set_fs_contig - Sets the maximum number of contiguous blocks for I/O

SYNOPSIS

```
cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib  -lsamapi  [library ... ]

#include "/opt/SUNWsamfs/include/samapi.h"

int sam_set_fs_contig(ushort_t  eq_number,  int  max_contig,
int wait_response);
```

DESCRIPTION

sam_set_fs_contig() sets the maximum number of contiguous blocks that can be read or written to a mass storage device in the family set at equipment number eq_number. The value for max_contig can be from 1 to 128. The default value is 8. The block size is 16384 bytes.

The call will return immediately after issuing the command if zero is specified for wait_response value. Other values for wait_response will give undefined results.

RETURN VALUES

Upon succesful completion a value of 0 is returned. Otherwise, a value of -1 is returned and errno is set to indicate the error.

ERRORS

sam_set_fs_contig() fails if one or more of the following are true:

ER_DEVICE_NOT_CORRECT_TYPE
: The specified eq_number device is not a family set device.

```
     ER_FIFO_PATH_LENGTH The path and filename for the FIFO  pipe
                         is too long.

     ER_NO_EQUIP_ORDINAL eq_number is  not  a  defined  equipment
                         number in this configuration.

     ER_NO_MASTER_SHM    No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment defined.  Check that the
                         Sun QFS  or  SAM-QFS  file  systems  are
                         mounted.

     ER_NO_MASTER_SHM_ATT
                         No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment  found.   Check that the
                         Sun QFS  or  SAM-QFS  file  systems  are

                         mounted.

     ER_NO_RESPONSE_FIFO Unable to create the response FIFO pipe.

     ER_OPERATOR_NOT_PRIV
                         Operator does not have permission to set
                         maximum contiguous blocks.
```

FILES
```
     mcf                 The configuration file for  Sun  QFS  or
                         SAM-QFS file systems.
```

SEE ALSO

samu(1M).

sam_set_fs_thresh(3)

NAME

sam_set_fs_thresh - Sets file system thresholds for the releaser

SYNOPSIS
```
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib  -lsamapi  [library ... ]

     #include "/opt/SUNWsamfs/include/samapi.h"

     int sam_set_fs_thresh(ushort_t eq_number, int min_threshold,
     int max_threshold, int wait_response);
```

DESCRIPTION

sam_set_fs_thresh() sets the minimum (low) and maximum (high) thresholds which control the execution of the releaser for archived files in the family set at equipment number eq_number. Reaching the maximum threshold in the family set initiates the releaser, which releases file space until the minimum threshold is reached or no release candidates exist.

The call will return immediately after issuing the command

```
     if  zero is specified for wait_response value.  Other values
     for wait_response will give undefined results.

RETURN VALUES
     Upon succesful completion a value of 0 is returned.   Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.

ERRORS
     sam_set_fs_thresh() fails if one or more  of  the  following
     are true:

     ER_DEVICE_NOT_CORRECT_TYPE
                         The specified eq_number device is not  a
                         family set device.

     ER_FIFO_PATH_LENGTH The path and filename for the FIFO  pipe
                         is too long.

     ER_NO_EQUIP_ORDINAL eq_number is  not  a  defined  equipment
                         number in this configuration.

     ER_NO_MASTER_SHM    No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment defined.  Check that the
                         Sun QFS  or  SAM-QFS  file  systems  are
                         mounted.

     ER_NO_MASTER_SHM_ATT
                         No Sun  QFS  or  SAM-QFS  master  shared

                         memory  segment  found.   Check that the
                         Sun QFS  or  SAM-QFS  file  systems  are
                         mounted.

     ER_NO_RESPONSE_FIFO Unable to create the response FIFO pipe.

FILES
     mcf                 The configuration file for Sun  QFS  and
                         SAM-QFS file systems.

SEE ALSO
     samu(1M).
```

sam_set_state(3)

```
NAME
     sam_set_state - Sets the new state  for  a  removable  media
     device

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib  -lsamapi  [library ... ]

     #include "/opt/SUNWsamfs/include/devstat.h"
     #include "/opt/SUNWsamfs/include/samapi.h"

     int sam_set_state(ushort_t  eq_number,  dstate_t  new_state,
     int wait_response);

DESCRIPTION
     sam_set_state()  sets  the  device   at   equipment   number
     eq_number to the specified state enumeration value.  The set
     of values are:

          typedef enum dstate{
              DEV_ON,                /* Normal operations */
              DEV_RO,                /* Read only operations */
              DEV_IDLE,              /* No new opens allowed */
              DEV_UNAVAIL,           /* Unavaiable for file system */
              DEV_OFF,               /* Off to this machine */
              DEV_DOWN               /* Maintenance use only */
          }dstate_t;

     Depending on the current state, only certain new states  can
     be set:

          Current State   Possible Next State
          DEV_ON          DEV_IDLE, DEV_OFF
          DEV_IDLE        Automatically goes to OFF when IDLE
          DEV_OFF         DEV_DOWN, DEV_ON
          DEV_DOWN        DEV_OFF

     The call will return immediately after issuing  the  command
     if  zero is specified for wait_response value.  Other values
     for wait_response will give undefined results.

RETURN VALUES
     Upon succesful completion a value of 0 is returned.   Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.

ERRORS
     sam_set_state() fails if one or more of  the  following  are
     true:

     ER_DEVICE_DOWN_NEW_STATE
                         If a device state is down, the new state
                         for  the specified eq_number device must
                         be off.

     ER_DEVICE_USE_BY_ANOTHER
                         The specified eq_number device is in use
                         by another process.

     ER_FIFO_PATH_LENGTH The path and filename for the FIFO  pipe
                         is too long.

     ER_NO_EQUIP_ORDINAL eq_number is  not  a  defined  equipment
                         number in this configuration.

     ER_INVALID_STATE_SPECIFIED
                         Specified state is not a valid value.

     ER_NO_MASTER_SHM    No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment defined.  Check that the
                         Sun QFS  or  SAM-QFS  file  systems  are
```

```
                          mounted.

     ER_NO_MASTER_SHM_ATT
                          No Sun  QFS  or  SAM-QFS  master  shared
                          memory  segment  found.   Check that the
                          Sun QFS  or  SAM-QFS  file  systems  are
                          mounted.

     ER_NO_RESPONSE_FIFO Unable to create the response FIFO pipe.

     ER_NOT_REMOV_MEDIA_DEVICE
                          The   device   with   equipment   number
                          eq_number  is not a removable media dev-
                          ice.

     ER_OPERATOR_NOT_PRIV
                          Operator does not have permission to set
                          the state for devices.

FILES
     mcf                  The configuration file for Sun  QFS  and
                          SAM-QFS file systems.

SEE ALSO
     samu(1M).
```

sam_setfa(3)

```
NAME
     sam_setfa - Sets attributes on a file or directory

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

     #include "/opt/SUNWsamfs/include/lib.h"

     int sam_setfa(const char *path, const char *ops);

DESCRIPTION
     sam_setfa() sets attributes on a file or directory  using  a
     SAM-QFS  system  call.  path is the file on which to set the
     attributes.  ops is the character  string  of  options,  for
     example:  "ds1".  Individual options are described below.

OPTIONS
     A n  Specifies the number of bytes to be allocated ahead  of
          a write to the file.  The n must be an integer and must
          be greater than or equal to one kilobyte and less  than
          4  terabytes.   The n is rounded down to units of kilo-
          bytes.  This option is only valid for a  regular  file.
          This  option  should  be  used when writing large files
          where more sequential  allocation  is  desired.   Note,
          when  the  file  is  closed the blocks are reset to the
          size of the file.

     B    Specifies  the  direct  I/O  attribute  be  permanently
          cleared  for this file.  This means data is transferred
          indirectly between the user's buffer and  disk  through
          the  system's page cache.  The default I/O mode is buf-
          fered (uses the page cache).  The directio attribute is
          persistent,  remaining  until  specifically  cleared or
          reset.  See directio(3C) for Solaris 2.6 and above  for
          more details.

     d    Return the file attributes on the file to the  default,
          i.e.  the  stripe  is reset to the mount default.  When
          this option is specified, the attributes are  reset  to
          the  default.   If  it  is used, it should be the first
          character in the string.

     D    Specifies the direct I/O attribute be  permanently  set
          for this file.  This means data is transferred directly
          between the user's buffer  and  disk.   This  attribute
          should  only  be set for large block aligned sequential
          I/O.  The default I/O mode is buffered (uses  the  page
          cache).   Direct  I/O  will  not be used if the file is
          currently memory mapped.   The  directio  attribute  is
          persistent,  remaining  until  specifically  cleared or
          reset.  See directio(3C) for Solaris 2.6 and above  for
                     more details.

     g n  Specifies the number of the  striped  group  where  the
          file is to be preallocated.  n is a number 0 .. 127.  n
          must be a striped_group defined in the file system.

     l n  Specifies the number of bytes to be preallocated to the
          file.   The n must be an integer.  This option can only
          be applied to a regular file.  If an I/O event attempts
          to  extend  a  file preallocated with the L option, the
          caller receives an ENXIO error.  The l option allocates
          using  extent  allocation.   This means striping is not
          supported and the file is allocated on 1 disk device or
          1  striped  group.   The  L  and l options are mutually
          exclusive.  If the file has existing disk blocks,  this
          option is changed to the L option.

     L n  Specifies the number of bytes to be preallocated to the
          file.   The  n must be an integer.  This option is only
          valid for a regular file.  The L option allocates using
          standard allocation.  This means striping is supported.
          This also means the file can be extended.  The L and  l
          options are mutually exclusive.

     q    Specifies that this file will be linked to  the  pseudo
          character  device  driver,  samaio,  for the purpose of
          issuing asynchronous I/O.  Note, this option also  sets
          Direct  I/O and qwrite.  Setting this option may result
          in greater performance.

     s n  Specifies the number of allocation units  to  be  allo-
          cated  before  changing  to  the next unit.  If n is 1,
          this means the file will stripe across all units with 1
          disk  allocation  unit (DAU) allocated per unit.  If  n
          is 0, this means the file will be allocated on one unit
          until  that  unit  has no space.  The default stripe is
```

```
          specified at mount.  (see mount_samfs(1M)).  Note, EIN-
          VAL  is  returned  if  the  user  sets  stripe  > 0 and
          mismatched  stripe  groups  exist.   Mismatched  stripe
          groups  means  all  striped groups do not have the same
          number  of  partitions.   Striping  across   mismatched
          stripe groups is not allowed.

RETURN VALUES
     Upon successful completion a value of 0 is returned.  Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.

ERRORS
     sam_setfa() fails if one or more of the following are true:

     EINVAL              An invalid option was specified, or  the

                         file  is  neither  a  regular file nor a
                         directory.

     EPERM               Not the owner or superuser.

     EFAULT              path  or  ops  points  to   an   illegal
                         address.

     EINTR               A   signal   was   caught   during   the
                         sam_setfa() function.

     ELOOP               Too many symbolic links were encountered
                         in translating path.

     EMULTIHOP           Components of path  require  hopping  to
                         multiple  remote  machines  and the file
                         system does not allow it.

     ENAMETOOLONG        The length of the path argument  exceeds
                         {PATH_MAX}, or the length of a path com-
                         ponent    exceeds    {NAME_MAX}    while
                         {_POSIX_NO_TRUNC} is in effect.

     ENOENT              The named file does not exist or is  the
                         null pathname.

     ENOLINK             path points to a remote machine and  the
                         link   to  that  machine  is  no  longer
                         active.

     ENOTDIR             A component of the path prefix is not  a
                         directory.

SEE ALSO
     setfa(1), ssum(1).

     mount_samfs(1M).

     sam_advise(3), sam_ssum(3).

     directio(3C).
```

sam_settings(3)

```
NAME
     sam_settings - Obtains the Sun QFS or SAM-QFS  default  set-
     tings and system messages

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib  -lsamapi  [library ... ]

     #include "/opt/SUNWsamfs/include/samapi.h"

     int sam_settings(struct sam_def_values *defaults, int size);

DESCRIPTION
     sam_settings() obtains the default settings for the Sun  QFS
     or SAM-QFS environment and the current system messages being
     issued to the operator.

RETURN VALUES
     Upon succesful completion a value of 0 is returned.   Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.

ERRORS
     sam_settings() fails if one or more  of  the  following  are
     true:

     ER_NO_MASTER_SHM    No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment defined.  Check that the
                         Sun QFS  or  SAM-QFS  file  systems  are
                         mounted.

     ER_NO_MASTER_SHM_ATT
                         No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment  found.   Check that the
                         Sun QFS  or  SAM-QFS  file  systems  are
                         mounted.

     ER_STRUCTURE_TOO_SMALL
                         The input argument size does not specify
                         enough     space     to     contain    the
                         sam_def_values structure.

FILES
     mcf                 The configuration file for Sun  QFS  and
                         SAM-QFS file systems.

SEE ALSO
     samu(1M).
```

sam_ssum(3)

```
NAME
     sam_ssum - Sets checksum attributes on a file

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

     #include "/opt/SUNWsamfs/include/lib.h"

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

DESCRIPTION
     sam_ssum() sets the checksum attributes on a  file  using  a
     Sun  QFS  or SAM-QFS system call.  path is the file on which
     to set the attributes.   ops  is  the  character  string  of
     options,   for   example:   "gu".   Individual  options  are
     described below.

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

     The generate attribute must be set  on  a  file  before  any
     archive  copy  has  been made.  Likewise, the selected algo-
     rithm 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 gen-
          erate  the 128-bit checksum value.  The simple checksum
          algorithm provided by Sun  Microsystems,  Inc.  is  the
          default  if  no algorithm is specified but the generate
          attribute is set.  n may be one of the following:

          0    Use no algorithm.

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

          128 or higher
               Site-specified algorithms.

          For example, a valid options string is  "gu1",  setting
          the  generate  and  use attributes, and specifying that
          the Sun-provided simple checksum algorithm be  used  to
          generate the value.

ERRORS
     sam_ssum() fails if one or more of the following are true:

     EINVAL              An invalid option was specified, or  the
                         file  is  neither  a  regular file nor a
                         directory.

     EPERM               Not the owner or superuser.

     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 com-
                         ponent    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), release(1), sls(1) ssum(1), stage(1).

     sam_archive(3), sam_release(3), sam_stage(3).
```

sam_stage(3)

```
NAME
     sam_stage - Sets stage attributes on a file

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

     #include "/opt/SUNWsamfs/include/lib.h"

     int sam_stage(const char *path, const char *ops);

DESCRIPTION
     sam_stage() sets stage attributes on  a  file  or  directory
     using  a Sun QFS or SAM-QFS system call. path is the file on
     which to set the attributes.  ops is the character string of
     options, for example:  dn.  Individual options are described
     below.

OPTIONS
     a    Sets the associative staging attribute on the  file  or
          directory. Associative staging is activated when a reg-
          ular 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 attri-
          bute 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 stage -n attribute is not
          supported on Sun QFS shared file  system  clients;  the
          entire file is staged when accessed on a client.

     p    Stage the partial blocks back online.

     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 com-
          pleting.  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 superuser.
ENXIO	No archive copy exists, or the specified archive copy does not exist.
EFAULT	path or ops points to an illegal address.
EINTR	A signal was caught during the sam_stage() function.
ELOOP	Too many symbolic links were encountered in translating path.
EMULTIHOP	Components of path require hopping to multiple remote machines and the file system does not allow it.
ENAMETOOLONG	The length of the path argument exceeds {PATH_MAX}, or the length of a path component exceeds {NAME_MAX} while {_POSIX_NO_TRUNC} is in effect.
ENOENT	The named file does not exist or is the null pathname.
ENOLINK	path points to a remote machine and the link to that machine is no longer active.
ENOTDIR	A component of the path prefix is not a directory.
ENOTSUP	License does not support option.

NOTE

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

SEE ALSO

stage(1), ssum(1).

sam-stagealld(1M), mount_samfs(1M).

mmap(2), write(2).

sam_ssum(3).

sam_stat(3)

NAME

sam_stat, sam_lstat, sam_segment_stat - Gets file or segment status

SYNOPSIS

```
cc [flag ...] file ...  -L/opt/SUNWsamfs/lib
-R/opt/SUNWsamfs/lib -lsam [library ...]

#include "/opt/SUNWsamfs/include/stat.h"

int sam_stat(const char *path, struct sam_stat *buf, size_t
bufsize);

int sam_lstat(const char *path, struct sam_stat *buf, size_t
bufsize);

int sam_segment_stat(const char *path, struct sam_stat *buf,
size_t bufsize);
```

AVAILABILITY

SUNWqfs
SUNWsamfs

DESCRIPTION

The sam_stat() function returns file system attributes for the file to which path points. The sam_segment_stat() function works with segmented files. It returns attributes for the file segments to which path points.

The sam_lstat() function returns file attributes similar to sam_stat(). The difference is that if file is a symbolic link, sam_lstat() returns information about the link, while sam_stat() returns information about the file or the file's segments that the link references.

If these functions succeed, they write file attributes to the structure, or to the array of structures, to which buf points. If they are returning information about a segmented file, they write information about the first file segment to the first structure in the array of structures. They write information about the second file segment to the second structure in the array of structures, etc.

Note that when sam_stat() and sam_lstat() are executed on a segmented file, the functions return information about the index inode.

The sam_stat and sam_lstat functions are supported in Sun

QFS and SAM-QFS environments. The sam_segment_stat function is supported in Sun QFS and SAM-QFS environments.

OPTIONS

These functions accept the following arguments:

path	Specifies the path to the file. This is the file or segmented file for which the file status is to be obtained. Read, write, or execute permission of the named file is not required, but all directories listed in the path leading to the file must be searchable.
buf	Specifies a pointer to a structure into which information is placed concerning the file. The functions use one sam_stat structure from this argument for each single file or file segment. The length of buf, in bytes, must be sized as follows: bytes = number_of_segments * sizeof(struct sam_stat) The number_of_segments is 1 for a nonsegmented file (used by sam_stat and sam_lstat). The number_of_segments is greater than 1 for a segmented file (used by sam_segment_stat). For an unsegmented file, buf must be a sam_struct structure. For a segmented file, buf must be an array of sam_struct structures.
bufsize	Specifies the length of the user’s buffer, in bytes, to which buf points.

STRUCTURE CONTENTS

Table 1 and Table 2 show the content of the structure pointed to by buf.

TABLE 1. Members of struct sam_stat That Contain POSIX Standard File Attributes

Data Type	Field Name	Description
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	Numeric user ID of the file’s owner
ulong_t	st_gid	Numeric 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

The following list describes Table 1’s fields in more detail.

```
     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 can be
               returned only 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 numeric user ID of the file’s owner.

     st_gid    The numeric 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.
```

TABLE 2. Members of struct sam_stat That Contain Sun QFS and SAM-QFS File Attributes

Data Type	Field Name	Description
uint_t	old_attr	Backward compatible, see attr
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]	Array of archive copy information
uchar_t	cs_algo	Checksum algorithm indicator
uchar_t	flags	Flags: staging, stage err, etc.
uchar_t	stripe_width	Stripe width set by setfa -s or -h
uchar_t	stripe_group	Stripe group set by setfa -g or -o
ulong_t	gen	Inode generation number
ulong_t	partial_size	Partial size in kilobytes
dev_t	rdev	ID of device if S_IFBLK or S_IFCHR
u_longlong_t	st_blocks	Block count in 512 byte blocks
ulong_t	segment_size	Segment size in megabytes
ulong_t	segment_number	Number of this segment
uint_t	stage_ahead	Number of segment to stage ahead
uint_t	admin_id	admin ID; inherited from directory
uint_t	allocahead	Allocate ahead set by setfa -A
uint_t	obj_depth	Stripe depth (KB) set by setfa -v

u_longlong_t	csum_val[2]	128 checksum value
time_t	rperiod_start_time	Time WORM retention period started
uint_t	rperiod_duration	WORM retention period duration
u_longlong_t	attr	File attributes

The following list describes Table 2's fields in more detail.

attr
: Attributes assigned to the file by Sun QFS and SAM-QFS functions and operations.

attribute_time
: Time when the Sun QFS and SAM-QFS attributes last changed. Changed by the following functions: sam_archive, sam_release, and sam_stage. Also changed by the 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 the algorithm that is used when calculating the data verification value (checksum) for the file. For more information, see ssum(1).

flags
: Flags containing miscellaneous additional information about the file. Includes a bit that indicates that a stage is pending or is in progress on the file. Also includes a bit that indicates 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

The sam_stat() and sam_lstat() functions fail if one or more of the following are true:

EACCES	Search permission is denied for a component of the path prefix.
EFAULT	Either buf or path points to an illegal address.
EINTR	A signal was caught during sam_stat() or sam_lstat() function processing.
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 path
                         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 to which buf points.

EXAMPLES
     This example uses sam_segment_stat to obtain the status of a
     segmented file.

     struct sam_stat file_info;
     struct sam_stat *data_seg_info_ptr;
     int number_of_data_segments;
     int result;

     /*
      * Initialize file_info to be all zero bits:
      */
     memset((void *) &file_info, 0, sizeof(struct sam_stat));

     /*
      * Stat the file using the regular sam_stat function:
      */
     result = sam_stat(path, &file_info, sizeof(struct sam_stat));

     if (result != 0) {
         fprintf(stderr, "Error failed to sam stat the file, %s.\n", path);
         exit -70;
     }

     if (SS_ISSEGMENT_F(file_info.attr)) {
         /*
          * File is segmented, how many data segments does it have?
          */

         /*
          * Determine how many complete (full) segments it has:
          */
         number_of_data_segments = file_info.st_size /
                                  (file_info.segment_size * 1048576);

         /*
          * Determine if it has one data segment that isn’t "full":
```

```
         */
        if (file_info.st_size >
            number_of_data_segments * file_info.segment_size * 1048576) {
            number_of_data_segments++;
        }
    } else {
        /*
         * File isn’t segmented
         */
        number_of_data_segments = 1;
    }

    /*
     * Allocate enough memory to hold all of the stat information for each
     * data segment:
     */
    data_seg_info_ptr = (struct sam_stat *) malloc(number_of_data_segments *
                                                   sizeof(struct sam_stat));

    if (data_seg_info_ptr == NULL) {
        fprintf(stderr, "Error failed to allocate memory for data segment stat operation.\n");
        exit -80;
    }

    /*
     * Initialize file_info to be all zero bits:
     */
    memset((void *) data_seg_info_ptr, 0, number_of_data_segments *
                                          sizeof(struct sam_stat));

    if (SS_ISSEGMENT_F(file_info.attr)) {
        /*
         * Use sam_segment_stat to get the stat information for all of the
         * data segments of the file.
         */
        result = sam_segment_stat(path, data_seg_info_ptr,
                                               number_of_data_segments *
                                               sizeof(struct sam_stat));
    } else {
        /*
         * File is not segmented, just use the stat information from the
         * sam_stat call
         */
        memcpy((void *) data_seg_info_ptr, (void *)file_info, sizeof(struct sam_stat));
    }

    if (!SS_ISSEGMENT_F(file_info.attr)) {
        number_of_data_segments = 1;
        data_seg_info_ptr = &file_info_ptr;
    }

    /*
     * data_seg_info_ptr now points to an array of sam_stat structures.
     * There is one sam_stat structure for each data segment and they are
     * indexed 0 through  number_of_data_segments - 1.
     *
     * Do not forget to deallocate the memory buffer pointed to by
     * data_seg_info_ptr using free.
     */
```

```
SEE ALSO
     ssum(1).

     mknod(2), stat(2).
```

sam_tplabel(3)

```
NAME
     sam_tplabel - Labels a tape on the specified device

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib  -lsamapi  [library ... ]

     #include "/opt/SUNWsamfs/include/samapi.h"

     int  sam_tplabel(ushort_t  eq_number,  char  *new_vsn,  char
     *old_vsn,  uint_t  ea,  int  modifier,  int  block_size, int
     erase, int wait_response);

DESCRIPTION
     sam_tplabel() labels a tape on  the  specified  device  with
     equipment  number  eq_number  and  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.

     If the device is a robotic  media  changer,  a  ea  must  be
     specified.   If  old_vsn is specified as a NULL pointer, the
     media will be assumed to be not labeled and a new label will
     be  written.   A new_vsn must be specified.  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:  !"%&'()*+,-
     ./:;<=>?_.

     block_size  specifies  the  blocksize  for  this  tape.   If
     nonzero, 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.

     If erase is specified as nonzero, the  media  is  completely
     erased before a label is written.

     The call will return immediately after issuing  the  command
     if  zero is specified for wait_response value.  Other values
     for wait_response will give undefined results.
```

RETURN VALUES

Upon succesful completion a value of 0 is returned.

Otherwise, a value of -1 is returned and errno is set to indicate the error.

ERRORS

sam_tplabel() fails if one or more of the following are true:

ER_BLOCK_SIZE_TOO_LARGE	The specified block_size is greater than the maximum block size allowed.
ER_DEVICE_NOT_LABELED	The specified eq_number device is not a labeled device.
ER_DEVICE_NOT_MANUAL_LOAD	The specified eq_number device is not a manual load type device.
ER_DEVICE_NOT_THIS_TYPE	The specified eq_number device is not the correct media type.
ER_FIFO_PATH_LENGTH	The path and filename for the FIFO pipe is too long.
ER_INVALID_BLOCK_SIZE	The specified block_size is not 16, 32, 64, 128, 256, 512, 1024 or 2048.
ER_INVALID_MEDIA_TYPE	Invalid media type specified to be labeled.
ER_INVALID_VSN_CHARACTERS	The specified new_vsn or old_vsn contains invalid characters to conform to ANSI X3.27-1987 File Structure and Labeling of Magnetic Tapes for Information Interchange.
ER_INVALID_VSN_LENGTH	The specified new_vsn or old_vsn is not from one to six characters in length.
ER_MEDIA_VSN_NOT_OLD_VSN	The old_vsn does not match the current VSN on the media.
ER_NO_EQUIP_ORDINAL	eq_number is not a defined equipment number in this configuration.
ER_NO_MASTER_SHM	No Sun QFS or SAM-QFS master shared memory segment defined. Check that the Sun QFS and SAM-QFS file systems are

```
                       mounted.

     ER_NO_RESPONSE_FIFO Unable to create the response FIFO pipe.

     ER_NO_MASTER_SHM_ATT
                       No Sun  QFS  or  SAM-QFS  master  shared
                       memory  segment  found.   Check that the
                       Sun QFS and  SAM-QFS  file  systems  are
                       mounted.

     ER_NO_STAT_ROBOT_CATALOG
                       The  robot  media  changer  catalog  for
                       equipment  number  eq_number  cannot  be
                       accessed for status.

     ER_NOT_VALID_SLOT_NUMBER
                       ea specified  is  not  a  valid  element
                       address of the robotic media changer.

     ER_OLD_VSN_NOT_UNK_MEDIA
                       old_vsn not matching unknown media VSN.

     ER_OPERATOR_NOT_PRIV
                       Operator does  not  have  permission  to
                       label removable media.

     ER_ROBOT_CATALOG_MISSING
                       No robot catalog was found for equipment
                       number  eq_number  which is defined as a
                       robotic media changer.

     ER_ROBOT_DEVICE_REQUIRED
                       No devices were found to be defined  for
                       equipment   number  eq_number  which  is
                       defined as a robotic media changer.

     ER_SLOT_NOT_OCCUPIED
                       No media was found to occupy the element
                       address  in  the media changer at equip-
                       ment number eq_number

     ER_VSN_BARCODE_REQUIRED
                       new_vsn must be specified.

     ER_UNABLE_TO_MAP_CATALOG
                       The  catalog  for  the  removable  media
                       changer at equipment number eq_number is
                       unable to be mapped into memory.

FILES
     mcf               The configuration file for Sun  QFS  and
                       SAM-QFS file systems.

SEE ALSO
     odlabel(1M), tplabel(1M).

     sam_odlabel(3).
```

sam_unarchive(3)

```
NAME
     sam_unarchive - Removes archive copies for a file or  directory

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

     #include "/opt/SUNWsamfs/include/lib.h"

     int sam_unarchive(const char *path, int num_opts, ... );

DESCRIPTION
     sam_unarchive() lets you remove an archive copy of a file or
     a directory using a Sun Storage Archive Manager system call.
     path is the file of which to delete  archive  entries,  fol-
     lowed by a sequence of num_opts input characters or options.
     Individual options are described below.

     For example, if you have used the sam_archive(3) function to
     request   that   a   file  be  archived,  you  can  use  the
     sam_unarchive(3) function to delete that archive copy.   The
     specifications  for  the archive copy (c copy_no) and/or the
     media type and VSN (m media_type  [v vsn])  determine  which
     archive copy is deleted.

     There are several  ways  to  specify  one  or  more  archive
     entries to be unarchived. These ways are as follows:

     o  By copy number

     o  By copy number, media type, and VSN

     o  By copy number and media type

     o  By media type

     o  By media type and VSN

OPTIONS
     c copy_no
          Deletes the specified archive copy_no.  Specify  1,  2,
          3,  or  4  for copy_no.  If one or more 'c' options are
          are specified, only those archive copies (1, 2,  3,  or
          4)  are  deleted.   Either a "c copy_no" or a "m media"
          option must be specified.

     M    Unarchives metadata only.  This  includes  directories,
          the  segment index, and removable media files.  Regular
          files and symbolic links are not  unarchived.   If  you
          are  unarchiving  a directory, you must specify the "M"
          option.

     m media
          Deletes all archive copies on the specified media_type.
          For the list of possible media_type specifications, see
          the mcf(4) man page.  Either a  "c  copy_no"  or  a  "m
          media"  option  must be specified.  If you specify a "m
```

```
               media" option, you can also specify a "v vsn" option.

          o    Specifies that the  file  must  be  online  before  its
               archive  entry  is deleted. If the file is offline, the
               sam_unarchive function stages the file to  disk  before
               deleting any entries.

          v vsn
               Deletes the archive copies on vsn.  For vsn, specify  a
               volume  serial  name  (VSN).   If you specify a "v vsn"
               option, you must also specify a "m media" option.

     RETURN VALUES
          Upon successful completion a value of 0 is returned.  Other-
          wise, a value of -1 is returned and errno is set to indicate
          the error.

     ERRORS
          sam_unarchive() 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 superuser.

          EFAULT              Argument points to an illegal address.

          EINTR               A   signal   was   caught   during   the
                              sam_unarchive() 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 com-
                              ponent    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 last (undamaged) copy of a file would be  unarchived,
          sam_unarchive would not unarchive that copy.

     SEE ALSO
          unarchive(1m), archive(1m), sam_archive(3), mcf(4)
```

sam_undamage(3)

```
NAME
     sam_undamage - Clears damaged and stale  status  of  archive
     entries of a file or directory

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

     #include "/opt/SUNWsamfs/include/lib.h"

     int sam_undamage(const char *path, int num_opts, ... );

DESCRIPTION
     Using  a  Sun   Storage   Archive   Manager   system   call,
     sam_undamage()  lets  you mark archive copies of a file or a
     directory as undamaged and not stale, based on  the  archive
     copy  number  and/or  the  media type and VSN specified. The
     function also marks the file itself as undamaged.   path  is
     the  file  on  which  to clear the attributes, followed by a
     sequence of num_opts input characters or  options.   Indivi-
     dual options are described below.

     There are several ways to mark one or more copies as  undam-
     aged and unstale.  These ways are as follows:

     o  By copy number

     o  By copy number, media type, and VSN

     o  By copy number and media type

     o  By media type

     o  By media type and VSN

OPTIONS
     a         Rearchives the damaged copy.

     c copy_no Marks the specified archive copy number as  undam-
               aged.   If  one or more 'c' options are specified,
               only those archive copies (1,  2,  3,  or  4)  are
               marked  as  undamaged.   Specify 1, 2, 3, or 4 for
               copy_no.  Either a "c  copy_no"  or  a  "m  media"
               option must be specified.

     M         Marks only metadata as undamaged.   This  includes
               directories, the segment index and removable-media
               files.  Regular files are not marked as undamaged.
               If  you  are marking a directory as undamaged, you
               must specify the "M" option.

     m media_type

               Marks all copies from the specified media_type  as
               undamaged.   For  the  list of possible media_type
               specifications, see the mcf(4) man page.  Either a
               "c  copy_no"  or a "m media" option must be speci-
               fied.  If you specify a "m media" option, you  can
               also specify a "v vsn" option.

     v vsn     Marks the archive copies on vsn as undamaged.  For
               vsn,  specify  a volume serial name (VSN).  If you
               specify a "v vsn" option, you must also specify  a
               "m media" option.

RETURN VALUES
     Upon successful completion a value of 0 is returned.  Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.

ERRORS
     sam_undamage() 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 superuser.

     EFAULT              Argument points to an illegal address.

     EINTR               A   signal   was   caught   during   the
                         sam_undamage() 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 com-
                         ponent    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
     damage(1m), undamage(1m), sam_damage(3), mcf(4)
```

sam_unload(3)

```
NAME
     sam_unload - Unloads media on the removable media device

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib  -lsamapi  [library ... ]

     #include "/opt/SUNWsamfs/include/samapi.h"

     int sam_unload(ushort_t eq_number, int wait_response);

DESCRIPTION
     sam_unload() requests that the media be  unloaded  from  the
     device with equipment number eq_number. The device must be a
     removable media device or a robotic media changer.  The dev-
     ice cannot be under the control of another process.

     If the equipment number eq_number is a removable media  dev-
     ice  controlled  by a robotic media changer, the medium will
     be moved into storage.

     If  the  equipment  number  eq_number  is  a  robotic  media
     changer,  the  unload moves catalog entries from the robotic
     media changer’s catalog to the Historian’s catalog.

     The call will return immediately after issuing  the  command
     if  zero is specified for wait_response value.  Other values
     for wait_response will give undefined results.

RETURN VALUES
     Upon succesful completion a value of 0 is returned.   Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.

ERRORS
     sam_unload() fails if one or more of the following are true:

     ER_DEVICE_NOT_READY The specified eq_number  device  is  not
                         ready.

     ER_DEVICE_NOT_UNAVAILABLE
                         The specified eq_number device  must  be
                         in    the    unavailable    state    (see
                         set_state(1M)).

     ER_DEVICE_USE_BY_ANOTHER
                         The specified eq_number device  is  busy
                         and is being used by another process.

     ER_FIFO_PATH_LENGTH The path and filename for the FIFO  pipe
                         is too long.

     ER_NO_DEVICE_FOUND  The   device   with   equipment   number
                         eq_number  is not available in this con-
                         figuration.

     ER_NO_EQUIP_ORDINAL eq_number is  not  a  defined  equipment
```

```
                       number in this configuration.

     ER_NO_MASTER_SHM    No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment defined.  Check that the
                         Sun QFS and  SAM-QFS  file  systems  are
                         mounted.

     ER_NO_MASTER_SHM_ATT
                         No Sun  QFS  or  SAM-QFS  master  shared
                         memory  segment  found.   Check that the
                         Sun QFS and  SAM-QFS  file  systems  are
                         mounted.

     ER_NO_RESPONSE_FIFO Unable to create the response FIFO pipe.

     ER_NOT_REMOV_MEDIA_DEVICE
                         The specified eq_number device is not  a
                         removable media device.

     ER_OPERATOR_NOT_PRIV
                         Operator does  not  have  permission  to
                         unload removable media.

FILES
     mcf                 The configuration file for Sun  QFS  and
                         SAM-QFS file systems.

SEE ALSO
     load(1M), sam-robots(1M), set_state(1M), unload(1M).

     sam_load(3).

     mcf(4).
```

sam_unrearch(3)

```
NAME
     sam_unrearch - Removes rearchive attributes  on  a  file  or
     directory

SYNOPSIS
     cc [ flag  ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

     #include "/opt/SUNWsamfs/include/lib.h"

     int sam_unrearch(const char *path, int num_opts, ... );

DESCRIPTION
     sam_unrearch() lets you remove a request to rearchive a file
     or  a  directory  using a Sun Storage Archive Manager system
     call.  path is the file on which to remove  the  attributes,
     followed  by  a  sequence  of  num_opts  input characters or
     options.  Individual options are described below.

     For example, if you have used the sam_rearch(3) function  to
     request   that  a  file  be  rearchived,  you  can  use  the
```

sam_unrearch function to clear the bit that the sam_rearch(3) function had set. The specifications for the archive copy (c copy_no) and/or the media type and VSN (m media_type [v vsn]) determine which archive copy is affected.

There are several ways to remove the request to rearchive from one or more archive entries. These ways are as follows:

o By copy number

o By copy number, media type, and VSN

o By copy number and media type

o By media type

o By media type and VSN

OPTIONS

c copy_no
: Removes the rearchive request for copy_no. Specify 1, 2, 3, or 4 for copy_no. If one or more ’c’ options are are specified, the function removes the rearchive request from only those archive copies (1, 2, 3, or 4). Either a "c copy_no" or a "m media" option must be specified.

M
: Removes rearchive requests for metadata only. This includes directories, the segment index, and removable

media files. Regular files and symbolic links are not unrearchived. If you are unarchiving a directory, you must specify the "M" option.

m media
: Removes rearchive requests from all archive copies on the specified media_type. For the list of possible media_type specifications, see the mcf(4) man page. Either a "c copy_no" or a "m media" option must be specified. If you specify a "m media" option, you can also specify a "v vsn" option.

v vsn
: Removes the rearchive requests for the archive copies on vsn. For vsn, specify a volume serial name (VSN). If you specify a "v vsn" option, you must also specify a "m media" 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_unrearch() 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 superuser.
EFAULT	Argument points to an illegal address.
EINTR	A signal was caught during the sam_unrearch() 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
unrearch(1m), rearch(1m), sam_rearch(3), mcf(4)

sam_vsn_stat(3)

NAME
sam_vsn_stat, sam_segment_vsn_stat - Gets VSN status for an archive copy that overflows VSNs

SYNOPSIS
```
cc [ flag ... ] file    ...  -L/opt/SUNWsamfs/lib  -lsam  [library ... ]

#include </opt/SUNWsamfs/include/stat.h>

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

int sam_segment_vsn_stat(const char  *path,  int  copy,  int
segment_ord, struct sam_section *buf, size_t bufsize);
```

DESCRIPTION
The sam_vsn_stat() function obtains information about the VSNs for the archive copy indicated by copy of path, where path points to a non-segmented file.

If sam_vsn_stat() is called and path points to a segmented file, then VSN information about the archive copy of the segmented file's index inode is returned.

The sam_segment_vsn_stat() function obtains information about the VSNs for the archive copy indicated by copy of the data segment indicated by segment_ord of the segmented file pointed to by path.

sam_vsn_stat() and sam_segment_vsn_stat() obtain information about the VSNs for the indicated archive copy when the indicated archive copy uses multiple VSNs.

sam_vsn_stat() and sam_segment_vsn_stat() fail if called to obtain VSN stat information for an archive copy that only uses one VSN. Use the sam_stat() or sam_segment_stat() subroutines to determine the number of VSNs used by a given archive copy and to get VSN information for archive copies that only use one VSN.

sam_vsn_stat() places VSN information for all of the sections that comprise the overflowed archive copy into buf.

Read, write, or execute permission of the named file is not required, but all directories listed in the path name leading to the file must be searchable.

copy is the archive copy number (0, 1, 2 or 3).

segment_ord is the data segment number (0, ..., n_segs - 1) where n_segs is the current number of data segments that

comprise the file pointed to by path.

buf is a pointer to a sam_section structure into which VSN information is placed concerning the file's archive copy.

bufsize is the length of the user's buffer to which buf points. sam_vsn_stat and sam_segment_vsn_stat place VSN information for each overflowed section that comprises the archive copy into buf. Hence, bufsize should be at least sizeof(struct sam_vsn_stat) * n_vsns bytes, where n_vsns is the number of VSNs used by the archived copy.

The contents of the structure pointed to by buf include the following struct sam_section members:

```
     char         vsn[32];
     u_longlong_t length;
     u_longlong_t position;
     u_longlong_t offset;
```

vsn	The VSN of the section. This is a null-terminated string with a maximum of 31 characters.
length	The length of the section on the volume.
position	The position of the start of the archive file that contains this section.
offset	The offset of this file on the archive file.

RETURN VALUES

Upon successful completion, a value of 0 is returned. Otherwise, a value of -1 is returned and errno is set to indicate the error.

ERRORS

sam_vsn_stat() and sam_segment_vsn_stat() 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_vsn_stat() function.
ELOOP	Too many symbolic links were encountered in translating path.
EMULTIHOP	Components of path require hopping to multiple remote machines and the file system does not allow it.
ENAMETOOLONG	The length of the path argument exceeds {PATH_MAX}, or the length of a path component exceeds {NAME_MAX} while {_POSIX_NO_TRUNC} is in effect.
ENOENT	The named file does not exist or is the null pathname.
ENOLINK	path points to a remote machine and the link to that machine is no longer active.
ENOTDIR	A component of the path prefix is not a directory.
EOVERFLOW	A component is too large to store in the structure pointed to by buf.

USAGE

sam_vsn_stat	Call sam_stat to get the number of VSNs used for the archive copy. The call to sam_stat will write the number of VSNs used by the archive copy in your struct sam_stat buffer in the member copy[copy].n_vsns. If the archive copy uses only one VSN (the number of VSNs is 1), then your program or script must retrieve the VSN information for the archive copy from the copy member of the sam_stat structure that was filled in when your program or script called sam_stat. The copy member of the sam_stat structure

```
                      is of type struct sam_copy_s.

sam_segment_vsn_stat  Call sam_stat to determine whether the
                      file pointed to by path is segmented.

                      If the file pointed to by path is  not
                      segmented,  then  use  sam_vsn_stat to
                      obtain  VSN  information  as  detailed
                      above.

                      If the file pointed to by path is seg-
                      mented,  then call sam_segment_stat to
                      get the number of VSNs  used  for  the
                      archive  copy indicated by copy of the
                      data segment indicated by segment_ord.

                      The  call  to  sam_segment_stat   will
                      write  the  number of VSNs used by the
                      archive copy  of  the  indicated  data
                      segment  in  your  array  of  sam_stat
                      structures in the  member  located  in
                      sam_stat_buff_array[segment_ord].copy[copy].n_vsns.

                      If the archive copy uses only one  VSN
                      (the  number  of VSNs is 1), then your
                      program or script  must  retrieve  the
                      VSN  information  for the archive copy
                      from the copy member of the element in
                      the  array of sam_stat structures that
                      was filled in  when  your  program  or
                      script  called  sam_segment_stat.  The
                      copy member of the sam_stat  structure
                      is  of  type  struct sam_copy_s and is
                      found in the array of sam_stat  struc-
                      tures under the index segment_ord.

A struct sam_copy_s structure has the following members:

     u_longlong_t position;
     time_t       creation_time;
     uint_t       offset;
     ushort_t     flags;
     short        n_vsns;
     char         media[4];
     char         vsn[32];

position       Location of the archive file

creation_time  Time that the archive copy was created

offset         Location of the copy in the archive file

flags          Sun  QFS  and  SAM-QFS  archive  copy  status
               flags.   These  indicate  whether the archive
               copy has been made,  is  stale,  is  damaged,
               etc.   See  /opt/SUNWsamfs/include/stat.h for
               bit masks which can be applied to these flags
               to  resolve  the  state  and  status  of  the
               archive copy.
```

```
          n_vsns         Number of VSNs used  by  the  archived  copy.
                         Will  be  1  in  case of no overflow, will be
                         greater than one if the  archive  copy  over-
                         flows volumes.

          media          Media type.  This is a null-terminated string
                         with a maximum of 3 characters.

          vsn            The VSN  of  the  copy.   This  is   a  null-
                         terminated  string with a maximum of 31 char-
                         acters.

          If the archive copy uses more than one VSN  (the  number  of
          VSNs  is  greater  than 1), then your program or script must
          call sam_vsn_stat or sam_segment_vsn_stat  to  retrieve  the
          VSN  information  for  all of the sections that comprise the
          archive copy.

          Do not call  sam_vsn_stat  or  sam_segment_vsn_stat  if  the
          archive copy uses only one VSN (does not overflow).

     SEE ALSO
          sam_stat(3)

     NOTES
          The Sun QFS and SAM-QFS file systems  permit  a  maximum  of
          MAX_VOLUMES  sections  per  archive copy.  Hence, instead of
          dynamically allocating a buffer of structures, a more  effi-
          cient   method   is  to  to  declare  a  static  array  with
          MAX_VOLUMES number of elements.

          The  constant  MAX_VOLUMES  is  declared  in  the  following
          include file:  /opt/SUNWsamfs/include/rminfo.h .
```

usam_mig_cancel_stage_req(3)

```
NAME
     usam_mig_cancel_stage_req - Cancels a  foreign  media  stage
     request

SYNOPSIS
     cc [ flag  ... ] file  ...  -L/opt/SUNWsamfs/lib  -lsamut  [library ... ]

     #include "/opt/SUNWsamfs/include/mig.h"

     int usam_mig_cancel_stage_req(tp_stage_t *stage_req );

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     usam_mig_cancel_stage_req() cancels a stage request from the
     foreign data migration program, written by the integrator.

     The stager daemon, sam-stagerd, is expected  to  cancel  the
     stage  request on its worklist.  Only the inode and fseq can
     be used to find the stage request to be canceled.  stage_req
     is a pointer to a tp_api structure into which information is
     placed regarding the offset, size, position,  etc.   of  the
     data  file.   The  contents  of  the structure pointed to by
     stage_req include the following members:

          offset_t  offset;        /* Offset from beginning of the file */
          offset_t  size;          /* Size of the file to stage */
          long long position;      /* The position field from the archive info in the inode */
          ino_t     inode;         /* Inode number from the file system */
          vsn_t     space;         /* VSN field from the archive information in the inode */
          equ_t     fseq;          /*  Equipment number of family set in the inode */
          char      media_type[2]; /*  2 character media type for the foreign media*/

     offset    The offset from the beginning of the file for this
               stage  request.  As the system is reading a "stage
               never" file, the file offset moves down the  file.
               For  a  normal stage of a file the stage offset is
               zero.

     size      The size of the  file  to  stage  for  this  stage
               request.   During a "stage never" request, this is
               the size the file system wants to deliver at  this
               time.   For  a  normal stage of a file the size is
               the size of the file.

     position  The position field(s) from the archive information
               in the inode.

     inode     The inode number from the file system.

     vsn       The vsn field from the archive information in  the
               inodes.

     fseq      The equipment number of the  family  set  for  the
               inode.

     media_type[2]
               The two  character  media  type  for  the  foreign
               media.  Upon succesful initialization a value of 0
               is returned.  Otherwise, a value of 1 is  returned
               and errno is set to indicate the error.

ERRORS
     usam_mig_cancel_stage_req() fails if the following is true:

     ECANCELED

FILES
     /opt/SUNWsamfs/migkit/mig_cd.c
                         The example Migration Toolkit program.

     /etc/opt/SUNWsamfs/mcf
                         The configuration file for Sun  QFS  and
                         SAM-QFS file systems.

SEE ALSO
     mcf(4).
```

usam_mig_initialize(3)

```
NAME
     usam_mig_initialize - Initializes the migration interface

SYNOPSIS
     cc [ flag  ... ] file  ...  -L/opt/SUNWsamfs/lib  -lsamut  [library ... ]

     #include "/opt/SUNWsamfs/include/mig.h"

     int usam_mig_initialize(int stage_count);

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     usam_mig_initialize() is the initialization routine for  the
     SAM-QFS  Migration  Toolkit.   It  is  called by the foreign
     media "device" to allow  the  interface  to  initialize  any
     local structs, threads, etc.

     stage_count is the maximum number of stage requests that may
     be  outstanding  at  one  time.   Every  stage request for a
     foreign  media  type  is  handed  to  the   stager   daemon.
     sam-stagerd(1M)  must  be  able  to  handle this stage_count
     requests at one time.

RETURN VALUES
     Upon succesful initialization a  value  of  0  is  returned.
     Otherwise,  a  value  of  1  is returned and errno is set to
     indicate the error.

FILES
     /opt/SUNWsamfs/migkit/mig_cd.c
                         The example Migration Toolkit program.

     /etc/opt/SUNWsamfs/mcf
                         The configuration file for Sun  QFS  and
                         SAM-QFS file systems.

SEE ALSO
     sam-stagerd(1M).  mcf(4).
```

usam_mig_stage_file_req(3)

```
NAME
     usam_mig_stage_file_req - Stages request  from  the  foreign
     data migration program

SYNOPSIS
     cc [ flag  ... ] file  ...  -L/opt/SUNWsamfs/lib  -lsamut  [library ... ]

     #include "/opt/SUNWsamfs/include/mig.h"

     int usam_mig_stage_file_req();
```

```
AVAILABILITY
     SUNWsamfs

DESCRIPTION
     usam_mig_stage_file_req() is the stage request  routine  for
     the  SAM-QFS  Migration Toolkit.  The foreign data migration
     program adds this stage request to  an  internally-generated
     worklist being maintained by third party API.  This worklist
     should   be   processed   by    a    thread    started    by
     usam_mig_initialize().  This worklist is processed by a pro-
     gram started with usam_mig_initialize().  The  thread  finds
     and positions the media to stage the file through stage_api.

RETURN VALUES
     A successful stage request returns a value of 0.  Otherwise,
     a  value  of  1  is returned and errno is passed to the file
     system.

ERRORS
     usam_mig_stage_file_req() fails if the following is true:

     EEXIST              This is a duplicate stage request for  a
                         file.

FILES
     /opt/SUNWsamfs/migkit/mig_cd.c
                         The example Migration Toolkit program.

     /etc/opt/SUNWsamfs/mcf
                         The configuration file for Sun  QFS  and
                         SAM-QFS file systems.

SEE ALSO
     sam_migd(3).

     mcf(4).
```

CHAPTER 4

Library Functions (Man Pages Section 3X)

This chapter provides the section 3X man pages for Sun QFS and Sun Storage Archive Manager.

intro_libsam(3X)

```
NAME
     intro_libsam, intro_libsamrpc - Introduces the Sun QFS and
     and SAM-QFS Application Programmer Interface (API) routines

AVAILABILITY
     SUNWqfs

     SUNWsamfs

DESCRIPTION
     The Sun QFS and SAM-QFS API allows a Sun QFS or SAM-QFS file
     to be requested from within an application program.  The
     aplication program can reside either on the machine upon
     which the Sun QFS or SAM-QFS file system is running or on
     another machine on the network.  This man page provides an
     introduction to the API routines.

     The following topics are presented:

     o API overview

     o API library routines

     o Using libsam

     o Using libsamrpc

API OVERVIEW
     When a request is made, the process or program making the
     request is the client process or program, running on the
     client machine.  The requests are received and processed by
     the server, running on the server, or host, machine.  For
     the API routines, the server machine is always the machine
     upon which the Sun QFS or SAM-QFS file system is running.
```

In the simplest case, the client and server machines are the same, and no network communication is necessary. In other cases, however, the application programmer needs to allow for the client program to run on a machine where the Sun QFS or SAM-QFS file system is not running. In this case, networked library calls from libsamrpc must be used.

The two API libraries available with the Sun QFS and SAM-QFS file systems are as follows:

o libsam. The library calls in libsam do not perform network communication. They only make local requests. In this case, each library call makes a system call, and the server is the local operating system.

o libsamrpc. The library calls in libsamrpc use Remote Procedure Calls (RPCs) to communicate with a special server process, sam-rpcd. Because of the RPC mechanism, the client and server can exist on the same machine or on different machines in the network. The server process always runs on the machine upon which the Sun QFS or SAM-QFS file system is running.

Both libsam and libsamrpc are released in shared object (.so) and archive (.a) format for Solaris platforms. libsam.so and libsam.a are installed in /opt/SUNWsamfs/lib. libsamrpc.so and libsamrpc.a are installed in /opt/SUNWsamfs/client/lib, with symbolic links to them in /opt/SUNWsamfs/lib.

API LIBRARY ROUTINES

The library calls for the Sun QFS and SAM-QFS software are supported in libsam, and a subset is supported in libsamrpc.

Table 1 lists the API library routines and indicates the environments in which they are supported. In addition, table 1 indicates the libraries in which they are included:

Table 1. Library routine availability

Routine	Description
sam_advise	Sets file attributes. Availability: Sun QFS and SAM-QFS environments. Libraries: libsam.
sam_archive	Sets archive attributes on a file. Availability: SAM-QFS environments. Libraries: libsam and libsamrpc.
sam_rearchive	Sets rearchive attributes on a file. Availability: SAM-QFS environments. Libraries: libsam.
sam_exarchive	Exchanges archive copies of a file or directory.

```
                   Availability:  SAM-QFS environments.
                   Libraries:  libsam.

     sam_unarchive  Removes archive copies for a file or
                   directory.
                   Availability:  SAM-QFS environments.
                   Libraries:  libsam.

     sam_unrearch   Removes rearchive attributes on a file or
                   directory.
                   Availability:  SAM-QFS environments.
                   Libraries:  libsam.

     sam_damage     Sets damaged attribute on a file or
                   directory.
                   Availability:  SAM-QFS environments.
                   Libraries:  libsam.

     sam_undamage   Clears damaged and stale status of a file or
                   directory.
                   Availability:  SAM-QFS environments.
                   Libraries:  libsam.

     sam_cancelstage
                   Cancels a pending or in-progress stage on a
                   file.
                   Availability:  SAM-QFS environments.
                   Libraries:  libsam.

     sam_closecat   Ends access to the catalog for an automated
                   library.
                   Availability:  SAM-QFS environments.
                   Libraries:  libsam.

     sam_closerpc   Closes down the RPC connection.
                   Availability:  SAM-QFS environments.
                   Libraries:  libsamrpc.

     sam_devstat, sam_ndevstat
                   Gets device status.  sam_ndevstat accepts a
                   longer device name.
                   Availability:  SAM-QFS environments.
                   Libraries:  libsam.

     sam_devstr     Translates numeric device status into a
                   character string.
                   Availability:  SAM-QFS environments.
                   Libraries:  libsam.

     sam_getcatalog Obtains a range of entries from the catalog
                   for an automated library.
                   Availability:  SAM-QFS environments.
                   Libraries:  libsam.

     sam_initrpc    Initializes the RPC connection.
                   Availability:  SAM-QFS environments.
                   Libraries:  libsamrpc.

     sam_opencat    Accesses the VSN catalog for an automated
```

```
                       library.
                       Availability:  SAM-QFS environments.
                       Libraries:  libsam.

        sam_readrminfo Gets information for a removable media file.
                       Availability:  SAM-QFS environments.

                       Libraries:  libsam.

        sam_release    Releases and sets release attributes on a
                       file.
                       Availability:  SAM-QFS environments.
                       Libraries:  libsam and libsamrpc.

        sam_request    Creates a removable media file.
                       Availability:  SAM-QFS environments.
                       Libraries:  libsam.

        sam_restore_copy
                       Creates an archive copy for a file.
                       Availability:  SAM-QFS environments.
                       Libraries:  libsam.

        sam_restore_file
                       Creates an offline file.
                       Availability:  SAM-QFS environments.
                       Libraries:  libsam.

        sam_segment    Sets segment attributes on a file or
                       directory.
                       Availability:  SAM-QFS environments.
                       Libraries:  libsam and libsamrpc.

        sam_segment_stat
                       Obtains file information and follows symbolic
                       links to a segmented file.
                       Availability:  SAM-QFS environments.
                       Libraries:  libsam.

        sam_setfa      Sets file attributes.
                       Availability:  Sun QFS and SAM-QFS
                       environments.
                       Libraries:  libsam and libsamrpc.

        sam_ssum       Sets checksum attributes on a file.
                       Availability:  SAM-QFS environments.
                       Libraries:  libsam.

        sam_stage      Stages and sets stage attributes on a file.
                       Availability:  SAM-QFS environments.
                       Libraries:  libsam and libsamrpc.

        sam_stat, sam_lstat
                       sam_stat obtains file information and follows
                       symbolic links to the file.  sam_lstat
                       obtains file information, and if that file is
                       a link, it returns information about the
                       link.
                       Availability:  Sun QFS and SAM-QFS
```

```
                    environments.
                    Libraries:  libsam and libsamrpc.

     sam_vsn_stat, sam_segment_vsn_stat
                    Obtain VSN status for a file or a file's data
                    segment that overflows VSNs.
                    Availability:  SAM-QFS environments.
                    Libraries:  libsam.

     All APIs in libsam, except for sam_closecat, sam_getcatalog,
     and sam_opencat, are available for use with 64-bit programs.
     Oracle Corporation does not support a 64-bit version of
     libsamrpc.

     For more details about each library routine, see the
     individual corresponding man page for that routine.  Library
     routines contained in libsam are found in section 3 of the
     online man pages.  Library routines contained in libsamrpc
     are found in section 3X of the online man pages.

USING libsam
     No special initialization or configuration is required prior
     to using the API library routines in libsam.  The
     application program must be linked with libsam, however.
     For information on the routines, see the individual libsam
     man pages, all of which are listed in the SEE ALSO section
     of this man page.

USING libsamrpc
     The source code for libsamrpc is included in the release for
     customers who wish to write and run application programs on
     platforms that do not run the Solaris operating system.  In
     these cases, the library must be ported to the client
     machine.  The source code is located in
     /opt/SUNWsamfs/client/src.  Example application programs are
     located in /opt/SUNWsamfs/client/examples.

  Specifying the Server Machine
     A call to sam_initrpc is required before any other RPC
     client API calls can be executed successfully.  Only one
     sam_initrpc call is required, followed by any number of
     other client API calls (other than sam_closerpc).  The
     sam_initrpc call accepts one argument:  a pointer to a
     character string that specifies the name of the server
     machine.  If this pointer is NULL, sam_initrpc checks for an
     environment variable named SAMHOST.  If this environment
     variable is set, that name is used for the server machine.
     If there is no SAMHOST environment variable, the default
     server name samhost is used.

     In summary, the name of the server machine can be specified
     in any of three ways, which are checked by sam_initrpc in

     the following order:

     1. As an argument to the sam_initrpc call.

     2. As the environment variable SAMHOST.
```

3. By accepting the default server name, samhost.

RPC Server Process

The RPC API server process receives and processes requests from the client. This server process, /opt/SUNWsamfs/sbin/sam-rpcd, must be run on the same machine as the file system. The sam-rpcd daemon must be running for client requests to execute successfully.

The sam-rpcd daemon is started automatically by sam-amld if the appropriate entry is made in the defaults.conf file. For information on editing the defaults.conf file, see Configuring the API later in this man page.

The sam-rpcd daemon can also be started manually. It should be run as superuser. The sam-rpcd command accepts no arguments.

The sam-rpcd daemon services the requests it receives by making the appropriate system call on the server machine and then returning the output or result to the client. For more information on this daemon, see the sam-rpcd(1M) man page.

Configuring the API

The following steps describe setting up the API server and clients. These steps assume that your software is properly configured and running.

Step 1: Configure the API Server

For the server portion of the API to run successfully, the following conditions must be present:

- o The RPC program name and number pair must be known on the server machine

- o The RPC program name and number pair must be the same as the pair used on the API client machines.

Make an entry for the RPC program name and number. The RPC program number is a number chosen by you. The RPC program name is samfs. The name and number pair must be the same on the server and all clients. The /etc/nsswitch.conf file determines where you should specify the RPC program name and number pair. For more information on this, see the nsswitch.conf(4) man page.

In /etc/rpc (or the NIS database), add the following line:

```
samfs          150005
```

In /etc/services (or the NIS database), add the following line:

```
samfs          5012/tcp  # SAM-QFS API
```

The API server is started automatically by the sam-amld daemon if the following entry is made in the defaults.conf file (note that changes to the defaults.conf file do not take effect until the next time the sam-amld daemon is initialized):

```
samrpc = on
```

The sam-rpcd daemon is not automatically started if no entry for it appears in the defaults.conf file or if the following entry appears in the file:

```
samrpc = off
```

For more information about the defaults.conf file, see the defaults.conf(4) man page.

Step 2: Configure the API Client Machines

The following two configuration components must be present on the client machine for the RPC communication to be successful:

o The name of the server machine.

o The RPC program name and number pair.

Make an entry for the RPC program name and number on all client machines, as you did on the API server machine previously. Again, the RPC program name must be samfs. The RPC program number is a number chosen by you, but it must be the same on the server and client machines.

In /etc/rpc (or the NIS database), add the following line:

```
samfs          150005
```

The host name of the server machine must be known on the client machine. For default cases, the host name samhost must be listed as an alias for the SAM-QFS file system server machine. For more information, see the sam_initrpc(3X) man page.

Authentication and libsamrpc

Authentication information is generated at the time of the sam-initrpc call. This information consists of the user identification (uid) and group identification (gid) of the calling process. It is associated with the connection made to the RPC server process.

Subsequent libsamrpc calls have this information associated. When the request is received by the RPC server process on the server machine, the uid and gid information is used. File access and operations are granted or denied based on this information.

It is important that the server machine have a common uid and gid space with the client machines.

SEE ALSO

sam_advise(3), sam_archive(3), sam_rearch(3),

```
          sam_exarchive(3), sam_unarchive(3), sam_unrearch(3),
          sam_damage(3), sam_undamage(3), sam_cancelstage(3),
          sam_closecat(3), sam_devstat(3), sam_devstr(3),
          sam_getcatalog(3), sam_lstat(3), sam_ndevstat(3),
          sam_opencat(3), sam_readrminfo(3), sam_release(3),
          sam_request(3), sam_restore_copy(3), sam_restore_file(3),
          sam_segment(3), sam_setfa(3), sam_ssum(3), sam_stage(3),
          sam_stat(3).

          sam_archive(3X), sam_closerpc(3X), sam_initrpc(3X),
          sam_lstat(3X), sam_release(3X), sam_stage(3X), sam_stat(3X).
```

intro_libsamrpc(3X)

```
     NAME
          intro_libsam, intro_libsamrpc - Introduces the Sun QFS and
          and SAM-QFS Application Programmer Interface (API) routines

     AVAILABILITY
          SUNWqfs

          SUNWsamfs

     DESCRIPTION
          The Sun QFS and SAM-QFS API allows a Sun QFS or SAM-QFS file
          to be requested from within an application program.  The
          aplication program can reside either on the machine upon
          which the Sun QFS or SAM-QFS file system is running or on
          another machine on the network.  This man page provides an
          introduction to the API routines.

          The following topics are presented:

          o API overview

          o API library routines

          o Using libsam

          o Using libsamrpc

     API OVERVIEW
          When a request is made, the process or program making the
          request is the client process or program, running on the
          client machine.  The requests are received and processed by
          the server, running on the server, or host, machine.  For
          the API routines, the server machine is always the machine
          upon which the Sun QFS or SAM-QFS file system is running.

          In the simplest case, the client and server machines are the
          same, and no network communication is necessary.  In other
          cases, however, the application programmer needs to allow
          for the client program to run on a machine where the Sun QFS
          or SAM-QFS file system is not running.  In this case,
          networked library calls from libsamrpc must be used.
```

The two API libraries available with the Sun QFS and SAM-QFS file systems are as follows:

- o libsam. The library calls in libsam do not perform network communication. They only make local requests. In this case, each library call makes a system call, and the server is the local operating system.

- o libsamrpc. The library calls in libsamrpc use Remote Procedure Calls (RPCs) to communicate with a special

  server process, sam-rpcd. Because of the RPC mechanism, the client and server can exist on the same machine or on different machines in the network. The server process always runs on the machine upon which the Sun QFS or SAM-QFS file system is running.

Both libsam and libsamrpc are released in shared object (.so) and archive (.a) format for Solaris platforms. libsam.so and libsam.a are installed in /opt/SUNWsamfs/lib. libsamrpc.so and libsamrpc.a are installed in /opt/SUNWsamfs/client/lib, with symbolic links to them in /opt/SUNWsamfs/lib.

API LIBRARY ROUTINES

The library calls for the Sun QFS and SAM-QFS software are supported in libsam, and a subset is supported in libsamrpc.

Table 1 lists the API library routines and indicates the environments in which they are supported. In addition, table 1 indicates the libraries in which they are included:

Table 1. Library routine availability

Routine	Description
sam_advise	Sets file attributes. Availability: Sun QFS and SAM-QFS environments. Libraries: libsam.
sam_archive	Sets archive attributes on a file. Availability: SAM-QFS environments. Libraries: libsam and libsamrpc.
sam_rearchive	Sets rearchive attributes on a file. Availability: SAM-QFS environments. Libraries: libsam.
sam_exarchive	Exchanges archive copies of a file or directory. Availability: SAM-QFS environments. Libraries: libsam.
sam_unarchive	Removes archive copies for a file or directory. Availability: SAM-QFS environments. Libraries: libsam.

```
     sam_unrearch   Removes rearchive attributes on a file or
                    directory.
                    Availability:  SAM-QFS environments.
                    Libraries:  libsam.

     sam_damage     Sets damaged attribute on a file or
                    directory.
                    Availability:  SAM-QFS environments.
                    Libraries:  libsam.

     sam_undamage   Clears damaged and stale status of a file or
                    directory.
                    Availability:  SAM-QFS environments.
                    Libraries:  libsam.

     sam_cancelstage
                    Cancels a pending or in-progress stage on a
                    file.
                    Availability:  SAM-QFS environments.
                    Libraries:  libsam.

     sam_closecat   Ends access to the catalog for an automated
                    library.
                    Availability:  SAM-QFS environments.
                    Libraries:  libsam.

     sam_closerpc   Closes down the RPC connection.
                    Availability:  SAM-QFS environments.
                    Libraries:  libsamrpc.

     sam_devstat, sam_ndevstat
                    Gets device status.  sam_ndevstat accepts a
                    longer device name.
                    Availability:  SAM-QFS environments.
                    Libraries:  libsam.

     sam_devstr     Translates numeric device status into a
                    character string.
                    Availability:  SAM-QFS environments.
                    Libraries:  libsam.

     sam_getcatalog Obtains a range of entries from the catalog
                    for an automated library.
                    Availability:  SAM-QFS environments.
                    Libraries:  libsam.

     sam_initrpc    Initializes the RPC connection.
                    Availability:  SAM-QFS environments.
                    Libraries:  libsamrpc.

     sam_opencat    Accesses the VSN catalog for an automated
                    library.
                    Availability:  SAM-QFS environments.
                    Libraries:  libsam.

     sam_readrminfo Gets information for a removable media file.
                    Availability:  SAM-QFS environments.

                    Libraries:  libsam.
```

```
sam_release    Releases and sets release attributes on a
               file.
               Availability:  SAM-QFS environments.
               Libraries:  libsam and libsamrpc.

sam_request    Creates a removable media file.
               Availability:  SAM-QFS environments.
               Libraries:  libsam.

sam_restore_copy
               Creates an archive copy for a file.
               Availability:  SAM-QFS environments.
               Libraries:  libsam.

sam_restore_file
               Creates an offline file.
               Availability:  SAM-QFS environments.
               Libraries:  libsam.

sam_segment    Sets segment attributes on a file or
               directory.
               Availability:  SAM-QFS environments.
               Libraries:  libsam and libsamrpc.

sam_segment_stat
               Obtains file information and follows symbolic
               links to a segmented file.
               Availability:  SAM-QFS environments.
               Libraries:  libsam.

sam_setfa      Sets file attributes.
               Availability:  Sun QFS and SAM-QFS
               environments.
               Libraries:  libsam and libsamrpc.

sam_ssum       Sets checksum attributes on a file.
               Availability:  SAM-QFS environments.
               Libraries:  libsam.

sam_stage      Stages and sets stage attributes on a file.
               Availability:  SAM-QFS environments.
               Libraries:  libsam and libsamrpc.

sam_stat, sam_lstat
               sam_stat obtains file information and follows
               symbolic links to the file.  sam_lstat
               obtains file information, and if that file is
               a link, it returns information about the
               link.
               Availability:  Sun QFS and SAM-QFS

               environments.
               Libraries:  libsam and libsamrpc.

sam_vsn_stat, sam_segment_vsn_stat
               Obtain VSN status for a file or a file’s data
               segment that overflows VSNs.
               Availability:  SAM-QFS environments.
```

Libraries: libsam.

All APIs in libsam, except for sam_closecat, sam_getcatalog, and sam_opencat, are available for use with 64-bit programs. Oracle Corporation does not support a 64-bit version of libsamrpc.

For more details about each library routine, see the individual corresponding man page for that routine. Library routines contained in libsam are found in section 3 of the online man pages. Library routines contained in libsamrpc are found in section 3X of the online man pages.

USING libsam

No special initialization or configuration is required prior to using the API library routines in libsam. The application program must be linked with libsam, however. For information on the routines, see the individual libsam man pages, all of which are listed in the SEE ALSO section of this man page.

USING libsamrpc

The source code for libsamrpc is included in the release for customers who wish to write and run application programs on platforms that do not run the Solaris operating system. In these cases, the library must be ported to the client machine. The source code is located in /opt/SUNWsamfs/client/src. Example application programs are located in /opt/SUNWsamfs/client/examples.

Specifying the Server Machine

A call to sam_initrpc is required before any other RPC client API calls can be executed successfully. Only one sam_initrpc call is required, followed by any number of other client API calls (other than sam_closerpc). The sam_initrpc call accepts one argument: a pointer to a character string that specifies the name of the server machine. If this pointer is NULL, sam_initrpc checks for an environment variable named SAMHOST. If this environment variable is set, that name is used for the server machine. If there is no SAMHOST environment variable, the default server name samhost is used.

In summary, the name of the server machine can be specified in any of three ways, which are checked by sam_initrpc in

the following order:

1. As an argument to the sam_initrpc call.

2. As the environment variable SAMHOST.

3. By accepting the default server name, samhost.

RPC Server Process

The RPC API server process receives and processes requests from the client. This server process, /opt/SUNWsamfs/sbin/sam-rpcd, must be run on the same machine as the file system. The sam-rpcd daemon must be running for client requests to execute successfully.

The sam-rpcd daemon is started automatically by sam-amld if the appropriate entry is made in the defaults.conf file. For information on editing the defaults.conf file, see Configuring the API later in this man page.

The sam-rpcd daemon can also be started manually. It should be run as superuser. The sam-rpcd command accepts no arguments.

The sam-rpcd daemon services the requests it receives by making the appropriate system call on the server machine and then returning the output or result to the client. For more information on this daemon, see the sam-rpcd(1M) man page.

Configuring the API

The following steps describe setting up the API server and clients. These steps assume that your software is properly configured and running.

Step 1: Configure the API Server

For the server portion of the API to run successfully, the following conditions must be present:

- o The RPC program name and number pair must be known on the server machine
- o The RPC program name and number pair must be the same as the pair used on the API client machines.

Make an entry for the RPC program name and number. The RPC program number is a number chosen by you. The RPC program name is samfs. The name and number pair must be the same on the server and all clients. The /etc/nsswitch.conf file determines where you should specify the RPC program name and number pair. For more information on this, see the nsswitch.conf(4) man page.

In /etc/rpc (or the NIS database), add the following line:

```
samfs          150005
```

In /etc/services (or the NIS database), add the following line:

```
samfs          5012/tcp  # SAM-QFS API
```

The API server is started automatically by the sam-amld daemon if the following entry is made in the defaults.conf file (note that changes to the defaults.conf file do not take effect until the next time the sam-amld daemon is initialized):

```
samrpc = on
```

The sam-rpcd daemon is not automatically started if no entry for it appears in the defaults.conf file or if the following entry appears in the file:

```
samrpc = off
```

For more information about the defaults.conf file, see the defaults.conf(4) man page.

Step 2: Configure the API Client Machines

The following two configuration components must be present on the client machine for the RPC communication to be successful:

- o The name of the server machine.
- o The RPC program name and number pair.

Make an entry for the RPC program name and number on all client machines, as you did on the API server machine previously. Again, the RPC program name must be samfs. The RPC program number is a number chosen by you, but it must be the same on the server and client machines.

In /etc/rpc (or the NIS database), add the following line:

```
samfs          150005
```

The host name of the server machine must be known on the client machine. For default cases, the host name samhost must be listed as an alias for the SAM-QFS file system server machine. For more information, see the sam_initrpc(3X) man page.

Authentication and libsamrpc

Authentication information is generated at the time of the sam-initrpc call. This information consists of the user identification (uid) and group identification (gid) of the calling process. It is associated with the connection made to the RPC server process.

Subsequent libsamrpc calls have this information associated. When the request is received by the RPC server process on the server machine, the uid and gid information is used. File access and operations are granted or denied based on this information.

It is important that the server machine have a common uid and gid space with the client machines.

SEE ALSO

sam_advise(3), sam_archive(3), sam_rearch(3), sam_exarchive(3), sam_unarchive(3), sam_unrearch(3), sam_damage(3), sam_undamage(3), sam_cancelstage(3), sam_closecat(3), sam_devstat(3), sam_devstr(3), sam_getcatalog(3), sam_lstat(3), sam_ndevstat(3), sam_opencat(3), sam_readrminfo(3), sam_release(3), sam_request(3), sam_restore_copy(3), sam_restore_file(3), sam_segment(3), sam_setfa(3), sam_ssum(3), sam_stage(3), sam_stat(3).

```
     sam_archive(3X), sam_closerpc(3X), sam_initrpc(3X),
     sam_lstat(3X), sam_release(3X), sam_stage(3X), sam_stat(3X).
```

sam_archive(3X)

```
NAME
     sam_archive - Sets archive attributes on a file or directory

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib -lsamrpc  -lnsl [ library ... ]

     #include "/opt/SUNWsamfs/include/samrpc.h"

     int sam_archive(const char *path, const char *ops);

DESCRIPTION
     This is the RPC-based version  of  sam_archive(3),  allowing
     archive  attributes  on a file or directory to be set from a
     remote machine.

     sam_archive(3X) sets archive attributes on a file or  direc-
     tory  by  sending  its request to the Sun QFS or SAM-QFS RPC
     server, sam-rpcd.

     A call to sam_initrpc(3X) must be issued before calling this
     routine.

RETURN VALUES
     Upon successful completion a value of 0 is returned.  Other-
     wise, 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).
```

sam_closerpc(3X)

```
NAME
     sam_closerpc - Performs RPC shutdown for Sun QFS and SAM-QFS
     RPC API library

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib -lsamrpc  -lnsl [ library ... ]
     #include "/opt/SUNWsamfs/include/samrpc.h"

     int sam_closerpc();
```

DESCRIPTION
     sam_closerpc() is the shutdown routine for the libsamrpc library. It destroys the RPC client handle and deallocates private data structures that were allocated with sam_initrpc().

RETURN VALUES
     Upon successful completion a value of 0 is returned. Otherwise, a value of -1 is returned.

SEE ALSO
     sam_initrpc(3X), sam_archive(3X), sam_release(3X), sam_stage(3X), sam_stat(3X).

sam_initrpc(3X)

NAME
     sam_initrpc - Performs RPC initialization for Sun QFS and SAM-QFS RPC API library

SYNOPSIS
```
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib -lsamrpc  -lnsl [ library ... ]

     #include "/opt/SUNWsamfs/include/samrpc.h"

     int sam_initrpc(char *rpchost);
```

DESCRIPTION
     sam_initrpc() is the initialization routine for the libsamrpc library. It finds the RPC entry for the Sun QFS or SAM-QFS server and creates an RPC client handle. In essence, this routine sets up the connection to the Sun QFS or SAM-QFS host machine, required for other API calls in the libsamrpc library.

     rpchost is the hostname of the Sun QFS or SAM-QFS host. If NULL, sam_initrpc() will check for an environment variable named SAMHOST. If such an environment variable exists, its setting will be taken for the hostname of the Sun QFS or SAM-QFS host, otherwise the built-in default, samhost, is used.

     sam_initrpc() gets the RPC entry (program number) using the program name samfs. This information (the RPC program name and number), and the hostname, is used to set up communication with the Sun QFS or SAM-QFS RPC API server process, sam-rpcd, which runs on the Sun QFS or SAM-QFS host machine.

RETURN VALUES
     Upon successful completion a value of 0 is returned. Otherwise, a value of -1 is returned and errno is set to indicate the error.

```
ERRORS
     sam_initrpc() fails if one or  more  of  the  following  are
     true:

     EADDRNOTAVAIL       No RPC entry for the program name  samfs
                         could be found.

SEE ALSO
     sam_closerpc(3X),     sam_archive(3X),      sam_release(3X),
     sam_stage(3X), sam_stat(3X).
```

sam_lstat(3X)

```
NAME
     sam_stat, sam_lstat - Gets file status over a  network  con-
     nection

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib -lsamrpc  -lnsl [ library ... ]

     #include "/opt/SUNWsamfs/include/stat.h"
     #include "/opt/SUNWsamfs/include/samrpc.h"

     int sam_stat(const char *path, struct sam_stat *buf);

     int sam_lstat(const char *path, struct sam_stat *buf);

DESCRIPTION
     These  are  the  RPC-based  versions  of   sam_stat(3)   and
     sam_lstat(3).

     sam_stat(3X) and sam_lstat(3X) get file status by sending  a
     request to the Sun QFS or SAM-QFS RPC server, sam-rpcd.

     If the server machine is different from the  local  machine,
     path must be an absolute path.  If the server machine is the
     local machine, path may be an absolute path or  relative  to
     the user's current working directory.

     A call to sam_initrpc(3X) must be issued before these calls.

RETURN VALUES
     Upon successful completion a value of 0 is returned.  Other-
     wise, 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_lstat(3), sam_stat(3).

     sam_closerpc(3X), sam_initrpc(3X).
```

sam_release(3X)

```
NAME
     sam_release - Sets release attributes on a file or directory

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib -lsamrpc  -lnsl [ library ... ]

     #include "/opt/SUNWsamfs/include/samrpc.h"

     int sam_release(const char *path, const char *ops);

DESCRIPTION
     This is  the  RPC-based  version  of  sam_release(3),  which
     allows release attributes to be set from a remote machine.

     sam_release(3X) sets release attributes on a file or  direc-
     tory  by  sending  its request to the Sun QFS or SAM-QFS RPC
     server, sam-rpcd.

     A call to sam_initrpc(3X) must be issued before calling this
     routine.

RETURN VALUES
     Upon successful completion a value of 0 is returned.  Other-
     wise, 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).
```

sam_segment(3X)

```
NAME
     sam_segment - Sets segment attributes on a file or directory

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib -lsamrpc  -lnsl [ library ... ]

     #include "/opt/SUNWsamfs/include/samrpc.h"
     int sam_segment(const char *path, const char *ops);

DESCRIPTION
     This is  the  RPC-based  version  of  sam_segment(3),  which
     allows file attributes to be set from a remote machine.

     sam_segment(3X) sets segment attributes on a file or  direc-
     tory  by  sending  its  request  to  the  Sun QFS or SAM-QFS
     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.  Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.

ERRORS
     EDESTADDRREQ        sam_initrpc was not successfully called,
                         as required, before making this call.

SEE ALSO
     segment(1).

     sam_segment(3).

     sam_initrpc(3X), sam_closerpc(3X).
```

sam_setfa(3X

```
NAME
     sam_setfa - Sets attributes on a file or directory

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib -lsamrpc  -lnsl [ library ... ]

     #include "/opt/SUNWsamfs/include/samrpc.h"

     int sam_setfa(const char *path, const char *ops);

DESCRIPTION
     This is the RPC-based version of sam_setfa(3), which  allows
     file attributes to be set from a remote machine.

     sam_setfa(3X) sets attributes on  a  file  or  directory  by
     sending  its  request to the Sun QFS or SAM-QFS server, sam-
     rpcd.

     A call to sam_initrpc(3X) must be issued before calling this
     routine.

RETURN VALUES
     Upon successful completion a value of 0 is returned.  Other-
     wise, a value of -1 is returned and errno is set to indicate
     the error.
```

```
ERRORS
     EDESTADDRREQ        sam_initrpc was not successfully called,
                         as required, before making this call.

     EINVAL              A valid filename was not provided.

     EPERM               The calling process is not superuser  or
                         the owner of the file specified.

     EROFS               The file system is a read-only file sys-
                         tem.

SEE ALSO
     setfa(1).

     sam_setfa(3).

     sam_initrpc(3X), sam_closerpc(3X).
```

sam_stage(3X)

```
NAME
     sam_stage - Sets stage attributes on a file

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib -lsamrpc  -lnsl [ library ... ]

     #include "/opt/SUNWsamfs/include/samrpc.h"

     int sam_stage(const char *path, const char *ops);

DESCRIPTION
     This is the RPC-based version of sam_stage(3), which  allows
     stage attributes on a file to be set from a remote machine.

     sam_stage(3x) sets stage attributes on a file  or  directory
     by sending its request to the Sun QFS or SAM-QFS RPC server,
     sam-rpcd.

     A call to sam_initrpc(3X) must be issued before calling this
     routine.

RETURN VALUES
     Upon successful completion, a value of 0 is returned.   Oth-
     erwise,  a value of -1 is returned and errno is set to indi-
     cate the error.

ERRORS
     EDESTADDRREQ        sam_initrpc was not successfully called,
                         as required, before making this call.

SEE ALSO
     stage(1).

     sam-stagealld(1M).
     sam_stage(3).

     sam_initrpc(3X), sam_closerpc(3X).
```

sam_stat(3X)

```
NAME
     sam_stat, sam_lstat - Gets file status over a  network  con-
     nection

SYNOPSIS
     cc [ flag  ... ] file  ... -L/opt/SUNWsamfs/lib -lsamrpc  -lnsl [ library ... ]

     #include "/opt/SUNWsamfs/include/stat.h"
     #include "/opt/SUNWsamfs/include/samrpc.h"

     int sam_stat(const char *path, struct sam_stat *buf);

     int sam_lstat(const char *path, struct sam_stat *buf);

DESCRIPTION
     These  are  the  RPC-based  versions  of   sam_stat(3)   and
     sam_lstat(3).

     sam_stat(3X) and sam_lstat(3X) get file status by sending  a
     request to the Sun QFS or SAM-QFS RPC server, sam-rpcd.

     If the server machine is different from the  local  machine,
     path must be an absolute path.  If the server machine is the
     local machine, path may be an absolute path or  relative  to
     the user's current working directory.

     A call to sam_initrpc(3X) must be issued before these calls.

RETURN VALUES
     Upon successful completion a value of 0 is returned.  Other-
     wise, 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_lstat(3), sam_stat(3).

     sam_closerpc(3X), sam_initrpc(3X).
```

CHAPTER 5

File Formats (Man Pages Section 4)

This chapter provides the section 4 man pages for Sun QFS and Sun Storage Archive Manager.

archiver.cmd(4)

```
NAME
     archiver.cmd - SAM-QFS archiver commands file

SYNOPSIS
     /etc/opt/SUNWsamfs/archiver.cmd

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     Commands  for  controlling  the  archiver  are   read   from
     /etc/opt/SUNWsamfs/archiver.cmd,   which   is  the  archiver
     commands file.  The archiver.cmd  file  must  be  free  from
     errors, or the archiver does not execute.

     Use the archiver -lv command to check the archiver.cmd  file
     for syntax errors. When it is free from errors, use the

     samd config

     command to reconfigure the daemons.

     Archive  Sets  and  associated  media  are  defined  in  the
     archiver  command file.  Archive Sets are the mechanism that
     the archiver uses to direct files in a samfs file system  to
     media during archiving.

     All files in the file system are members of one and only one
     Archive   Set.   Characteristics  of  a  file  are  used  to
     determine Archive Set membership.  All files in  an  Archive
     Set are copied to the media associated with the Archive Set.
     The Archive Set name is simply a synonym for a collection of
     media volumes.
```

Files are written to the media in an Archive File, which is written in tar format. The combination of the Archive Set and the tar format results in an operation that is just like using the command find to select files for the tar command.

In addition, the meta data (directories, the indices of segmented files, and the removable media information), are assigned to an Archive Set to be copied to media. The Archive Set name is the name of the file system. (See mcf(4)).

For segmented files, the archivable unit is the segment, not the entire file, so the properties and priorities apply to the segments themselves rather than to the entire file. The index of a segmented file contains no user data and so is assigned to the meta data archive set.

Symbolic links are considered data files for archival purposes.

Each Archive Set may have up to four archive copies defined. The copies provide duplication of files on different media. Copies are selected by the Archive Age of a file.

The archiver command file consists of directive lines. In this man page, the archiver directives are separated into the following sections and subsections:

```
     General Directives section
     Archive Set Assignments section
     Archive Copy Definitions section
     Archive Set Copy Parameters section
        Archive Set Copy Parameters - General
        Archive Set Copy Parameters - Priority
        Archive Set Copy Parameters - Scheduling
        Archive Set Copy Parameters - Recycling
     VSN Pool Definitions section
     VSN Associations section
```

Each of these lines consists of one or more fields separated by white space. Leading white space is ignored. Everything after a '#' character is ignored. Lines may be continued by using '\' as the last character on the line.

All parameter settings and Archive Set definitions apply to all file systems (global) until a file system directive is encountered. Thereafter, the settings and definitions apply only to the named file system (local). The directives archmax, bufsize, drives, notify, and ovflmin can only be global and hence are not allowed after the first fs= directive.

GENERAL DIRECTIVES SECTION

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

archmax = media target_size

Set the Archive File maximum size for media media to target_size. Files to be archived will be placed on

```
          the media in a single Archive File of length less  than
          or  equal  to target_size.  If a single file is greater
          than target_size, then this restriction does not apply.

          Sizes appropriate to the media are used by default. The
          default  size for STK Titanium, all LTO, IBM TS1120 and
          IBM 3592 media is 22GB. The default size for  SKT  9940
          media  is  11GB. The default size for STK 9840 media is
          4GB.  The default size for linear  tape  is  11GB.  The
          default  size  for  all  other  tape  media is 8GB. The

          default size for disk  is  1G.  The  default  size  for
          optical media is 1GB.

     archivemeta = state
          Set the meta data archiving state on or off.  state may
          be  "on"  or  "off".   Meta  data  archiving  is off by
          default.

     background_interval = time
          Set the interval between background scans to time.

          The default is 24 hours.  If  time  is  a  multiple  of
          days,  the  background  scan  will be performed at the
          background_time .

     background_time = hhmm
          Set the time of day for the  background  scan  to  hhmm
          local time.

          The default 0000 (midnight).

     dircache_size = size
          Set the maximum size of the directory  cache  to  size.
          The  directory  cache  for name lookups will not exceed
          this size.  If the  file  system  contains  very  large
          directories,   increasing   this   value   may   help
          performance.  The minimum value is 8M and  the  maximum
          is 512M.

          The default is 64M.

     bufsize =  media buffer_size [ lock ]
          Set  the  archive  buffer  size  for  media  media   to
          buffer_size  *  dev_blksize,  and (optionally) lock the
          buffer.

          For media, specify a valid media type from the list  on
          the mcf(4) man page.

          For buffer_size, specify a number from 2 through  8192.
          The  default  is  16.   This value is multiplied by the
          dev_blksize value for the media type, and the resulting
          buffer  size is used.  The dev_blksize can be specified
          in the defaults.conf file.

          The lock argument indicates whether or not the archiver
          should  use  locked buffers when making archive copies.
          If lock is specified, the archiver sets file  locks  on
          the  archive  buffer  in memory for the duration of the
          sam-arcopy(1M)  operation.   This  avoids  paging   the
          buffer,  and  it can provide a performance improvement.
          The lock argument should be  specified  only  on  large

          systems  with large amounts of memory.  If insufficient
          memory is present,  it  can  cause  an  out  of  memory
          condition.  The  lock  argument  is  effective  only if
          direct I/O is enabled for the file being archived.   By
          default, lock is not specified and the file system sets
          the locks on all direct I/O  buffers,  including  those
          for archiving.

          This directive can also be specified on an archive  set
          basis  by  placing  the  -bufsize=buffer_size and -lock
          directives between  params  and  endparams  directives.
          For    more    information    on    this,    see    the
          -bufsize=buffer_size  and  -lock  directives  mentioned
          later on this man page.

          For  more   information   on   dev_blksize,   see   the
          defaults.conf   man  page.   For  more  information  on
          enabling direct I/O, see the  setfa(1)  man  page,  the
          sam_setfa(3)   library   routine   man   page,  or  the
          -o forcedirectio  option  on  the  mount_samfs(1M)  man
          page.

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

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

          Example:
               drives = gr50 3

     examine = method
          Set the  file  system  examination  method  to  method.
          Files  in  a  file system are examined using the method
          defined by this directive.  method may be one of:

          scan        Scan the file  system  in  the  traditional
                      manner.   The  first  scan  is  a directory
                      scan, all successive scans are inode scans.

          scandirs    All scans are directory scans.

          scaninodes  All scans are inode scans.

          noscan      No periodic scans are performed.  Files are
                      examined when they change.

          The default examine method is noscan.
```

```
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 directive may  be
     followed  by copy definitions to define multiple copies
     for the file system meta data.

     The defaults are no local definitions and  one  archive
     copy for the file system data.

interval = time
     Set the interval between archive operations to time.

     The default time is 10 minutes.

logfile = filename
     Set the name of the  archiver  log  file  to  filename,
     specified  as  an  absolute pathname.  The archiver log
     file contains a line for each file archived.  The  line
     contains  information  about the file that includes the
     date, time, media, volume, Archive Set, and the name of
     the  file.  Note that it is possible to have a separate
     log file for each file system (by placing a "logfile ="
     definition after a "fs =" definition).

     The default is no log file.

notify = filename
     Set the name of the archiver event notification  script
     file  to  filename.   This  file  is  executed  by  the
     archiver to allow the system administrator  to  process
     various  events in a site specific fashion.  The script
     is called with a keyword for the first  argument.   The
     keywords   are:   emerg,  alert,  crit,  err,  warning,
     notice, info,  and  debug.   Additional  arguments  are
     described in the default script.

     The    name    of    the     default     script     is:
     /etc/opt/SUNWsamfs/scripts/archiver.sh.

ovflmin = media minimum_size
     Set the minimum size of a file which will require  more
     than one volume for media media to minimum_size.  Files
     to be archived that are smaller than this size will  be
     placed  on  only  a  single volume of the media.  Files
     that are larger than this size will be  allowed  to  be
     written to multiple volumes.

     If not specified, volume overflow will not take place.

scanlist_squash = state
     Control the sam-arfind scanlist  consolidation.   state
     may  be  "on"  or  "off".   If  files  in  two  or more
     subdirectories with the same parent directory  need  to
     be scanned by sam-arfind at a much later time, the scan
     entries can be consolidated if state is on.   The  sam-
     arfind scanlist consolidation is off by default.

setarchdone = state
```

```
          Control the changing of the  state  of  the  'archdone'
          flag  for  a  file  when  the  file is examined by sam-
          arfind.  state may be on or off.

          When all archive copies for a file have been made,  the
          archdone  flag is set for that file to indicate that no
          further archive action is required.  The archdone  flag
          is  used  by the archiver only during an inodes scan to
          avoid looking up the path name for the inode.   Setting
          archdone for files that will never be archived can be a
          time  consuming  operation   during   directory   scans
          impacting   performance   when  large  directories  are
          scanned.  Therefore, this will no  longer  be  done  by
          default.   To  get the previous behavior, set the state
          to on.

          The default  value  of  state  is  off  for  examine  =
          scandirs and examine = noscan.

          This option  does  not  affect  setting  the  state  of
          archdone when archive copies are made.

     wait The archiver will not begin archiving until it receives
          a  start  command from archiver, samu, or samcmd.  This
          is  a  mechanism  to  allow  other  activities  to   be
          performed  before  archiving  begins.   The wait may be
          applied globally or to one or more file systems.

          The default is no waiting.   However,  if  archiver.cmd
          does not exist then the default is to wait.

     timeout = [ operation | media ] time
          External events may cause the archiving I/O  operations
          to  stop  for  indefinite  periods  of time.  This will
          hamper timely archiving of other  files  that  are  not
          affected by the external delays.  Timeouts are provided
          for the operations that may get stopped.   The  timeout
          values  for  the  write operation may also be specified
          for individual media.

          operation may be one of:

          read        Reading the file from the disk.  Default  =

                      1  minute.  This timeout will be set to the
                      same value as the write timeout (default 15
                      minutes) when offline_copy = direct.

          request     Requesting the archive media.  Default = 15
                      minutes.

          stage       Staging the file to be archived.  Default =
                      0 (no timeout).

          write       Writing to the archive media.  Default = 15
                      minutes   for   removable   archive   media.
                      Default = 0 (no timeout) for  disk  archive
                      media.
```

ARCHIVE SET ASSIGNMENTS SECTION

Archive Set assignments are made by describing the characteristics of the files that should belong to the set. The statements that do this are patterned after the find(1) command. The Archive Set name is the first field, followed by the path relative to the SAM-QFS file system mount point. The path may be enclosed in quotation mark characters, for instance, "project/gifs". Within the quoted string, the usual character escapes are allowed, including octal character value.

The remaining fields are either the file characteristics for membership in the set, or controls for the set.

It is possible that the choice of file characteristics for several Archive Sets will result in ambiguous set membership. These situations are resolved in the following manner:

1. The Archive Set with the earliest definition in the command file is chosen.

2. Local definitions for the file system are chosen before the global definitions.

These rules imply that more restrictive Archive Set definitions should be closer to the beginning of the command file.

It is also possible to use the same Archive Set name for several different file characteristics. An example would assign files that are owned by several users into a single Archive Set.

Assigning files to a special archive set called no_archive prevents files from being archived. This can be useful for temporary files. The no_archive archive set assignment

definition must be a local definition to be effective.

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

You can specify one or more of the following file characteristics:

-user uname
    Include files belonging to user uname.

-group gname
    Include files belonging to group gname.

-minsize size
    Include files greater than or equal to size. size may be specified with the suffices ’b’, ’k’, ’M’, ’G’, and ’T’, for bytes, kilobytes, megabytes, gigabytes, and terabytes.

-maxsize size

```
          Include files less than size.

     -name regular_expression
          Include   files   with   full    paths    that    match
          regular_expression.   The regular expression is limited
          to 255 characters.

     -access age
          Include files whose access time is older than age.  The
          age  may  be specified with the suffixes 's', 'm', 'h',
          'd', 'w' and 'y', for seconds,  minutes,  hours,  days,
          weeks and years.

     -nftv
          By default, the access and modification times of  files
          are  validated  to  assure that these times are greater
          than or equal to the file creation time, and less  than
          or  equal  to  the  time at which the file is examined.
          This is to provide proper archiving and unarchiving.

          For files that have been "migrated" into  a  directory,
          this  may  not  be the desired behavior.  The -nftv (no
          file time validation) parameter may be used to  prevent
          the  validation  of  file access and modification times
          for files that are in the archive set defined by  these
          definitions.

     -after date_time
          Include files that have been created or modified  since
          date_time.    date_time   is   in  the  form  "YYYY-MM-
          DD[Thh:mm:ss][Z]" (ISO 8601 format).

          If the  time  portion  is  not  specified,  'Thh:mm:ss'
          missing,  it  is assumed to be 00:00:00.  If the 'Z' is
          present, date_time is UTC, otherwise it is local.

          Examples:
           2005-10-08T12:15:47
           2005-10-08
           2005-10-08T17:15:47Z

     Example:
          When controlling archiving for a specific  file  system
          (using  the fs = fsname directive), directives local to
          the file system level are evaluated before  the  global
          directives.   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
```

SETTING FILE ATTRIBUTES

The following directives are available to set file attributes:

-release attributes
: Set the release attributes (see release(1)) for all files matching the file characteristics on this Archive Set definition. attributes may be any of 'a' always, 'd' reset to default, 'n' never, 'p' partial or 'sxx' partial size 'xx'.

-stage attributes
: Set the stage attributes (see stage(1)) for all files matching the file characteristics on this Archive Set definition. attributes may be any of 'a' associative, 'd' reset to default, or 'n' never.

ARCHIVE COPY DEFINITIONS SECTION

The Archive Copy definitions determine when the archive copies are made for the files matching file characteristics. These definitions consist of lines beginning with a digit. This digit is the copy number.

The first fields after the copy number are the option flags as described below:

-release
: This causes the cache disk space for the files to be released immediately after the copy is made.

-norelease
: This flag may be used to prevent automatic release of cache disk space until all copies marked with this flag are made. The -norelease option makes the archiver set eligible to be released after all copies have been archived, but the files will not be released until the releaser is invoked and selects them as release

candidates. Using this flag on just one copy will have no effect on automatic release.

The combination of -release and -norelease will cause the archiver to release the file when all the copies having this combination are made. With this usage, the archive set is released immediately, rather than waiting for the releaser to be invoked, as is the case with the -norelease option alone.

If the -release option is used on a copy that does not have the -norelease option set, the file will get released when that copy is made, overriding the effect of any -norelease usage on other copies.

The next field is the Archive Age of the file when the archive copy is made. The age may be specified with the suffixes 's', 'm', 'h', 'd', 'w' and 'y', for seconds,

minutes, hours, days, weeks and years. The default Archive Age is 4 minutes.

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

ARCHIVE SET COPY PARAMETERS SECTION

Archive Set parameters may be set after all Archive Sets are defined. The beginning of this section is noted by the directive params. The section is ended by the end of the archiver command file or the directive 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 directive. If the allsets is specified without a copy number, the parameters apply to all Archive Set Copies. If specified with a copy number, the parameters apply to only those Archive Set Copies with the same copy number. All allsets directives must occur before those for any actual Archive Set Copies.

Note: All parameter default values are 0 or none unless otherwise specified.

Example:
```
     allsets -sort path
     allsets.1 -drives 3
     allsets.2 -drives 2
```

All Archive Set Copies are assigned the -sort path parameter. All Archive Set Copy 1 will use 3 drives. All Archive Set Copy 2 will use 2 drives.

If an archive copy of a file is being rearchived, an

```
   internal Archive Set Copy is used for scheduling the archive
   operation.  It is called a Rearchive Set Copy, and uses  the
   archive  parameters  from  the  actual Archive Set Copy.  If
   desired, the Archive Set parameters may  be  set  using  the
   Archive  Set  Copy  name followed by the character 'R'.  The
   Rearchive Set Copy allows the users to  differentiate  'new'
   and  rearchive  operations, and use different parameters for
   each operation.

   Example:
        archset.2 -drives 3

        archset.2R -drives 1 -priority -1000

        All 'new' archive copies are  written  using  up  to  3
        drives.   Rearchive  copies are limited to 1 drive, and
        have a lower priority than the 'new' copies.

   In addition, the  allsets.copy  forms  may  be  used.   (For
   example, allsets.copyR)

Archive Set Copy Parameters - General
   The general archive set copy parameters are as follows:

   -archmax target_size
        Set the Archive File maximum size for this Archive  Set
        to target_size.  Files to be archived will be placed on
        the media in a single Archive File of length less  than
        or  equal  to target_size.  If a single file is greater
        than target_size, then this restriction does not apply.

        If not specified, the archmax value for  the  media  is
        used.

   -bufsize = buffer_size
        Set  the  archive  buffer   size   to   buffer_size   *
        dev_blksize.   The  default  buffer_size  is 16.  Valid
        values are 2 through 8192.

        If not specified, the default buffer size value for the
        media is used.  This directive can also be specified as
        a global directive.  For more information on specifying
        an  archive  buffer  size,  see  the  bufsize  =  media
        buffer_size [lock] directive described on this man page
        in the GENERAL DIRECTIVES section.

   -directio state
        Set the file reading method for archival.  state may be
        "on"  or  "off".  The  reading performance of files for
        archival can be changed by using  this  parameter.   If
        users  are not reading files at the same time that they
        are  being  archived,  then  selecting  on  allows  the
        archiver  to  read  the  file  without using the system
        buffer cache and using pages that users might need.  In
        the  event  that users are reading files while they are
        being archived, then off may be a better choice because
        the  system  buffer cache will provide data to the user
        and the archiver.  The default is on.
```

```
-disk_archive diskvol(Obsolete)
     Defines  a  disk  archive  set.   This   parameter   is
     obsolete.   Disk  archive sets should be defined in the
     VSN associations or VSN pool definitions section.   For
     more information on disk archiving, see the Sun Storage

     Archive Manager Configuration and Administration Guide.

     All of the other Archive Set parameters work with  disk
     archiving  except:  -fillvsns, -ovflmin minimum_size, -
     reserve method, -tapenonstop.  None of these  cause  an
     error  if applied to an Archive Set that is assigned to
     disk archiving.

-drivemax max_size
     Set the multiple drives maximum size for  this  Archive
     Set   to  max_size.   When  the  -drives  parameter  is
     selected, the amount of data selected to be archived to
     each  drive  will  be  limited to max_size.  Using this
     parameter  can  result  in  better  drive  utilization,
     because  drives  can  take different amounts of time to
     archive files.

     The default is to not have this parameter set.

-drivemin min_size
     Set the multiple drives minimum size for  this  Archive
     Set   to  min_size.   When  the  -drives  parameter  is
     selected, multiple drives will be  used  only  if  more
     than  min_size  data  is  to  be archived at once.  The
     number of drives to be used in  parallel  will  be  the
     lesser  of  total_size  /  min_size  and  the number of
     drives specified by -drives.

     The default value is archmax.

-drives number
     Set the maximum number of drives to  use  when  writing
     the  archive  images  for  this  Archive  Set  Copy  to
     removable media.

     Segments are striped across  the  specified  number  of
     drives.  The segments are separated into number archive
     files.

     Example:
          set_name.3 -drives 3

     Allows the archiver to use up to 3 drives for archiving
     files in the archive set named set_name.3.

     If not specified, one drive will be used.

-fillvsns [ minfill ]
     The default action of the archiver is  to  utilize  all
     volumes  associated  with an Archive Set for archiving.
     When a group of files is to be  archived  at  the  same
     time, a volume with enough space for all the files will
```

```
          be selected for use.  This action may cause volumes  to
          not be filled to capacity.

          Selecting this parameter causes the archiver to attempt
          to  fill  volumes by separating the group of files into
          smaller groups.

          The optional minfill parameter  specifies  the  minimum
          free  space  that  a  volume  must  have in order to be
          included in the above calculation. minfill is specified
          as a file size.

          Example:
               -fillvsns 1G

          Volumes will be filled until they  have  less  than  1G
          free space, after which they are considered full.

     -lock
          Lock the archive copy buffer for the  duration  of  the
          sam-arcopy(1M)   operation.   The  -lock  directive  is
          effective only if direct I/O is enabled  for  the  file
          being  archived.   If  not  specified,  the file system
          controls the locks on  the  archive  copy  buffer.   By
          default, this directive is disabled.

          This directive  can  also  be  specified  as  a  global
          directive.   For  more  information  on controlling the
          archive  buffer  locks,  see  the   bufsize   =   media
          buffer_size [lock] directive described on this man page
          in the GENERAL DIRECTIVES section.

     -offline_copy method
          This parameter specifies the  method  to  be  used  for
          archiving  files  that are offline at the time archival
          is to be made.

          For  selecting  the  desired  offline  file   archiving
          method, method may be:

          none    Files are staged as needed for each archive tar
                  file before copying to the archive volume.

          direct  Direct copy.   Copy  files  directly  from  the
                  offline  volume  to  the archive volume without
                  using the cache.  Source volume and destination
                  volume   are   different  and  two  drives  are
                  available.  For best performance in this  mode,
                  you  should  increase  the  file  system  mount
                  parameter "stage_n_window" from its default  of
                  256k.

          stageahead
                  Stage the  next  archive  tar  file  while  the
                  current  archive  tar  file  is  written to the
                  destination. With this method, one archive  tar
                  file  is  created  on  one  tape drive (or disk
                  archive) while  the  offline  files  needed  to
                  create  the  next  archive  tar  file are being
```

```
                staged  from  another  tape  drive   (or   disk
                archive).  Two drives are available and room is
                available on cache for all files in one archive
                tar file.

        stageall
                Stage all files before archiving.  Use only one
                drive,  and  room is available on cache for all
                files.

   -ovflmin minimum_size
        Set the minimum size of a file that will  require  more
        than  one  volume  in this Archive Set to minimum_size.
        Files to be archived that are smaller  than  this  size
        will  be  placed  on only a single volume of the media.
        Files that are this size or larger will be  allowed  to
        overflow one volume to at least one additional volume.

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

   -rearch_stage_copy copy_number
        Use  copy_number  for  staging  an  offline  copy  when
        rearchiving  the  copy  defined by the Archive Set.  By
        default, the file will be staged from  the  copy  being
        rearchived.   This option can be used if the copy being
        rearchived is not available or copy_number  is  located
        on a faster media.

   -reserve [ set | dir | user | group | fs ]
        This parameter specifies  that  the  volumes  used  for
        archiving files in this Archive Set are "reserved".  If
        this option is not used, Archive Sets are mixed on  the
        media  specified.   This  option  specifies  that  each
        archive  set   has   unique   volumes.    A   so-called
        "ReserveName"  is  assigned  to  volumes  as  they  are
        selected for use by the Archive Set.   The  ReserveName
        has  three  components:   Archive  Set, Owner, and file
        system.  The keyword set  activates  the  Archive  Set.
        The keyword fs activates the file system 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
        immediately  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.

   -rsort method

   -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.  If no method is specified,
          path sorting is performed.  If -rsort is used, the sort
          is performed reversing the order specified by method.

          For selecting the  sort,  method  can  be  one  of  the
          following:

          age     Sort   each   Archive   File    by    ascending
                  modification   time.    The  oldest  files  are
                  archived first.

          none    No sorting of the Archive  File  is  performed.
                  Files  are 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.    The   higher  priority  files  are
                  archived first.

          size    Sort each Archive File by ascending file  size.
                  The  smallest  files  are  archived first.  The
                  largest files are archived last.

     -tapenonstop
          When files are archived to tape,  the  default  writing
          mechanism  closes  the  removable  media  tape  file in
          between each Archive File.  This action causes the tape
          subsystem to write a TapeMark followed by an EOF1 label
          and two TapeMarks.  Before another Archive File can  be
          written, the tape must be positioned backwards over the
          EOF1 label.

          Using the tapenonstop parameter causes the archiver  to
          not  close  the  removable media tape file between each
          Archive File, and write a Tape  Mark  to  separate  the
          Archive  Files.   This  speeds writing Archive Files to
          tape.  The tape cannot be unloaded in  between  Archive
          Files.

  Archive Set Copy Parameters - Priority
     The following parameters allow you to configure  a  priority
     system  for  archiving  files.   In  the  following priority
     parameters, the values are floating-point numbers such that
     -3.400000000E+38 < value < 3.402823466E+38.

     -priority age value
          Set the "Archive Age" property multiplier for files  in
          this Archive Set to value.

     -priority archive_immediate value
          Set the "Archive  immediate"  property  multiplier  for
          files in this Archive Set to value.
```

```
   -priority archive_overflow value
        Set the "Multiple archive volumes" property  multiplier
        for files in this Archive Set to value.

   -priority archive_loaded value
        Set the "Archive volume loaded" property multiplier for
        files in this Archive Set to value.

   -priority copy1 value
        Set the "Copy 1" property multiplier for files in  this
        Archive Set to value.

   -priority copy2 value
        Set the "Copy 2" property multiplier for files in  this
        Archive Set to value.

   -priority copy3 value
        Set the "Copy 3" property multiplier for files in  this
        Archive Set to value.

   -priority copy4 value
        Set the "Copy 4" property multiplier for files in  this
        Archive Set to value.

   -priority copies value
        Set the "Copies made" property multiplier for files  in
        this Archive Set to value.

   -priority offline value
        Set the "File off line" property multiplier  for  files
        in this Archive Set to value.

   -priority queuewait value
        Set the "Queue wait" property multiplier for  files  in
        this Archive Set to value.

   -priority rearchive value
        Set the "Rearchive" property multiplier  for  files  in
        this Archive Set to value.

   -priority reqrelease value
        Set the "Required for release" property multiplier  for
        files in this Archive Set to value.

   -priority size value
        Set the "File size" property multiplier  for  files  in
        this Archive Set to value.

   -priority stage_loaded value
        Set the "Stage volume loaded" property  multiplier  for
        files in this Archive Set to value.

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

Archive Set Copy Parameters - Scheduling
   As files are identified to be archived, they are placed in a
```

```
list  known  as  an Archive Request.  The Archive Request is
scheduled for archival at the end of  a  file  system  scan.
The  following  archive set parameters control the archiving
workload and assure timely archival of files:

-queue_time_limit time
     Set the schedule  queue  time  limit  for  the  Archive
     Request  to  time.   At  the  end  of the time limit, a
     notification  message  will  be  sent  once  to   alert
     monitoring  entities  that  the ArchReq has been in the
     schedule queue longer than the time limit.

-startage time
     Set the interval between the first file to be  archived
     in  the  Archive  Request and the start of archiving to
     time.  This allows time  to  accumulate  archival  work
     after  the  first file has been scheduled for archival.
     The default is set to two hours.

-startcount count
     Set the start archiving  file  count  to  count.   When
     count  files  have  been identified for archival in the
     Archive Request, the archival  operation  begins.   The
     default is set to 500,000.

-startsize size

     Set the minimum total size of all files to be  archived
     after  the  first  file  to  be archived in the Archive
     Request  to  size  (in   bytes).    This   allows   the
     accumulation  of archival work to be based on the total
     size  of  the  files  that  have  been  scheduled   for
     archival.   The  default  is set to 90% of the -archmax
     value.

If more than one of -startage,  -startcount,  or  -startsize
are  specified,  the  first condition encountered starts the
archival operation.

If  neither  -startage,  -startcount,  nor  -startsize   are
specified,  the  archive  request  is scheduled based on the
examine=method directive, as follows:

o  If examine = scan | scaninodes |  scandirs,  the  archive
   request  is scheduled for archiving after the file system
   scan.  Note that examine = noscan is the default.

o  If examine = noscan, the default values are as follows:
    startage 2 hours
    startcount 500,000
    startsize 90% of archmax

The  -startage,  -startcount,  and   -startsize   directives
optimize archive timeliness versus archive work done.  These
values override the examine=method specification, if any.

Example 1.  If it takes an  hour  to  create  files  for  an
Archive  Set  that  uses  -sort  path,  then you can specify
-startage 1h  ensure  that  all  files  are  created  before
```

```
     scheduling the Archive Request.

     Example 2.  You can specify -startsize 150G  to  direct  the
     archiver to wait until 150 gigabytes of data are ready to be
     archived in an Archive Set.

     Example 3.  If you know that 3000 files  will  be  generated
     for  archival,  then specify -startcount 3000 to ensure that
     the files get archived together.

  Archive Set Copy Parameters - Recycling
     The following archive set parameters  control  recycling  by
     archive  set.   If  none of the following parameters are set
     for an archive set and the name of the archive  set  is  not
     specified  on  the  recycler's command line, the archive set
     will not be recycled.  Volumes which comprise  that  archive
     set  (unless  also  assigned to other archive sets) could be
     recycled as part of recycling  the  library  which  contains
     them.

     -recycle_dataquantity size
          This option sets a limit of size bytes on the amount of
          data  the  recycler will schedule for rearchiving so as
          to clear volumes of useful data.  Note that the  actual
          number  of  volumes  selected for recycling may also be
          dependant  on  the  -recycle_vsncount  parameter.   The
          default is 1 gigabyte (1G).

     -recycle_hwm percent
          This option sets the high  water  mark  (hwm)  for  the
          archive  set.   The hwm is expressed as a percentage of
          the total capacity of the volumes associated  with  the
          archive  set.   When  the  utilization of those volumes
          exceeds percent, the recycler will begin to recycle the
          archive  set.  The  default  is  95%.   This  option is
          ignored for disk media recycling.

     -recycle_ignore
          This option inhibits the recycler from  recycling  this
          archive set.  All recycling processing occurs as usual,
          except any media selected to  recycle  are  not  marked
          "recycle".   This allows the recycler's choice of media
          to recycle to be observed, without  actually  recycling
          any media.

     -recycle_mailaddr mail-address
          This  option  specifies  an  email  address  to   which
          informational messages should be sent when this archive
          set is recycled. The default is not to send any mail.

     -recycle_mingain percent
          This option limits selection of volumes  for  recycling
          to  those  which  would  increase  their  free space by
          percent or  more.   Volumes  not  meeting  the  mingain
          parameter are not recycled. The default is 50%.

     -recycle_vsncount count
          This option sets a limit of  count  on  the  number  of
          volumes  the  recycler will schedule for rearchiving so
```

as to clear volumes of useful data. Note that the actual number of volumes selected for recycling may also be dependant on the -recycle_dataquantity parameter. The default is 1. This option is ignored for disk media recycling.

-recycle_minobs percent
: This option is used to set a threshold for the recycler's rearchiving process. When the percentage of obsolete files within an archived tar file on the disk reaches this threshold, the recycler begins moving the valid files from the archive into a new tar file. Once all of the valid files have been moved, the original

  tar file is marked as a candidate to be removed from the disk archive. This option is ignored for removable media recycling. The default is 50%.

-unarchage time_ref
: Set the Unarchive Age computation time reference for this archive set to time_ref. The age of the files will be computed for unarchiving a copy from this time reference. For selecting the desired time reference, time_ref may be:

access
: The age of files for unarchiving a copy is computed from the access time of the file.

modify
: The age of files for unarchiving a copy is computed from the modification time of the file.

  The default time_ref is access.

VSN POOL DEFINITIONS SECTION

Collections of volumes may be defined in this section. The beginning of the section is noted by the directive vsnpools. The section is ended by the end of the archiver command file or the directive endvsnpools.

A VSN pool definition requires at least three fields: the pool name, the media type, and at least one VSN.

The media type is the two character mnemonic as described in the mcf(4) man page. The dk or cb identifiers can be used to define a disk archive set. For more information on disk archiving, see the Sun Storage Archive Manager Configuration and Administration Guide.

VSNs are regular expressions as defined in regcmp(3C).

VSN ASSOCIATIONS SECTION

VSN associations are defined after all archive sets are defined. The beginning of the section is noted by the directive vsns. The section is ended by the end of the archiver command file or the directive 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 '.'.

VSN associations may be set for all archive sets by using the pseudo Archive Set Copy allsets for the directive. If

the allsets is specified without a copy number, the VSNs apply to all Archive Set Copies. If specified with a copy number, the VSNs apply to only those Archive Set Copies with the same copy number. All allsets directives must occur before those for any actual Archive Set Copies.

If an archive copy of a file is being rearchived, the Rearchive Set Copy uses the VSN associations from the actual Archive Set Copy. If desired, the VSN associations may be set using the Archive Set Copy name followed by the character 'R'. The Rearchive Set Copy allows the users to differentiate 'new' and rearchive operations, and use different VSNs for each operation.

The media type is the two character mnemonic as described in the mcf(4) man page.

VSNs are regular expressions as defined in regcmp(3C). 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(3C). The compiled form is saved with the Archive Set Copy definition. When a volume is needed for an Archive Set Copy, each VSN of each library or manual drive that has sufficient space and is allowed to be used for archives, is used as the "subject" argument to regex(3C). The archive set copy vsn expressions are used as the "re" argument to regex(3C). If regex(3C) returns with a successful match, the volume is used for the archive set copy.

Example:
```
     set_name.3 mo optic.*
```

Assigns all files in set_name.3 to the mo media with VSNs beginning with optic.

VSN associations may be defined for all archive sets by using the pseudo Archive Set Copy allsets for the directive. If the allsets is specified without a copy number, the VSN associations apply to all Archive Set Copies. If specified with a copy number, the VSN associations apply to only those Archive Set Copies with the same copy number. All allsets directives must occur before those for any actual Archive Set Copies.

SEE ALSO

release(1), stage(1).

archiver(1M), archiver.sh(1M), sam-archiverd(1M), sam-arcopy(1M), sam-arfind(1M), sam-recycler(1M).

```
     regcmp(3C).

     diskvols.conf(4), mcf(4).
```

defaults.conf(4)

```
NAME
     defaults.conf - Set default values for Sun QFS and SAM-QFS
     software

SYNOPSIS
     /etc/opt/SUNWsamfs/defaults.conf

AVAILABILITY
     SUNWqfs
     SUNWsamfs

DESCRIPTION
     The defaults configuration file allows the site to set
     certain default values within the Sun QFS and Sun Storage
     Archive Manager (SAM-QFS) environments.  The defaults.conf
     file is read when sam-fsd is started.  It may be changed at
     any time while sam-fsd is running.  The changes will take
     place when sam-fsd is restarted, or sent the signal SIGHUP.
     Temporary changes to the environment values can be made
     using the samset(1M) command.

     The defaults.conf file consists of directive lines that are
     separated into two sections, the environment variable
     section and the trace file control section.

  Environment variables.
     The commands for the environment section of the file
     consists of a list of keyword = value pairs that set
     site-definable defaults.  All keyword and value entries are
     case-sensitive and must be entered as shown.  Values can be
     either unquoted strings (if string values are expected) or
     integers in decimal (123), octal, (0123) or hex (0x123)
     format.

     The keywords and their expected arguments are as follows:

     attended = yes | no
               If attended = yes, it is assumed that an operator
               is available to mount media that is not flagged as
               unavailable by the historian; the default is yes.
               If attended = no, any request for media known to
               the historian is rejected unless it is already
               mounted.

     debug = options
               Sets the default for the debug flags used by the
               Sun QFS and SAM-QFS daemons for logging messages.
               For options, specify a space-separated list of
               debug options from the list of possible options
```

```
                    described on the samset(1M) man page.  The default
                    is logging.

          devlog = eq_number [ event ... ]
                    Manipulates the device log event flags for the
                    device specified by Equipment Number eq_number.
                    The eq_number must be either the keyword all (to
                    specify all devices) or must match an Equipment
                    Number from the mcf file.

                    The device log event flags control the events that
                    get written to the device log files.  For the list
                    of possible event arguments, see the samset(1M)
                    man page.  To specify more than one event,
                    separate the events in the list with space
                    characters.  The default is err retry syserr date.

          dev_blksize = size
                    Specifies the default block size for tapes of type
                    dev.  For size, specify 16, 32, 64, 128, 256, 512,
                    1024, or 2048.  The size value is multiplied by
                    1024 to arrive at the actual block size.

                    For information on supported dev arguments and for
                    information on the default released block sizes
                    for various media, see the mcf(4) man page.

                    The default is used when no size is specified or
                    during automatic labeling when labels = barcodes
                    has been specified.  For information on how the
                    default can be overridden when manually labeling a
                    tape, see the tplabel(1M) man page.

          dev_delay = seconds
                    Specifies the dismount time, in seconds, for
                    device type dev.  After a cartridge is loaded onto
                    this device type, this time must elapse before the
                    cartridge unloaded and another cartridge is
                    loaded.  By default, dev_delay = 30.  For
                    information on supported dev arguments, see the
                    mcf(4) man page.

          dev_position_timeout = seconds
                    Specifies the timeout value, in seconds, to be
                    used during tape positioning for device type dev.
                    During most tape positioning command processing
                    (such as locate and space) this is the maximum
                    amount of time to wait for the command to
                    complete.  For information on the default values,
                    see the example file
                    (/opt/SUNWsamfs/examples/defaults.conf) supplied
                    with your software.  Any device not in the example
                    file defaults to 1800 seconds.  For information on
                    supported dev arguments, see the mcf(4) man page.

          dev_unload = seconds
                    Specifies the unload wait time, in seconds, for
                    device type dev.  This is the amount of time that
                    the library daemons wait after the device driver
```

returns from a SCSI unload command. This interval gives the library time to eject the media, open the door, and perform other actions before the daemon commands the library to remove the media. The seconds specified should the longest time needed for the worst-case library configured. For information on the default values, see the example file (/opt/SUNWsamfs/examples/defaults.conf) supplied with your software. Any device not in the example file defaults to 0 seconds. For information on supported dev arguments, see the mcf(4) man page.

div = value

Enables or disables the STK T10000C tape drive DIV (Data Integrity and Validation) feature as follows:

- o If div = off, the STK T10000C tape drive does not use the DIV feature. The default is off.
- o If div = on, the STK T10000C tape drive uses the DIV feature.
- o If div = verify, the STK T10000C tape drive uses the DIV feature and the archiver verifies the archive file (tarball) before the file inodes are updated.

exported_media = value

Declares exported media to be available or unavailable to the historian, as follows:

- o If exported_media = available, media exported from a library is considered to be available in the historian. The default is available.
- o If exported_media = unavailable, media exported from a library is considered to be unavailable in the historian. Cartridges with this characteristic are not used by the archiver, stager, or other SAM-QFS tools. They are considered to reside outside of the SAM-QFS environment. This might be used, for example, for cartridges to be transported to offsite storage.

For more information, see the historian(7) man

page.

idle_unload = seconds

Specifies the time, in seconds, that a library-controlled device can be idle before the media in that device is unloaded. Specifying idle_unload = 0 disables this feature. By default, idle_unload = 600, which is 10 minutes.

shared_unload = seconds

```
                    Specifies the time, in seconds, that a shared
                    library-controlled device can be idle before the
                    media in that device is unloaded. A device is
                    shared if it is used by more than one SAM-QFS
                    server. For more information on shared devices see
                    the sony(7), the ibm3494(7), or the stk(7) man
                    page.  Specifying shared_unload = 0 disables this
                    feature.  By default, shared_unload = 60, which is
                    60 seconds.

          inodes    This keyword is still accepted for backward
                    compatibility, but it has no effect.  For more
                    information, see the samfs.cmd(4) man page.

          labels = mode
                    For tape libraries with bar code label readers,
                    this keyword sets the tape label equal to the
                    first or the last characters of the bar code label
                    (uppercased).  For mode, specify either barcodes,
                    barcodes_low, or read, as follows:

                    o If labels = barcodes, the first part of the bar
                      code is used as the label.  Default.

                    o If labels = barcodes_low, the last part of bar
                      code is used as the label.

                    o If labels = read, the label is read from the
                      tape.  If you wish to have the labels different
                      from the barcodes on a library with a bar code
                      label reader, you must set labels = read.

                    When labels is set to barcodes or barcodes_low, a
                    label is written to the tape before the write is
                    enabled for any tape mounted for a write operation
                    that is write enabled, unlabeled and has a
                    readable bar code label.

          log = facility
                    Sets the facility code used for issuing log
                    messages.  For information on the accepted
                    facility types, see the syslog(3) man page.  The

                    default is LOG_LOCAL7.

          oper_privileges = privilege
                    Adds privileges to the operator group.  By
                    default, members of the operator group do not have
                    the privileges to perform the following tasks:
                    media labeling, performing storage element
                    movement actions, submitting full audit requests,
                    changing a device state (except to ON a device),
                    and clearing mount requests.  To grant the
                    privileges needed to perform those actions,
                    specify one or more of the following privilege
                    arguments.

                    privilege   Result
                    all         Grants all privileges in this list.

                    clear       Grants the ability to clear cartridge
                                load requests.

                    fullaudit   Grants the ability to perform a full
                                library audit.

                    label       Allows cartridge labeling.

                    slot        Allows mounting, unloading, and moving
                                cartridges within a library.

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

                    Use a space character between privilege arguments
                    if specifying more than one.

          operator = group
                    Specifies the name of the group that to be granted
                    operational privileges within certain commands
                    (chmed(1M), load(1M), samfsdump(1M), and
                    samfsrestore(1M)) and command queues.  Only one
                    group name can be specified.  Users must have
                    their effective group IDs set to group in order to
                    gain operational privileges.

          optical = media_type
                    Sets the default media type to media_type when a
                    generic optical disk (od) is requested.  A string
                    value is expected.  For information on the
                    accepted media types, see the mcf(4) man page.

                    The default is mo.

          previews = requests
                    Sets the number of outstanding mount requests.
                    Care should be taken when changing this value.
                    Each entry takes about 500 bytes of shared memory.
                    By default, previews = 100.

          samrpc = on | off
                    Invokes the RPC API server process.  If samrpc =
                    on, the RPC API server process, sam-rpcd, is
                    automatically started when Sun QFS or SAM-QFS is
                    started.  By default, samrpc = off, so sam-rpcd is
                    not started automatically.

          remote_keepalive = seconds
                    Specifies the time in seconds the SAMremote server
                    can be idle before a SAMremote client sends a
                    packet to check for the existence of the server.
                    By default, remote_keepalive = 300, which is five
                    minutes.  Specifying remote_keepalive = 0 disables
                    the keepalive function.
```

```
alerts = on | off
          Specifies whether alert notification via Simple
          Network Management Protocol (SNMP) or fault
          history logging via the GUI is supported.  With
          this turned on, you can monitor a Sun QFS or SAM-
          QFS system remotely from a management console such
          as Sun Remote Services (SRS) By default, alerts=on
          is in effect.

avail_timeout = seconds
          Allows the stager to delay before unloading a
          volume being used to stage a file with the stage
          -n attribute set.  This allows a subsequent stage
          request for this file to be processed in
          preference to a file requesting a different
          volume.  Setting avail_timeout = 0 disables this
          function.  By default, avail_timeout = 0.

stale_time = minutes
          Sends an error to any request for removable media
          that has waited for minutes number of minutes.
          Setting stale_time = 0, disables this function.
          By default, stale_time = 30.

tape = media_type
          Sets the default media type to media_type when a
          generic tape (tp) is requested.  A string value is
          expected.  For information on the accepted media
          types, see the mcf(4) man page.  The default is

          lt.

timeout = seconds
          Sets the timeout interval, in seconds, for direct
          access removable media.  If a process fails to
          issue an I/O request to the device within this
          time, the device is removed from job assignment
          and the process receives an ETIME when the next
          I/O to the device commences.  Specifying timeout =
          0 disables this timeout.  The minimum value
          allowed is timeout = 600.  For backwards
          compatibility, values from 1 to 599 are allowed,
          but are overridden by the minimum value.  By
          default, timeout = 600.

tp_mode = mode
          Specifies the mode set for tape drive device nodes
          when not under control of the SAM-QFS software.
          For information, see the chmod(2) man page.  When
          the SAM-QFS software is controlling the drive, the
          mode bits are 0660.

tapealert = eq_number  on | off
          Enables or disables media changer or tape drive
          TapeAlert support by Equipment Number eq_number.
          The eq_number must be either the keyword all (to
          specify all devices) or must match a tape device
          Equipment Number from the mcf file.  By default,
          tapealert = all on.
```

```
   samstorade = on | off
             Enables or disables the StorADE API.  The API
             provides SAM-QFS device attributes and health
             information for StorADE fault analysis.  By
             default, samstorade = on.

   sef = eq_number  [on|off|default] interval
             Enables or disables support for tape drive
             implemented Log Sense delivered via sysevents by
             Equipment Number eq_number.  The eq_number must be
             either the keyword all (to specify all devices) or
             must match an Equipment Number from the mcf file.
             The interval specfies the log sense polling rate.
             A value of 300 is a polling interval once every
             five minutes.  A string value of "once" specifies
             one time just before media unload and is the
             default.  A value of 3600 is a polling interval
             once every hour.  The smallest polling interval is
             five minutes.  By default, sef = all on once.

             Note: The defaults.conf sef entry only controls
             the equipment number and frequency interval for

             sef data. It is the  presence or absence of the
             file /var/opt/SUNWsamfs/sef/sefdata at SAM-QFS
             initialization that determines if sef will run or
             not. When /var/opt/SUNWsamfs/sef/sefdata is
             present, sef will be initialized. You must create
             the sefdata file yourself. To turn off sef, the
             sefdata file must be removed or renamed.

   tapeclean = eq_number  autoclean [on|off] logsense [on|off]
             Enable or disable the robot initiated auto-
             cleaning feature.  Enable or disable additional
             auto-cleaning log sense cleaning indicators from
             the TapeAlert log sense page(2E) flags clean
             now(20), clean periodic(21) and expired cleaning
             media(23) and the Sequential-Access Device log
             sense page(0C) cleaning required flag in
             parameter(256).  Support is by Equipment Number
             eq_number.  The eq_number must be either the
             keyword all (to specify all devices) or must match
             a tape device Equipment Number from the mcf file.
             Note that the logsense on setting has no effect
             unless autoclean is also on.  By default,
             tapeclean = all autoclean off logsense on.  Note:
             When using the auto-cleaning feature with a
             library that has more than two drives, it is
             recommended that you have at least two cleaning
             cartridges per robot. If a cleaning cartridge is
             not available when a drive needs to be cleaned,
             the drive will be put into a down state.

Trace file controls.
   The daemon trace files are controlled by directives in the
   trace file section.  This section begins with the trace
   directive, and ends with the endtrace directive.  The trace
   file control directives are of the form:
```

```
daemon_name.variable_name = value
daemon_name = on
daemon_name = off
```

daemon_name can be one of the following: sam-archiverd, sam-catserverd, sam-fsd, sam-rftd, sam-recycler, sam-sharefsd, sam-stagerd, sam-serverd, sam-clientd, fsmgmt, or all .

Note that fsmgmt is used by fsmgmtd and libfsmgmt.so.

If daemon_name is all, then the variable_name is set to value for all daemons.

For the form: daemon_name = on the trace file controls will be set to the pre-defined values for daemon_name.

In particular, using only the directive
all = on
enables tracing for all daemons. The trace files are written to files named for the daemons (e.g. sam-rftd) in the /var/opt/SUNWsamfs/trace subdirectory.

For the form: daemon_name = off tracing will be turned off for daemon_name.

The variable_name is one of: file, options, age, or size.

daemon_name.file file_name

> set the name of the trace file to file_name. The default is no trace file.
>
> If the daemon_name is all, then file_name is the name of the trace subdirectory that will contain the daemon tracefiles. file_name must be absolute in this case. The default subdirectory is /var/opt/SUNWsamfs/trace.
>
> If file_name is relative (no leading '/'), the file name will be made relative to the trace base directory. If the file does not exist, sam-fsd will create it.

daemon_name.options = option_list

> Set the trace file options to option_list. option_list is a space separated list of trace options. A trace option is an event to trace, or an element to include in the trace line. To exclude an option, prefix the option with a '-'.
>
> For selecting events, option may be one or more of:

none	Clear all event types.
all	Set event types for tracing the most interesting events. These are: cust err fatal ipc misc proc rft.
alloc	Memory allocations.

```
          cust    Customer notification syslog or notify file
                  messages.

          err     Non-fatal program errors.

          fatal   Fatal syslog messages.

          files   File actions.

          rft     File transfer events.

          ipc     Inter process communication.

          misc    Miscellaneous.

          oprmsg  Operator messages.

          proc    Process initiation and completion.

          queue   Archiver queue contents when changed.

          For selecting message elements, option may be one or
          more of:

          date    Include the date in message (the time is always
                  included).

          module  Include source file name and line number in
                  message.

          type    Include event type in message.

          The pre-defined events are:  cust, err, fatal, misc,
          proc, rft.  The message elements program[pid] and time
          are always included and can't be deselected.

     daemon_name.age = age
          Set the time between trace file rotations to age.  age
          may be specified with the seconds, minutes, hours,
          days, weeks and years.  Note: Do not set this value to
          two minutes or less. If you do, the rotation will never
          take place.  sam-fsd can perform trace file "rotations"
          using the script /opt/SUNWsamfs/sbin/trace_rotate.
          Trace file rotations are useful to control the size of
          trace files.

     daemon_name.size = size
          Set the trace file size at which trace file rotations
          will be performed.  size may be specified with the
          suffices 'b', 'k', 'M', 'G', and 'T', for bytes,
          kilobytes, megabytes, gigabytes, and terabytes.

EXAMPLES
     Here is a sample defaults.conf configuration file.

     optical = mo
     debug = logging debug timing
     tape = lt
     log = LOG_LOCAL7
     timeout = 30
     idle_unload = 600
     tp_mode = 0666

     rc_delay = 10
     cy_delay = 10
     ml_delay = 10
     hp_delay = 10
     ds_delay = 10
     lt_unload = 7
     st_unload = 15
     lt_blksize = 16
     operator = sam
     oper_privileges = label slot
     trace
     all = on       # Turn on tracing for all daemons
     sam-archiverd.size = 10M # Rotate archiver trace file after 10 megabytes
     sam-rftd.file = /tmp/sam-rftd.trace  # change file name for sam-rft daemon
     sam-recycler = off  # Turn off tracing for sam-recycler daemon
     endtrace
```

FILES
    /opt/SUNWsamfs/examples/defaults.conf
        Contains an example of a defaults.conf file.

SEE ALSO
    request(1).

    samset(1M), sam-fsd(1M), tplabel(1M), tapealert(1M).

    chmod(2).

    syslog(3).

    mcf(4), samfs.cmd(4), trace_rotate(4), sefsysevent(4).

    historian(7).

devlog(4)

NAME
    devlog - Device log file

SYNOPSIS
    /var/opt/SUNWsamfs/devlog/nn

AVAILABILITY
    SUNWsamfs

DESCRIPTION
    In SAM-QFS environments, media or tape hardware events that require operator intervention (such as tape positioning errors and requests for cleaning) are logged to file in the following directory:

/var/opt/SUNWsamfs

Within the preceding directory, events are logged to files that are named for the devices listed in the mcf file. For example, file devlog/47 logs all events for the device identified by Equipment Number 47 in the mcf file.

After an event is logged, you can use the tapealert(1M) command to read the event logged in the devlog/nn file, interpret the event, and write it to a text file for easier viewing. For more information about the specific events logged to the device log files, see the tapealert(1M) man page.

The tapealert(1M) command logs the following two types of messages in the device log (devlog/nn) file:

o Device TapeAlert support

o Active TapeAlert flags

The preceding type of messages are the undecoded TapeAlert events. The tapealert(1M) command decodes these messages into a more readable format. The undecoded device log messages for device support contains the following information:

Field	Content
1	The date in year/month/day format.
2	The time expressed in a 24-hour clock.
3	The message number, followed by TapeAlert and supported. TapeAlert messages start at 12000.

The following is an example of a device support message:

```
2003/06/13 10:52:23 12001 TapeAlert supported
```

The device log messages for active TapeAlert flags contain the following information:

Field	Content
1	The date in year/month/day format.
2	The time expressed in a 24-hour clock.
3	The message number, followed by TapeAlert. TapeAlert messages start at 12000.
4	The characters eq= followed by the mcf(4) equipment number.
5	The characters type= followed by the inquiry peripheral device type.
6	The characters seq= followed by the sysevent

	sequence number. The sysevent sequence number is zero if the sysevent_post_event function fails or is not called. The sysevent event handler $sequence macro is the same as the devlog/nn file's seq=n number.
7	The characters len= followed by the number of valid TapeAlert flags.
8	The flags field. The 64 TapeAlert flags are written in big endian format. The most significant bit, on the left, is flag 64. The least significant bit is flag 1.

The following is an example of a TapeAlert flags message:

```
2003/06/13 10:52:23 12006 TapeAlert eq=91 type=1 seq=8 len=50 flags=0x0002004000000000
```

A decoded TapeAlert flag consists of four parts:

1. Flag

2. Severity

3. Application message

4. Probable cause

The T10 Technical Committee defines three types of flags. Table 1 lists these flags in order of increasing severity.

Table 1. Flag Types

Severity	Urgent Intervention	Risk of Data Loss	Explanation
Critical	X	X	
Warning		X	X
Information			X

If an Information-level flag is issued, you can perceive it as a predicted failure. Take the time to correct the problem before it worsens.

The tapealert(1M) command supports the minimum flag subset as defined by the T10 Committee. Table 2 shows these flags.

Table 2. Tape Drive TapeAlert Flags - Minimum Subset

Flag Number, Type	Explanation
3h, Hard error	Active for any unrecoverable read/write/positioning error. Internally deactivated when the media is unloaded. This flag is active as specified in flag number 5h and 6h.
4h, Media	Active for any unrecoverable read/write/positioning error that is due to faulty media. Internally deactivated when the media is unloaded.

5h, Read failure	Active for any unrecoverable read error where the diagnosis is uncertain and could either be faulty media or faulty drive hardware. Internally deactivated when the media is unloaded.
6h, Write failure	Active for any unrecoverable write/positioning error where the diagnosis is uncertain and could either be faulty media or faulty drive hardware. Internally deactivated when the media is unloaded.
14h, Clean now	Active when the tape drive detects a cleaning cycle is needed. Internally deactivated when the tape drive is successfully cleaned.
16h, Expired cleaning	Active when the tape drive detects a cleaning cycle was attempted but was not successful. Internally deactivated when the next cleaning cycle is attempted.
1fh, Hardware B	Active when the tape drive fails its internal Power-On-Self-Tests (POST). Not internally deactivated until the drive is powered off.

Table 3 summarizes the errors in the devlog/nn file.

Table 3. TapeAlert Flag Definition Groupings for Tape Drives With or Without an Autoloader

Flag Number(s)	Definition
01h to 13h	Tape drive write/read management
14h to 19h	Cleaning management
1Ah to 27h	Tape drive hardware errors
28h to 31h	Tape autoloader errors
32h to 40h	Further tape errors

The information in tables 1, 2, and 3 is derived from SCSI Stream Commands - 2 (SSC-2), Revision 08d.

SEE ALSO

tapealert(1M).

mcf(4).

diskvols.conf(4)

```
NAME
     diskvols.conf - Defines disk archive volumes for SAM-QFS
     environments

SYNOPSIS
     /etc/opt/SUNWsamfs/diskvols.conf

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     A SAM-QFS file can have one or more of its archive copies
     written to a disk archive resource.  A disk volume that
     represents the resource is stored in the inode of the
     archived file.

     The disk volume configuration file, diskvols.conf, defines
     the mapping between a disk volume and the corresponding
     resource.  The sam-fsd daemon reads the diskvols.conf file
     when the sam-fsd daemon is started.  The diskvols.conf file
     can be changed at any time while the sam-fsd daemon is
     running.  The changes take effect when the sam-fsd daemon is
     restarted or sent the signal SIGHUP.

     The mappings are specified one per line.  Each line consists
     of two fields separated by white space.  Leading white space
     is ignored.  Everything after a pound character (#) is
     ignored.  Lines can be continued by using a backslash
     character (\) as the last character on the line.  The syntax
     for this line is as follows:

     disk_volume resource

     where:

     disk_volume
               An alphanumeric string.  The string can contain up
               to 31 characters.

     resource  A resource specification in one of the following
               formats:

               pathname       This format contains the path name
                              of the disk archive directory on
                              the local host.

               [host:]pathname
                              This format specifies the host as
                              the name of the disk archive server
                              and pathname as the path name of
                              the disk archive directory on that
                              host.

               stk5800 host[:port]
                              This format defines a disk volume
                              as residing on a Sun StorageTek
```

```
                         5800 Storage System.  The host
                         field contains the name or IP
                         address and port as the port number
                         of the Sun StorageTek 5800 Storage
                         System.  By default, the port
                         number is 8080.
```

NOTE: Extreme care must be taken when configuring disk archiving in an environment with multiple SAM-QFS servers. The diskvols.conf file for each SAM-QFS server must point to a unique set of disk volume resource specifications (disk archiving target directories). If any of these are shared between different SAM-QFS servers, then running the recycler from one SAM-QFS server will destroy the disk archive data that is being managed by the other SAM-QFS server.

CLIENT DEFINITIONS SECTION

The clients and endclients directives delimit this section of the diskvols.conf file.

The client definitions section defines the trusted client systems. After the disk archiving server accepts a client connection, it verifies that the socket address belongs to a host in the trusted client definitions section. If not, the connection is refused.

The file transfer parameters which set the TCP window size and block size are defined in the rft.cmd configuration file on the server. These can be tuned for best performance.

EXAMPLES

This example shows two diskvols.conf files.

File 1 is a diskvols.conf file on client system earth that defines the following:

o There is one volume serial name (VSN) for a local disk archive.

o There are two remote VSNs. Remote VSN remote1 resides in /quidditch on the remote server gryffindor, and remote VSN remote2 resides in /quidditch on remote server ravenclaw.

o There is one volume serial name (VSN) for a Sun StorageTek 5800 Storage System.

```
#
# This is file /etc/opt/SUNWsamfs/diskvols.conf on local system earth

#
local_archive       /DiskArchive
remote1  gryffindor:/quidditch
remote2   ravenclaw:/quidditch
stk_archive  stk5800 mars
```

File 2 is the diskvols.conf file that resides on the server system gryffindor and ravenclaw. Only the diskvols.conf file for server gryffindor is shown.

```
#
# This is file /etc/opt/SUNWsamfs/diskvols.conf on server system gryffindor
#
clients
earth
endclients
```

SEE ALSO
     archiver(1M), sam-fsd(1M).

     archiver.cmd(4), rft.cmd(4).

WARNINGS
     If more than one SAM-QFS environment is sharing a Sun StorageTek 5800 Storage System you must take extra care when configuring the diskvols.conf file. If you are running multiple connections to a Sun StorageTek 5800 Storage System disk archive, then the disk_volume name needs to be unique across all SAM-QFS environments. For example, stk_archive_earth on one server, stk_archive_pluto on the next server, etc.

```
#
# This is file /etc/opt/SUNWsamfs/diskvols.conf on server earth
#
stk_archive_earth  stk5800 mars

#
# This is file /etc/opt/SUNWsamfs/diskvols.conf on server pluto
#
stk_archive_pluto  stk5800 mars
```

     It is important to follow this recommendation, because there is no enforcement of this restriction in the SAM-QFS software.

fsalogd.cmd(4)

NAME
     fsalogd.cmd - SAM-QFS fsalogd command file

SYNOPSIS
     /etc/opt/SUNWsamfs/fsalogd.cmd

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     Directives for controlling the file system activity log daemon (sam-fsalogd) can be read from the /etc/opt/SUNWsamfs/fsalogd.cmd file. The directives must appear one per line.

     Comment lines are permitted. Comment lines must begin with a pound character (#), and the comment can extend through the rest of the line.

Directives that appear prior to any fs= directive are applied to all file systems. Directives that appear after a fs= directive are applied to the specified file system only. Directives that are specific to a file system override general directives.

The following directives control the operation of the fsalogd daemon.

DIRECTIVES

The following miscellaneous directives can be specified in the fsalogd.cmd file:

log_path = n
: Specifies the path of the log file directory. The default is /var/opt/SUNWsamfs/fsalogd/family_set_name. Log files are created with the name familyset.YYYYMMDDhhmm.log.

fs = file_system_family_set_name
: Specifies that the subsequent directives apply to the indicated file_system_family_set_name only.

log_rollover_interval = n
: Sets the log file rollover interval time to n seconds. When the interval time since the creation of the log file has elapsed a new log file will be created. The default is 28800 seconds (8 hours).

log_expire = n
: Sets the log file expiration time to n seconds. When the time since the creation of the log file has exceeded log_expire, the log file is eligible for

  deletion. The default is 172800 seconds (48 hours). Expired log files are checked every log rollover interval.

event_interval = n
: Sets the event interval time used by SAM-QFS to call out to sam-fsalogd with accumulated events. The default time is 10 seconds.

event_buffer_size = n
: Sets the buffer size in events. The default buffer size is 256K which is about 8,000 events.

event_open_retry = n
: Sets the number of allowable retries when sam-fsalogd is establishing its connection with the file system. The default retry count is 5.

Example 1. This
: example file sets the event_interval and log_path directive for the samfs1 file system.

```
          fs = samfs1
```

```
                    event_interval = 10
                    log_path = /var/opt/SUNWsamfs/fsalogd/samfs1

          Example 2.  This example specifies the log_path for
          each file system.

                    event_interval = 10

                    fs = samfs1
                    log_path = /var/adm/fsalog/samfs1

                    fs = samfs2
                    log_path = /var/adm/fsalog/samfs2
```

SEE ALSO
     sam-fsalogd(1),

ftp.cmd(4)

NAME
     ftp.cmd - Renamed to "rft.cmd"

SEE ALSO
     rft.cmd(1M).

hosts.fs(4)

NAME
     hosts.fs - Host information for Sun QFS shared file systems

SYNOPSIS
     /etc/opt/SUNWsamfs/hosts.fs

AVAILABILITY
     SUNWqfs

     SUNWsamfs

DESCRIPTION
     The /etc/opt/SUNWsamfs/hosts.fs file specifies the hosts and network interfaces used by a Sun QFS shared file system. The fs suffix must be the family set name of the Sun QFS shared file system as specified in the mcf(4) file.

     The file /etc/opt/SUNWsamfs/hosts.fs is required by sammkfs(1M) at the time a Sun QFS shared file system is created. The sammkfs(1M) command reads /etc/opt/SUNWsamfs/hosts.fs and integrates the information into the file system when initializing the file system. The file system’s shared hosts information can be subsequently modified using the samsharefs(1M) command.

     Another file, hosts.fs.local(4), can also reside on each host system included in the shared file system. Daemons local to each host system use the shared hosts file and the local hosts file, if any, to initialize network connections for the shared file system.

Each file system's shared hosts file determines the host configuration for that file system. This includes the following:

o The identity of the file system's metadata server.

o The host systems (and host IP interfaces) that are allowed to connect to the Sun QFS shared file system's metadata server.

o The identities of the potential metadata server hosts. These are systems that can act as the file system's metadata server if the preferred metadata server is unavailable.

The hosts.fs file is comprised of lines containing five fields of information. Each line corresponds to one host that is permitted to access the file system. The fields are as follows:

Field Number	Content
1	The name of the host. The host name field contains the name of a host that is to be permitted access to the shared file system. The value of this field must match the output of the hostname(1) command on that host.
2	The host IP addresses. The host IP address field contains a list of one or more host IP interface addresses or names that the metadata server must be able to resolve to IP addresses. If there are multiple IP interfaces that a host can use to connect to a server, they must be separated by commas. You should avoid using a domain name in this field, because during the reboot process, when sam-fsd is trying to contact the metadata server, naming services are likely not up. This means that the name may not be resolvable if it is not in the /etc/inet/ipnodes or /etc/inet/hosts file; this will cause the mount to fail and could cause the reboot to hang.
3	The server priority of the host. The server priority field is a numeric field. If the field is zero, the host cannot act as the metadata server for the file system. If the field is nonzero, the host can act as the metadata server for the file system.
4	A number that indicates the stager priority.

```
                   This numeric field is not used by the shared
                   file system software.  It is recommended that
                   this field be set to zero.

     5             A server field.  This optional field must be
                   set for one of the hosts in the hosts.fs
                   file.  That host must have a nonzero server
                   priority field.  If present, this field must
                   contain the string server.
```

In this file, a pound character (#) indicates a comment. Comments continue from the pound character to the end of the line. All characters to the right of the pound character are ignored.

After the file system is initialized using the sammkfs(1M) command, only the metadata server host is permitted to run the samfsck(1M) to repair the file system. The server on which sammkfs(1M) is run is typically declared to be the metadata server.

When a client is attempting to connect to the metadata server, the client obtains the list of names and addresses from the second field, which is the host IP address field, of the server's row in the hosts.fs file. It attempts to connect to these names, in the order in which they appear, until it connects successfully. If the client has a local hosts.fs.local(4) file, only the names or addresses that are present in both files are used. The hosts.fs.local(4) file determines the order in which host connections are attempted.

When a metadata server receives a connect attempt, it performs address lookups on the values from the second column of the hosts.fs file until it finds one that matches the IP address of the incoming connection. If it fails to find one, it refuses the connection.

For file systems that are mounted at boot time, you should add the file system's hosts to the /etc/inet/hosts or /etc/inet/ipnodes files. On clients, the names of the servers should be added; on servers, all of the file system's hosts should be added.

EXAMPLES

Example 1. The following is a sample hosts.fs configuration file called /etc/opt/SUNWsamfs/hosts.shsam1.

```
#
# shsam1 config, titan/tethys servers, mimas/dione clients
#
# This file goes in titan:/etc/opt/SUNWsamfs/hosts.shsam1,
# and is used by 'sammkfs -S shsam1' to initialize the FS
# meta data.  Subsequent changes to the configuration are
# made using samsharefs(1M).
#
#
titan   titan       1 0 server
tethys  tethys      2 0
```

```
mimas   mimas       0 0
dione   dione       0 0
```

Example 2. This hosts configuration file is more complicated that the one in example 1. It supports a configuration where two potential servers also have a private interconnect between them.

```
#
# shsam1 config, titan/tethys servers, mimas/dione clients
#
# This file goes in titan:/etc/opt/SUNWsamfs/hosts.shsam1, and
# is used by mkfs -S to initialize the FS meta data.  Subsequent
# changes to the configuration are made using samsharefs(1M).

#
#
titan   titan-ge,titan.xyzco.com 1 0 server
tethys  tethys-ge,tethys.xyzco.com 2 0
mimas   mimas.xyzco.com 0 0
dione   dione.xyzco.com 0 0
```

To ensure that titan and tethys always connect to each other through their private interfaces, titan-ge and tethys-ge, each must have a hosts.shsam1.local file (see hosts.fs.local(4)). To avoid the inefficiencies of attempting to connect to the unreachable titan-ge and tethys-ge interfaces, mimas and dione should also have their own hosts.shsam1.local files.

FILES

/opt/SUNWsamfs/examples/hosts.shsam1
: Contains an example of a hosts.fs file.

/opt/SUNWsamfs/examples/hosts.shsam1.local.server

/opt/SUNWsamfs/examples/hosts.shsam1.local.client
: Contain examples of hosts.fs.local(4) files.

SEE ALSO

hostname(1).

samfsck(1M), samfsconfig(1M), sammkfs(1M), samsharefs(1M), sam-sharefsd(1M).

hosts.fs.local(4), mcf(4).

hosts.fs.local(4)

NAME

hosts.fs.local - Local host information for Sun QFS shared file systems

SYNOPSIS

/etc/opt/SUNWsamfs/hosts.fs.local

AVAILABILITY

SUNWqfs

SUNWsamfs

DESCRIPTION

A /etc/opt/SUNWsamfs/hosts.fs.local file can reside on each host system included in the Sun QFS shared file system. This file is used in conjuntion with the shared hosts file, which resides in the shared file system and is initialized by sammkfs(1M) from hosts.fs(4), to initialize network connections between the hosts of a shared file system. For more information, see the hosts.fs(4) and samsharefs(1M) man pages.

The Sun QFS shared file system daemon uses the /etc/opt/SUNWsamfs/hosts.fs.local file and the shared hosts file present in the file system during initialization and reconfiguration to determine the server interfaces to which it should attempt to connect. Its function is to restrict the server interfaces to which each client connects. The fs portion of the name must be the family set name of the Sun QFS shared file system as specified in the mcf file. For more information on the mcf file, see the mcf(4) man page.

Each line in the hosts.fs.local file corresponds to a possible metadata server. Each line contains the following fields:

Field Number	Content
1	The name of the host. This field contains the name of a potential metadata server host to which the local host can connect. This field must match the first field of the host in the shared hosts file. You can use the samsharefs(1M) command to verify the content of the fields of the shared hosts file.
2	A comma-separated list of host IP names or addresses. This should be a subset of the second field from the same hosts entry in the shared hosts file.

The hosts.fs.local file is typically generated by copying the shared file system's shared hosts file to /etc/opt/SUNWsamfs/hosts.fs.local on each host. Each line referring to a non-server host is then deleted, and the third through fifth fields in the remaining lines are deleted. The network topology of the hosts is then examined in conjunction with the file, and the server interfaces that the local host should not attempt to connect to are removed from the second field. When all of these have been removed, the file is written out. The samd(1M) command is then used to cause any configuration changes to take effect.

During startup and file system reconfiguration, the sam-sharefsd(1M) daemon attempts to connect to the server host.

To do this, it searches the shared hosts file for the server's identity, and it extracts the list of IP names and addresses from the server's shared hosts file entry. The daemon then looks up the server's name in the file system's local hosts file, if any. If a local hosts file does not exist, the daemon uses the list from the shared hosts file. If the local hosts file does exist, then the corresponding list of host addresses is found in the local hosts file, the two lists of host addresses are searched (lexically) for common entries, and a common list is generated. The ordering of the list is determined by the local hosts file (left-most first). The names or addresses in the common list are looked up and used to attempt to connect to the server. If an attempt fails, the daemon attempts using any remaining addresses in order until all the addresses have been tried.

EXAMPLES

The following shared hosts configuration file supports a configuration in which two potential servers share a private interconnection and communicate to the other hosts sharing the file system using a separate network. The examples in this section show the hosts.shsam1.local files that can be found on the various hosts.

```
#
# shsam1 config, titan/tethys servers, mimas/dione clients
#
# This file goes in titan:/etc/opt/SUNWsamfs/hosts.shsam1, and
# is used by 'mkfs -S shsam1' to initialize the FS meta data.
# Subsequent changes to the configuration are made using
# samsharefs(1M).
#
titan   titan-ge,titan.xyzco.com 1 0 server
tethys  tethys-ge,tethys.xyzco.com 2 0
mimas   mimas.xyzco.com 0 0
dione   dione.xyzco.com 0 0
```

To ensure that titan and tethys always connect to each other through their private interfaces, titan-ge and tethys-ge, each requires a hosts.fs.local(4) file. To achieve this, files titan:/etc/opt/SUNWsamfs/hosts.shsam1.local and tethys:/etc/opt/SUNWsamfs/hosts.shsam1.local would contain the following lines:

```
#
# shsam1 server local config, titan/tethys servers, mimas/dione clients
#
titan   titan-ge
tethys  tethys-ge
```

To avoid the delays and inefficiencies of having mimas and dione attempt to connect to titan and tethys through the inaccessible, private titan-ge and tethys-ge interfaces, mimas and dione should also have their own hosts.fs.local(4) files. Files mimas:/etc/opt/SUNWsamfs/hosts.shsam1.local and dione:/etc/opt/SUNWsamfs/hosts.shsam1.local contain the following lines:

```
     #
     # shsam1 client local config, titan/tethys servers, mimas/dione clients
     #
     titan   titan.xyzco.com
     tethys  tethys.xyzco.com
```

FILES

```
     /opt/SUNWsamfs/examples/hosts.shsam1
                         Contains an example of a hosts.fs file.

     /opt/SUNWsamfs/examples/hosts.shsam1.local.server

     /opt/SUNWsamfs/examples/hosts.shsam1.local.client
                         Contain examples of hosts.fs.local
                         files.
```

SEE ALSO

samfsck(1M), samfsconfig(1M), sammkfs(1M), samsharefs(1M), sam-sharefsd(1M).

hosts.fs(4), mcf(4).

inquiry.conf(4)

NAME

inquiry.conf - SCSI inquiry strings for SAM-QFS device types

SYNOPSIS

/etc/opt/SUNWsamfs/inquiry.conf

AVAILABILITY

SUNWsamfs

DESCRIPTION

The inquiry configuration file, inquiry.conf, maps a SCSI device to a SAM-QFS device type. The inquiry.conf file contains the vendor identification and product identification reported by a SCSI device in response to an inquiry command.

Entries in the file are made up of three quoted fields separated by a comma and/or white space and optionally followed by a comment. These entries have the following format:

```
"vendor_id", "product_id", "SAM-QFS_name"  #comment
```

The vendor_id and product_id are the vendor identification (8 characters) and product identification (16 characters) as reported in the inquiry data. The SAM-QFS_name is the SAM-QFS device name as described subsequently on this man page.

Trailing spaces do not need to be supplied in the vendor_id or the product_id fields. Any occurrence of a quotation mark ("), a comma (,), or a back slash (\) in any _id field should be prefaced with the escape character, which is a back slash (\). Blank lines and lines beginning with a pound character (#) are ignored.

The following device names are supported within the SAM-QFS environment:

Device Name	Device Type
acl2640	ACL 2640 tape library
acl452	ACL 4/52 tape library
adic448	ADIC 448 tape library
adic100	ADIC Scalar 100 tape library
adic1000	ADIC Scalar 1000 and Scalar 10K tape library
archdat	Archive Python 4mm DAT drive
atl1500	Sun StorEdge L25 and L100 and ATL M1500 and M2500 libraries.
atlp3000	ATL P3000, P4000 and P7000 tape library
cyg1803	Cygnet Jukebox 1803 library
dlt2000	Digital Linear Tape (2000, 4000, 7000, 8000 and SuperDLT series) drive
dlt2700	Digital Linear Tape Media Changer/Stacker (2000, 4000, 7000, 8000 and SuperDLT series)
docstor	DISC automated library
exb210	Exabyte 210 tape library
exbx80	Exabyte X80 tape library
exb8505	Exabyte 8505 8mm cartidge tape drive
exbm2	Exabyte Mammoth-2 8mm cartidge tape drive
fujitsu_128	Fujitsu M8100 128 track tape drive
grauaci	GRAU media library
hpc7200	HP L9/L20/L60 series libraries
hpslxx	HP SL48 tape 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
ibm3580	IBM 3580, Seagate Viper 200 and HP Ultrium (LTO) tape drives
ibm3584	IBM 3584 media changer
ibm3590	IBM 3590 tape drive
lms4100	Laser Magnetic Laserdrive 4100
lms4500	Laser Magnetic Laserdrive 4500
metd28	Metrum D-28 tape library
metd360	Metrum D-360 tape library
qual82xx	Qualstar 42xx, 62xx and 82xx series tape library
quantumc4	Quantum PX500 and Sun StorEdge C4 tape library
rap4500	Laser Magnetic RapidChanger 4500
rsp2150	Metrum RSP-2150 VHS video tape drive
sonyait	Sony AIT tape drive
sonysait	Sony Super AIT tape drive
sonydms	Sony Digital Mass Storage tape library
sonycsm	Sony CSM-20S tape library
sonydtf	Sony DTF tape drive
speclog	Spectra Logic Libraries
stk4280	StorageTek 4280 Tape drive
stk9490	StorageTek 9490 Tape drive
stk9840	StorageTek 9840 Tape drive
stktitan	StorageTek Titanium Tape drive
stkapi	StorageTek API library
stkd3	StorageTek D3 Tape drive
stk97xx	StorageTek 97xx Media Libraries
stklxx	StorageTek L20, L40, L80 and L500 Tape Libraries and Sun StorEdge L7 and L8 autoloaders.

```
     odi_neo        Overland Data Inc. Neo Series Tape Libraries

EXAMPLES
     The following is an example configuration file:

     "HP",       "Ultrium 1", "ibm3580"   # HP Ultrium Tape
     "HP",       "Ultrium 2", "ibm3580"   # HP Ultrium Tape
     "HP",       "Ultrium 3", "ibm3580"   # HP Ultrium Tape
     "Plasmon",  "G-Enterprise","plasmong"   # Plasmon G Enterprise
     "Plasmon",  "UDO",      "plasmonUDO"    # Plasmon UDO 30GB optical drive
     "STK",      "L700",     "stk97xx"   # STK L700 series SCSI
     "STK",      "L180",     "stk97xx"   # STK L180 series SCSI
     "STK",      "SL500",    "stklxx"    # STK SL500
     "STK",      "T10000A",  "stktitan"  # STK titanium drive T10000A

     The existence of a device in the previous example file does
     not imply that the device is supported by SAM-QFS.

SEE ALSO
     mcf(4).

NOTES
     Whenever a new version of SAM-QFS is installed, the existing
     inquiry.conf file is copied to inquiry.conf.MMDDYY for
     reference and back-up purposes.

     During device identification, the vendor_id and product_id
     values are only compared through the length of the string
     supplied in the inquiry.conf file.  To insure an exact
     match, the entries should be ordered with longer names
     first.

WARNINGS
     This interface is supplied to circumvent problems that occur
     when hardware vendors change the vendor_id and product_id
     values returned.  For example, some hardware vendors return
     a different value for a product_id if the hardware is
     supplied by an OEM.

     Oracle Corporation does not support mapping untested
     hardware to a SAM-QFS name.
```

mcf(4)

```
NAME
     mcf - Master configuration file for Sun QFS and SAM-QFS
     software

SYNOPSIS
     /etc/opt/SUNWsamfs/mcf

AVAILABILITY
     SUNWsamfs

     SUNWqfs
```

DESCRIPTION

The mcf file defines the devices and family sets used by Sun QFS and SAM-QFS software. The mcf file is read when sam-fsd is started. You can change it at any time while sam-fsd is running. The changes take effect when sam-fsd is restarted, or sent the signal SIGHUP.

The following examples show an mcf file for a SAM-QFS archiving environment and an mcf file for a Sun QFS file system.

Example 1. The following is an example of a SAM-QFS mcf file:

```
#
# SAM-QFS archiving file system configuration example
#
# Equipment       Eq Eq Family Dev Additional
# Identifier      Nm Tp Set    St  Parameters
# --------------- -- -- ------ --- ----------
samfs1            10 ms samfs1
/dev/dsk/c1t0d0s6 11 md samfs1 -
/dev/dsk/c2t0d0s6 12 md samfs1 -
#
samfs2                            20 ms samfs2 - shared
/dev/dsk/c2t50020F2300000C98d0s5 21 md samfs2 -
/dev/dsk/c2t50020F2300004921d0s5 22 md samfs2 -
#
/dev/samst/c3t500104F0008E6C2Cu0 30 rb SL500 on SL500
/dev/rmt/2bn                      31 tp SL500 on
/dev/rmt/3bn                      32 tp SL500 on
#
/dev/rmt/0cbn     40 tp -      on
#
/dev/samst/c1t3u1 50 rb ml50   on   /usr/tmp/ml50_cat
/dev/rmt/2cbn     51 tp ml50   on
#
```

Example 2. The following is an example of a Sun QFS mcf file:

```
#
# Sun QFS file system configuration example
#
# Equipment        Eq Eq Family Dev Additional
# Identifier       Nm Tp Set    St  Parameters
# --------------- -- -- ------ --- ----------
#
qfs1               10 ma qfs1
/dev/dsk/c1t1d0s3  11 mm qfs1   -
/dev/dsk/c2t1d0s3  12 mm qfs1   -
/dev/dsk/c3t1d0s3  13 md qfs1   -
/dev/dsk/c4t1d0s3  14 md qfs1   -
#
qfs2                              20 ma qfs2
/dev/dsk/c1t50020F2300000C98d0s0 21 mm qfs2  -
/dev/dsk/c1t50020F2300004921d0s0 22 mm qfs2 -
/dev/dsk/c2t50020F2300004655d0s1 23 g0 qfs2 -
/dev/dsk/c3t50020F230000651Cd0s1 24 g0 qfs2 -
```

```
/dev/dsk/c2t50020F2300004655d0s2 25 g1 qfs2 -
/dev/dsk/c3t50020F230000651Cd0s2 26 g1 qfs2 -
#
qfs3                             30 ma qfs3 - shared
/dev/dsk/c2t50020F2300000C98d0s2 31 mm qfs3 -
/dev/dsk/c2t50020F2300004921d0s2 32 mm qfs3 -
/dev/dsk/c2t50020F2300000C98d0s3 33 mr qfs3 -
/dev/dsk/c2t50020F2300004921d0s3 34 mr qfs3 -
#
```

Example 3. The following is an example of a Sun QFS mcf file within a SAN environment:

```
#
# Sun QFS file system configuration example
#
# Equipment        Eq Eq Family Dev Additional
# Identifier       Nm Tp Set    St  Parameters
# ---------------- -- -- ------ --- ----------
#
qfs1               10 ms  qfs1   -
/dev/dsk/c5t16d0s0 11 md  qfs1   -
/dev/dsk/c5t17d0s0 12 md  qfs1   -
/dev/dsk/c5t18d0s0 13 md  qfs1   -
/dev/dsk/c5t19d0s0 14 md  qfs1   -
/dev/dsk/c5t20d0s0 15 md  qfs1   -
/dev/dsk/c5t21d0s0 16 md  qfs1   -
#
```

As the preceding examples show, each line in the mcf file is divided into six fields. The format of the fields in the mcf file is as follows:

```
Equipment  Equipment  Equipment  Family  Device  Additional
Identifier Number     Type       Set     State   Parameters
```

The Equipment Identifier, Equipment Number, and Equipment Type fields are required for each entry. The mcf file can contain comments. Each comment line must begin with a pound character (#). Blank lines are ignored. The fields in the file must be separated by white space. A dash character (-) can be used to indicate a field with no entry.

This man page describes the content of a Sun QFS or SAM-QFS mcf file. For more configuration information, see the Sun QFS File System Configuration and Administration Guide. After your Sun QFS or SAM-QFS software is installed, you can see more examples of mcf files in the following directory:

```
/opt/SUNWsamfs/examples
```

mcf File Fields

This section defines the fields in the mcf file. Note that Sun QFS (non-archiving) environments do not include removable media devices in their mcf files.

When writing the mcf file, group together the lines that define similar devices. For example, create this file such that the devices for a file system appear on consecutive

```
lines and devices for a library appear in a separate set of
consecutive lines.

o The Equipment Identifier specifies a file system and its
  disk devices or it specifies the devices associated with
  an automated library.

    -  For file system definition lines, this field can
       contain two types of entries.  The first line in a
       file system definition must contain the file system
       name in the Equipment Identifier field, and it must
       be no longer than 31 characters in length.  The file
       system name specified must be identical to the
       content of the Family Set field.  For example:

       Equipment    Equipment  Equipment  Family  Device  Addl
       Identifier   Number     Type       Set     State   Params
       ----------   -------    ---------  ------  ------  ------
       samqfs1      1          ms         samqfs1 -       -

       Subsequent lines in the mcf file define disk devices
       to be included in the file system.  The Equipment
       Identifier fields in these lines can be no longer
       than 127 characters in length.

    -  For automated library definition lines, the Equipment

       Identifier field contains drive identifier
       information and can be no longer than 127 characters
       in length.  For example:

       Equipment     Equipment  Equipment  Family  Device  Addl
       Identifier    Number     Type       Set     State   Params
       ----------    -------    ---------  ------  ------  ------
       /dev/rmt/0cbn 61         tp         9730    on      -

o The Equipment number field contains a unique number for
  each disk or removable media device configured.  The
  number you specify must be in the following range:

  1 < Equipment_number < 65534

  Oracle Corporation recommends that you use low numbers in
  order to keep the internal software tables small.

o The Equipment Type field contains a 2-character code that
  specifies the device being defined as either a disk in a
  file system or as a removable media device.  This man page
  includes information on appropriate codes.

o The Family Set name is an arbitrary name that you select
  when the mcf is created.  This field can be no longer than
  31 characters in length.  The Family Set name defines and
  associates related groups of devices.  This can be either
  a file system name, an automated library identifier, or a
  dash character (-), as follows:

    -  If it is a file system name, all disk devices in the
       file system must use the same file system name in
```

```
              this field.

           -  If it is an automated library identifier, the library
              and all its associated drive devices must use the
              same identifier.

           -  If it is a standalone removable media device, use a
              dash (-) character in this field.

       o The Device State field defines the default status for the
         device at the time the system reads the mcf file.  Valid
         values are as follows:  on (default), off, unavail, or
         down.  This field is used for disk devices, libraries,
         drives, and other devices.

       o The Additional Parameters field provides additional
         information.  It can contain the path to a library catalog
         file, an interface file, or other configuration
         information.  The Additional Parameters field can be no

         longer than 127 characters.  For example, this field can
         be used to specify a nondefault location for the library
         catalog file. If mcf file is being configured on a
         SunCluster node running HA-SAM, this field must specify
         the library catalog file in default location. In HA-SAM
         configuration /var/opt/SUNWsamfs/catalog is linked to
         cluster filesystem which is shared among all nodes within
         the SunCluster.

  File System Disks
       When defining a disk cache family set, the following entries
       define a Sun QFS or SAM-QFS file system:

       ms   A Sun QFS or SAM-QFS disk cache family set with no meta
            devices.  Metadata resides on the data device(s).

       ma   A Sun QFS or SAM-QFS disk cache family set with one or
            more meta devices. Metadata resides on these meta
            devices.  File data resides on the data device(s).

       A maximum of 252 separate magnetic disk devices can be
       defined for each ms or ma disk cache family set.

       The Family Set field is required for file system disks.  It
       is used to define the magnetic disks that make up the family
       set.  For a magnetic disk device, the Family Set field entry
       must match a Family Set defined on an ms or ma entry.

       The keyword shared must be specified in the Additional
       Parameters field if the file system is a shared file system.
       A shared file system is built by using the -S option to the
       sammkfs(1M) command. For more information on this option,
       see the sammkfs(1M) man page.

       For each disk device, the Equipment Identifier field is the
       path to a special file, such as /dev/dsk/cntndnsn.  If the
       meta devices are not present on the clients in a shared file
       system, the keyword nodev must be specified in the Equipment
       Identifier field for the mm devices.
```

The following equipment types are used to define the disk devices that reside within an ms or ma file system:

mm	A magnetic disk that is part of an ma disk cache family set. Metadata is allocated on this device. At least one mm device is required in an ma file system.
md	A magnetic disk that is part of an ms or ma disk cache family set. This device stores file data allocated in small Disk Allocation Units (DAUs) of 4 kilobytes and large DAUs of 16, 32, or 64 kilobytes. The default is 64 kilobytes. In an ms family set, this device stores both metadata and file data. In an ma family set, this device stores only file data. At least one md or mr device is required in an ma file system.
mr	A magnetic disk that is part of an ma disk cache family set. This device stores file data allocated in large Disk Allocation Units (DAUs) that are a multiple of 8 kilobytes in a fully adjustable range from 8 to 65528 kilobytes. The default is 64 kilobytes. File data is allocated on this device. At least one mr or md device is required in an ma file system.
gXXX	A magnetic disk that is part of an ma disk cache family set. The XXX identifies a striped group of devices. This device stores file data allocated in a large DAU size multiplied by the number of members in the striped group. The DAU size is a multiple of 8 kilobytes in a fully adjustable range from 8 to 65528 kilobytes. The default is 256 kilobytes. The XXX must be a decimal number in the XXXphysical size. It is not possible to use the samgrowfs(1M) command to increase the size of a striped group. However, it is possible to add additional striped groups.

The Equipment Identifier is used during the mount(1M) process as the Device To Mount. The Device To Mount is the first field in /etc/vfstab file for the mount point. For more information on this, see the mount(1M), mount_samfs(1M), or vfstab(1M) man pages.

SCSI-attached Libraries

Several identifiers can be used to define SCSI-attached libraries in the mcf file. For each SCSI-attached library, the Equipment Identifier field must contain the path (such as /dev/samst/cntnun) to the special file for the device created by the samst device driver. For more information on the device driver, see the samst(7) man page.

The Family Set field is required. It is used to associate the library controller with the drives in the library. All devices associated with the library must have the same

Family Set name.

The Additional Parameters field is optional. This field can be used to specify a nondefault location for the library catalog file. By default, catalogs are written to /var/opt/SUNWsamfs/catalog/family_set_name. This file is used to store information about each piece of media in the

library. In an HA-SAM configuration, this field must specify the library catalog file in the default location, and /var/opt/SUNWsamfs/catalog is linked to the cluster file system that is shared among all nodes within the Sun Cluster.

The following Equipment Type field entries can be used to define manually mounted or automated libraries that are attached through a SCSI interface:

Equipment Type Field Content	Definition
rb	Generic SCSI library that is automatically configured by SAM-QFS software. NOTE: An rb definition is preferred for all SCSI-attached libraries. The remainder of the library definitions in this list are supported but are not recommended for use in an mcf file. If a library in this list is defined in the mcf file as rb, SAM-QFS sets the appropriate type based on the SCSI vendor code.
ad	ADIC Scalar 448 libraries.
ae	ADIC Scalar 100 libraries.
al	Sun StorEdge L25 and L100 and ATL M1500 and M2500 libraries.
as	ADIC Scalar 1000 and Scalar 10K libraries.
q8	Qualstar 42xx, 62xx, 82xx, TLS and RLS series libraries
ov	Overland Data Inc. Neo Series Tape Libraries.
ac	ATL Products 4/52, 2640, 7100, and P-series tape libraries, and Sun 1800, 3500, L1000 and L11000 tape libraries.
cy	Cygnet optical disk libraries.
ds	DocuStore and Plasmon optical disk libraries.
eb	Exabyte 210, Sun L280, and ATL Products L-series tape libraries.
e8	Exabyte X80 libraries.

hc	HP L9/L20/L60 series
h4	HP SL48 and SL24 libraries.
hp	Hewlett Packard optical disk libraries.
ic	IBM 3570 media changer.
me	Metrum and Mountain Gate libraries.
nm	Fujitsu LT250 and LT270 libraries.
pd	Plasmon D-Series DVD-RAM libraries.
pg	Plasmon G-Series UDO/MO libraries. The library must be configured to G-Enterprise mode, element address scheme 1 and barcode type 2 or 3 by using the front panel.
ml	Quantum DLTx700 tape libraries.
dm	Sony DMF and DMS libraries.
cs	Sony CSM-20S Tape Library.
sl	Spectra Logic and Qualstar tape libraries.
s3	Sun StorageTek SL3000 library series.
s9	StorageTek 97xx series libraries.
sn	StorageTek L20, L40, L80, and L500 tape libraries and Sun StorEdge L7 and L8 autoloaders.
c4	Quantum PX500 and Sun StorEdge C4 libraries. These libraries are supported in native mode (PX500) only. SAM-QFS does not support these libraries in M1500 emulation mode.
il	IBM 3584 tape libraries.

Network-attached Libraries

This subsection describes how to define a network-attached library in your mcf file.

For each Network-attached library, the Equipment Identifier field must contain the path to the "parameters file" for the device.

The Family Set field is required. It is used to associate devices with the library. All devices associated with the

library must have the same Family Set name.

The Additional Parameters field is optional. This field can be used to specify a nondefault location for the library catalog file. By default, catalogs are written to

/var/opt/SUNWsamfs/catalog/family_set_name. This file is used to store information about each piece of media in the library. In an HA-SAM configuration, this field must specify the library catalog file in the default location, and /var/opt/SUNWsamfs/catalog is linked to the cluster file system that is shared among all nodes within the Sun Cluster.

The network-attached library definitions are as follows:

Equipment Type Field Content	Definition
gr	ADIC/GRAU Network-attached library. The Equipment Identifier field must contain the path to the parameters file for the grauaci interface. For more information, see the grauaci(7) man page.
im	IBM 3494 interface. The Equipment Identifier field must contain the path to the parameters file for the ibm3494 interface. For more information, see the ibm3494(7) man page.
pe	Sony network-attached interface. The Equipment Identifier field must contain the path to the parameters file for the sony interface. For more information, see the sony(7) man page.
sk	StorageTek ACSLS interface. The Equipment Identifier field must contain the path to the parameters file for the ACSLS interface. For more information, see the stk(7) man page.

The Historian

The hy identifier in the Equipment Type field identifies the SAM-QFS historian.

The Equipment Identifier field must contain the string historian.

The Family Set must contain a dash character (-).

The Additional Parameters field is optional. This field can be used to specify a nondefault location for the historian. By default, the historian is written to

/var/opt/SUNWsamfs/catalog/historian. This file is used to store information about the media handled by the historian. For more information, see the historian(7) man page.

Optical Disk Drives

This subsection describes the optical disk drive devices supported by SAM-QFS.

NOTE that optical disk drive devices are not supported on x64 platforms.

In the mcf file, a line describing an optical device must contain the following:

- o The Equipment Identifier field must be the path to the special file, such as /dev/samst/cntnun, for the samst device driver. For more information, see the samst(7) man page.

- o The Family Set field is used to associate the drive with the library that has the same Family Set. If the family set is defined as a dash (-), the drive is assumed to be manually loaded.

- o The Equipment Type field contains the optical drive identifier, as follows:

Equipment Type Field Content	Definition
od	Generic optical disk. A disk that is automatically configured by SAM-QFS. If you specify od, SAM-QFS sets the appropriate type based on the SCSI vendor code. NOTE that an od definition is preferred for all optical drives. If you specify od in the Equipment Type field, the SAM-QFS software sets the appropriate type based on the SCSI vendor code. The remainder of the definitions in this list are supported but are not recommended for use in an mcf file.
o2	12 inch WORM drive.
wo	5 1/4 inch optical WORM drive.
mo	5 1/4 inch erasable optical drive. The SAM-QFS environment supports disks with 512-, 1024-, and 2048-byte sectors.
pu	Plasmon UDO drive.
mf	IBM Multi Function optical drive.

Note that for all magneto-optical media, the default archmax value is 5 megabytes.

Tape Drives

This subsection describes the set of tape drives supported by SAM-QFS software for use in manually mounted and automated libraries.

A line in the mcf file for a tape drive must contain information in the following other fields:

- o The Equipment Identifier must be the path to the raw device, typically, /dev/rmt/nbn. However, it can be any symbolic link that also points to the proper special file in the /devices tree. You must specify the BSD no-rewind path.

If the device supports compression, then that path should be specified for better tape usage; except if the ST_AUTODEN_OVERRIDE drive option bit is set in an st.conf entry, you cannot specify a compression preference by changing the dev entry. Any attempt to specify compression is ignored. This is determined by the Solaris SCSI tape driver, st. The compression state of the drive is determined by its power-on default.

For more information, see the mtio(7) man page.

- o The Family Set field must be used to associate the device with the library that has the same Family Set name. If the family set is a dash character (-), then the device is assumed to be a manually loaded device.

- o The Additional Parameters is required for a tape drive if the Equipment Identifier field does not contain information in a /dev/rmt/* format (the standard st device driver). If specified, the Additional Parameters field must contain the path to the special file, such as /dev/samst/cntnun, for the samst device driver. For more information, see the samst(7) man page.

If SAM-QFS has access to a tape device, no other user should be allowed access the device during that period. SAM-QFS changes the mode on the path supplied in the mcf file to 0660 at startup, or when the device state moves from down to on. When the state moves from on to down, the mode is set to the value of tp_mode in the defaults.conf file. For more information, see tbe defaults.conf(4) man page.

The following list shows the tape drives for each type of tape media supported. The tape drives supported by SAM-QFS are as follows:

Equipment Type Field Content	Definition
tp	Generic tape drive. These tapes are automatically configured by SAM-QFS. NOTE that a tp definition is preferred for all tape drives. If you specify tp in the Equipment Type field, the SAM-QFS software sets the appropriate type based on the SCSI vendor code. The remainder of the definitions in this list are supported but are not recommended for use in an mcf file.
dt	DAT 4mm tape drive. In the defaults.conf file, the default block size keyword for this media is dt_blksize = 16.
lt	Digital linear tape (DLT) drive (including Super DLT and DLT-S4). In the defaults.conf

```
                         file, the default block size keyword for this
                         type of media is lt_blksize = 128.

          xt             Exabyte (850x) 8mm tape drive.  In the
                         defaults.conf file, the default block size
                         keyword for this media is xt_blksize = 16.

          xm             Exabyte Mammoth-2 8mm tape drive.  In the
                         defaults.conf file, the default block size
                         keyword for this media is xm_blksize = 128.

          fd             Fujitsu M8100 128-track tape drive.  In the
                         defaults.conf file, the default block size
                         keyword for this media is fd_blksize = 256.

          i7             IBM 3570 tape drive.  In the defaults.conf
                         file, the default block size keyword for this
                         media is i7_blksize = 128.

          li             IBM 3580, Seagate Viper 200 and HP Ultrium
                         (LTO) In the defaults.conf file, the default
                         block size keyword for this media is
                         li_blksize = 256.

          ib             IBM 3590 tape drive.  In the defaults.conf
                         file, the default block size keyword for this
                         media is ib_blksize = 256.

          m2             IBM 3592 J1A and E05 tape drives.  In the
                         defaults.conf file, the default block size
                         keyword for this media is m2_blksize = 2048.

          vt             Metrum VHS (RSP-2150) tape drive.  In the
                         defaults.conf file, the default block size
                         keyword for this media is vt_blksize = 128.

          at             Sony AIT tape drive.  In the defaults.conf
                         file, the default block size keyword for this
                         media is at_blksize = 128.

          sa             Sony Super AIT tape drive.  In the
                         defaults.conf file, the default block size
                         keyword for this media is sa_blksize = 2048.

          so             Sony DTF tape drive.  In the defaults.conf
                         file, the default block size keyword for this
                         media is so_blksize = 1024.

          st             StorageTek 3480 tape drive.  In the
                         defaults.conf file, the default block size
                         keyword for this media is st_blksize = 128.

          se             StorageTek 9490 tape drive.  In the
                         defaults.conf file, the default block size
                         keyword for this media is se_blksize = 128.

          sg             StorageTek 9840 tape drive.  In the
                         defaults.conf file, the default block size
                         keyword for this media is sg_blksize = 256.
```

```
     d3             StorageTek D3 tape drive.  In the
                    defaults.conf file, the default block size
                    keyword for this media is d3_blksize = 256.

     sf             StorageTek T9940 tape drive.  In the
                    defaults.conf file, the default block size
                    keyword for this media is sf_blksize = 256.

     ti             StorageTek Titanium tape drive.  In the
                    defaults.conf file, the default block size
                    keyword for this media is ti_blksize = 2048.
```

For all tapes, the SAM-QFS system sets the block size to a media-specific default. For information on how to change the default block size, see the defaults.conf(4) man page.

For all tapes, the default archmax value is 512 megabytes.

Disk Archiving

The archiver can be configured to archive directly to online

disk cache. To enable disk archiving, you must perform the following steps:

1. Create directories in online disk cache to serve as destinations for the archive copies.

2. Create the /etc/opt/SUNWsamfs/diskvols.conf file.

3. Edit the archiver.cmd file and add the -disk_archive directive.

The media type for a disk volume is dk. The block size for a disk volume is dk_blksize=1024. This value cannot be changed.

The media type for a Sun StorageTek 5800 Storage System disk volume is cb. The 5800 schema specifies the metadata attributes that are stored with objects in the 5800 system. The system comes preconfigured with a default metadata schema. For a 5800 disk volume you must modify the default schema file to add metadata specific to SAM-QFS. For more information on configuring the schema, refer to the Sun StorageTek 5800 System Administration Guide. The file /opt/SUNWsamfs/examples/metadata_config_samfs.xml can be used to extend the default schema for SAM-QFS.

Disk archiving is explained in more detail in the Sun Storage Archive Manager Installation and Configuration Guide and in the Sun Storage Archive Manager Configuration and Administration Guide.

SAM-Remote Device Definitions

Several identifiers define devices when using the Sun SAM-Remote client or Sun SAM-Remote server software. For more information on configuring the Sun SAM-Remote client or the Sun SAM-Remote server, see the sam-remote(7) man page or see the Sun SAM-Remote Administrator’s Guide.

The identifiers used when configuring the Sun SAM-Remote client or Sun SAM-Remote server are as follows:

Equipment Type Field Content	Definition
ss	Sun SAM-Remote server. The Equipment Identifier field must contain the path name to the server configuration file. The Family Set field must identify the server. That is, it must be the same as the Family Set name of the server. It must match the name used in the client side definition. It is used by the clients to associate the device with the server of the same Family Set name.
sc	Sun SAM-Remote client. The Equipment Identifier field must contain the path name to the client configuration file. The Family Set field must contain an identifier that is the same as the family set name of the server. It is used by the clients to associate the device with the server of the same Family Set name. The Additional Parameters field must contain the full path name of the client's library catalog file.
rd	Sun SAM-Remote pseudo-device. The Equipment Identifier field must be the path to the pseudo-device, such as /dev/samrd/rd2. The Family Set field must be the name of the server. It is used by the clients to associate the device with the server of the same Family Set name.

FILES

/opt/SUNWsamfs/examples Contains example mcf files.

SEE ALSO

Sun SAM-Remote Administrator's Guide.

Sun QFS File System Administration Guide.

Sun Storage Archive Manager Administration Guide.

chmod(1).

build_cat(1M), dump_cat(1M), mount(1M), mount_samfs(1M), sammkfs(1M). sam-fsd(1M),

defaults.conf(4), inquiry.conf(4), vfstab(4).

dst(7), fujitsulmf(7), grauaci(7), historian(7), ibm3494(7), mtio(7), sam-remote(7), samst(7), sony(7), st(7), stk(7).

notify.cmd(4)

```
NAME
     notify.cmd - Sun QFS or SAM-QFS email subscriptions commands
     file

SYNOPSIS
     /etc/opt/SUNWsamfs/notify.cmd

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The Sun QFS or SAM-QFS system has the ability  to  inform  a
     user of certain events or conditions by generating a message
     and automatically sending email notifications.

     The /etc/opt/SUNWsamfs/notify.cmd stores the email addresses
     for the following notifications:  No space available on file
     system  (ENospace),  Recovery  Point  warnings   (DumpWarn),
     Recovery   Point   errors   (DumpInterrupted),  File  System
     exceeded its high water mark (HwmExceeded), ACSLS configura-
     tion  warnings  (AcslsWarn),  and ACSLS configuration errors
     (AcslsErr).

     The email subscriptions are added, modified and deleted only
     via  the SAM-QFS Manager software, a browser-based graphical
     user interface to the Sun QFS or SAM-QFS software.

     This file is created automatically  after  installation  and
     'root' is assigned as the default subscriber for all notifi-
     cations.

     If you would like to add, modify  or  delete  subscriptions,
     use   the  Email  Alerts  feature  in  the  SAM-QFS  Manager
     software.

     To  preserve  compatibility, the  email  subscriptions  for
     Archiving  is Interrupted, Recycling is complete, Library or
     Tape drive is down and, Requested volume is unavailable, are
     automatically added to the following:
     /etc/opt/SUNWsamfs/scripts/archiver.sh
     /etc/opt/SUNWsamfs/scripts/recycler.sh
     /etc/opt/SUNWsamfs/scripts/dev_down.sh
     /etc/opt/SUNWsamfs/scripts/load_notify.sh

MANUAL EDITING
     The /etc/opt/SUNWsamfs/notify.cmd consists of the  notifica-
     tion  type followed by the list of email addresses that have
     subscribed to it.

     Each notification type and its  respective  subscriber  list
     are  space  separated,  while  the email addresses are comma
     separated.

     It is is highly recommended that you only  use  the  SAM-QFS
     Manager  software  to  add, modify, or delete subscriptions,
     but if you must manually modify  this  file,  take  care  to
```

preserve the formatting of this file.

EXAMPLE

The following is an example /etc/opt/SUNWsamfs/notify.cmd file:

```
    DumpInterrupted root, samadmin@xxx
    ENospace root
    HwmExceeded
```

The above entries indicate that root is to receive email notifications if the file system is full or if errors are encountered when taking recovery points. There are no subscribers for the 'File System exceeded its high water mark' notification.

To remove 'root' as a subscriber, the file should now read as follows:

```
    DumpInterrupted samadmin@xxx
    ENospace
    HwmExceeded
```

SEE ALSO

sendtrap(1M).
defaults.conf(4).

nrecycler.cmd(4)

NAME

nrecycler.cmd - SAM-QFS sam-nrecycler commands file

SYNOPSIS

/etc/opt/SUNWsamfs/nrecycler.cmd

AVAILABILITY

SUNWsamfs

DESCRIPTION

Commands for controlling sam-nrecycler(1M) are read from /etc/opt/SUNWsamfs/nrecycler.cmd. These commands are given one per line.

logfile = filename
     Set the name of the log file to filename.

SAMFSDUMP DEFINITIONS SECTION

The samfsdumps and endsamfsdumps directives delimit this section of the nrecycler.cmd file.

The samfsdump definitions section defines the SAM-QFS dump files for archiver retention capabilities. Each line specifies the path to a directory containing SAM-QFS dump files. The user is responsible for making sure the list of directories is complete and all SAM-QFS dumps files are contained in the directory list. The nrecycler cannot validate the

SAM-QFS dump file list.

EXAMPLE

The following is an example /etc/opt/SUNWsamfs/nrecycler.cmd file:

```
samfsdumps
/samdumps
endsamfsdumps
```

SEE ALSO

sam-nrecycler(1M).

preview.cmd(4)

NAME

preview.cmd - SAM-QFS preview directives file

SYNOPSIS

/etc/opt/SUNWsamfs/preview.cmd

AVAILABILITY

SUNWsamfs

DESCRIPTION

An archive or stage request for a volume that is not currently loaded goes to the preview area for future consideration. A user can control the scheduling of preview requests, thus overriding the default behavior, by entering directives in the preview.cmd file.

The preview.cmd file contains directives for modifying preview request priorities. The directives allow users to increase the priority for specific VSNs and change archive request priorities based on the file system states regarding High Water Mark (HWM) and Low Water Mark (LWM). These directives are read by sam-amld at start-up time, and all values specified are stored in shared memory. The priority specifications cannot be changed while the sam-amld daemon is running.

The preview.cmd file can contain comments. A comment begins with a pound character (#) and extends through the end of the line.

DIRECTIVES

The directives in the preview.cmd file are specified one per line. With regard to their placement within the preview.cmd file, there are two types of directives:

o Global directives. These directives apply to all file systems. Directives are assumed to be global if they appear in the preview.cmd file prior to any fs = directives.

o Directives specific to a particular file system. File

```
     system specific directives must appear after the global
     directives in the preview.cmd file.  A directive line with
     the following form names a specific file system and
     indicates that all subsequent directives apply only to
     that file system:

     fs = file_system_family_set_name

     A subsequent fs = directive in the preview.cmd file
     declares a set of directives that apply to another file
     system.  File system specific directives override general

     directives.

   Some directives can be used as both global and file system
   specific directives.  This can be useful, for example, if
   you want to specify the hwm_priority directive globally to
   apply to most SAM-QFS file systems but you also want to use
   it as a file system specific directive to specify a
   different value for one particular file system.

   The following sections describe the directives that can
   appear in a preview.cmd file.  You can specify either an
   integer or a floating point value as an argument to the
   _priority directives, but the system stores the value as a
   floating point value internally.

GLOBAL DIRECTIVES
   Global directives must appear in the preview.cmd file before
   any fs = directives.  They cannot appear after an fs =
   directive.  The global directives are as follows:

   vsn_priority = value
             This directive specifies the value by which the
             priority is to increase for VSNs marked as high-
             priority VSNs.  For more information, see the
             chmed(1M) man page.  The vsn_priority = 1000.0 by
             default.

   age_priority = factor
             This global directive specifies a factor to to be
             applied to the time (in seconds) that a request is
             allowed to wait in the preview area to be
             satisfied.  The factor is as follows:

             o A factor > 1.0, increases the weight of the time
               when calculating the total priority.

             o A factor < 1.0, decreases the weight of the time
               when calculating the total priority.

             o A factor = 1.0 has no effect on the default
               behavior.  The age_priority = 1.0 by default.

   For more information, see the PRIORITY CALCULATION section
   of this man page.

FILE SYSTEM SPECIFIC DIRECTIVE
   The fs = directive specifies a particular file system and
```

```
     applies only to that specified file system.  This
     directive’s syntax is as follows:

     fs = file_system_family_set_name
               This directive indicates that the subsequent

               directives apply only to the indicated
               file_system_family_set_name.

  GLOBAL OR FILE SYSTEM SPECIFIC DIRECTIVES
     Several directives can be used either globally or as file
     system specific directives.  These directives are as
     follows:

     hwm_priority = value
               This directive indicates the value by which the
               priority is to increase for archiving requests
               versus staging after the file system crosses the
               HWM level.  This means that the releaser is
               running.  The hwm_priority = 0.0 by default.

     hlwm_priority = value
               This directive indicates the value by which the
               priority is to increase for archiving requests
               versus staging.  This directive is effective when
               the file system is emptying, and the amount of
               data is between the HWM and the LWM.  Because the
               file system is emptying, you may want to give
               priority to loads for stage requests.  The
               hlwm_priority = 0.0 by default.

     lhwm_priority = value
               This directive indicates the value by which the
               priority is to increase for archiving requests
               versus staging.  This directive is effective when
               the file system is filling up, and the amount of
               data is between the HWM and the LWM.  Because the
               file system is filling up, you may want to give
               priority to loads for archive requests.  The
               lhwm_priority = 0.0 by default.

     lwm_priority = value
               This directive specifies the value by which the
               priority is to increase for archiving requests
               versus staging when the file system is below the
               LWM level.  The lwm_priority = 0.0 by default.

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

     Total priority = vsn_priority + wm_priority + age_priority *
     time_in_sec

     The wm_priority in the previous equation refers to whichever
     condition is in effect at the time, either hwm_priority,
     hlwm_priority, lhwm_priority, or lwm_priority.  All
     priorities are stored as floating point numbers.
```

```
EXAMPLES
     Example 1.  This example preview.cmd file sets both the
     vsn_priority and hwm_priority for the samfs1 file system.
     Other SAM-QFS file systems not specified here use the
     default priority for the HWM.  All file systems use the
     default priorities for the LWM and the state between LWM and
     HWM.

     vsn_priority = 1000.0
     fs = samfs1
     hwm_priority = 100.0

     Example 2.  The next example preview.cmd file sets priority
     factors for all SAM-QFS file systems, but it sets an
     explicit and different HWM priority factor for the samfs3
     file system.

     hwm_priority = 1000.0
     hlwm_priority = -200.0
     lhwm_priority = 500.0
     fs = samfs3
     hwm_priority = 200.0

SEE ALSO
     chmed(1M), sam-amld(1M).
```

recycler.cmd(4)

```
NAME
     recycler.cmd - SAM-QFS sam-recycler commands file

SYNOPSIS
     /etc/opt/SUNWsamfs/recycler.cmd

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     Commands for  controlling  sam-recycler(1M)  are  read  from
     /etc/opt/SUNWsamfs/recycler.cmd.   These  commands are given
     one per line.

     logfile = filename
          Set the name of the log file to  filename.   This  file
          shows  the  overall media utilization and a sorted list
          of VSNs in the order in which they  will  be  recycled.
          The  default  is no log file.  See sam-recycler(1M) for
          more information.

     no_recycle media-type VSN-regexp [VSN-regexp...]
          Disallow sam-recycler(1M) from recycling the VSNs which
          match  the  media-type  and  the regular expression(s),
          VSN-regexp.

     robot-family-set parameters
          This command sets recycling parameters for a particular
          library  identified  by  robot-family-set  (this is the
          name   given   as   the   fourth   field   in   the
          /etc/opt/SUNWsamfs/mcf  file  line defining the library
          for which you wish to set the parameters).

          parameter
               can be one more of the following:

          -dataquantity size
               This parameter sets a limit of size bytes  on  the
               amount  of  data  the  recycler  will schedule for
               rearchiving in order to clear  volumes  of  useful
               data.   Note  that  the  actual  number of volumes
               selected for recycling may also  be  dependent  on
               the  -vsncount  parameter.  The default is 1 giga-
               byte (1G).

          -hwm percent
               establishes the  high-water  mark  for  the  media
               utilization in the indicated library, specified as
               an integer percentage of total capacity.  When the
               utilization  of  those  volumes  exceeds  percent,
               sam-recycler(1M)  will  begin   to   recycle   the
               library.  The default is 95.

          -ignore
               will keep sam-recycler(1M) from selecting any can-
               didates from the specified library.  The intent of
               this parameter is to allow  a  convenient  way  of
               testing other parameters.

          -mail mailaddress
               will cause sam-recycler(1M) to mail a  message  to
               the  indicated  mailaddress when a library's media
               utilization exceeds the high-water mark.  Omission
               of mailaddress prevents recycling.  If you specify
               -mail, you must specify a valid mailaddress.

          -mingain percent
               This parameter limits  selection  of  volumes  for
               recycling to those which would increase their free
               space by percent or more.  Volumes not meeting the
               -mingain  parameter are not recycled.  The default
               is based on the capacity  of  the  volume  (<200GB
               60%, >=200GB 90%).

          -vsncount count
               This parameter sets a limit of count on the number
               of  volumes  the  recycler will schedule for rear-
               chiving in order to clear volumes of useful  data.
               Note  that  the  actual number of volumes selected
               for recycling may also be dependent on the  -data-
               quantity parameter.  The default is 1.

     To     preserve     compatibility     with      pre-existing
     /etc/opt/SUNWsamfs/recycler.cmd  files, an alternative, less
     powerful, syntax is allowed for the library recycling param-
     eters command.
```

robot-family-set robot-high-water VSN-minimum-percent-gain options
This command sets recycling parameters for a particular library identified by robot-family-set (this is the name given as the fourth field in the /etc/opt/SUNWsamfs/mcf file line defining the library for which you wish to set the parameters). robot-high-water establishes the high-water mark for the media utilization in the indicated library, specified as an integer percentage of total capacity. When the utilization of those volumes exceeds percent, sam-recycler(1M) will begin to recycle the library. The VSN-minimum-percent-gain (aka min-gain) value specifies a threshold of space available to be reclaimed (as an integer percent of total capacity of the VSN) below which VSNs will not be selected for recycling. The options consist of zero or more of the following: ignore - which will keep sam-recycler(1M) from

selecting any candidates from the specified library. mail mailaddress - which will cause sam-recycler(1M) to mail a message to the indicated mailaddress when a library's media utilization exceeds the high-water mark. Omission of mailaddress prevents any mail from being sent.

script = filename
Supply the name of the file executed when a volume is to be relabeled. The default is /etc/opt/SUNWsamfs/scripts/recycler.sh

ARCHIVER'S COMMAND FILE

The archiver's command file, /etc/opt/SUNWsamfs/archiver.cmd, can also specify recycling parameters for archive sets. Each archive set which has recycling parameters applied in /etc/opt/SUNWsamfs/archiver.cmd will be considered as a pseudo library containing just the VSNs which the archiver assigns to the archive set. See archiver.cmd(4) for more information. Archive set names may not be specified in the /etc/opt/SUNWsamfs/recycler.cmd file.

DEFAULT FILE

If there is no /etc/opt/SUNWsamfs/recycler.cmd file, then, for each library, a line is constructed:

```
library -dataquantity 1G -hwm 95 -ignore -mail root -mingain
50 -vsncount 1
```

and logging is disabled.

EXAMPLE

The following is an example /etc/opt/SUNWsamfs/recycler.cmd file:

```
logfile = /var/adm/recycler.log
lt20 -hwm 75 -mingain 60 -ignore
hp30 -hwm 90 -mingain 60 -mail root
gr47 -hwm 95 -mingain 60 -ignore mail root
```

```
                  no_recycle lt DLT.*

          The results  of  sam-recycler(1M)  operation  are  found  in
          /var/adm/recycler.log.   Three  libraries  are  defined with
          various high-water marks.  The first library  is  not  recy-
          cled,  but  the  usage  information for the VSNs it contains
          will appear in the log, and no mail will be generated.   The
          second  library  is  recycled  (that is, VSNs are emptied of
          valid archive images and relabeled) and root is sent  e-mail
          when the library exceeds the 90% high-water mark.  The third

          library is not recycled,  but  root  is  notified  if  usage
          exceeds the high-water mark.

          For hp30, only VSNs whose recycling would free up  at  least
          60% of the capacity of the VSN are considered.

          No medium which is of media type lt  and  whose  VSN  begins
          with DLT will be recycled.

     SEE ALSO
          sam-recycler(1M).
          archiver.cmd(4), mcf(4).
```

releaser.cmd(4)

```
     NAME
          releaser.cmd - SAM-QFS releaser command file

     SYNOPSIS
          /etc/opt/SUNWsamfs/releaser.cmd

     AVAILABILITY
          SUNWsamfs

     DESCRIPTION
          Directives for controlling the releaser can be read from the
          /etc/opt/SUNWsamfs/releaser.cmd file.  The directives must
          appear one per line.

          Comment lines are permitted.  Comment lines must begin with
          a pound character (#), and the comment can extend through
          the rest of the line.

          Directives that appear prior to any fs= directive are
          applied to all file systems.  Directives that appear after a
          fs= directive are applied to the specified file system only.
          Directives that are specific to a file system override
          general directives.

          The directives on this man page are divided into groups.
          The weight directives for size and age determine the release
          priority of a file.  The miscellaneous directives control
          whether a log file is written, whether there is a minimum
          age required for files, and other aspects of releasing.
```

WEIGHT DIRECTIVES

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

Size Priority

The size priority is determined by the value of the weight_size directive. This directive has the following format:

weight_size=weight_size_value

Sets the weight factor for the size of the file to weight_size_value. Specify a floating-point number in the following range:
0.0 < weight_size_value < 1.0. The default is 1.0.

The weight_size_value is multiplied by the size of the file in 4-kilobyte blocks to arrive at the size component of the file's release priority.

Age Priority

The age priority can be calculated in one of the following ways:

- The first method multiplies the value of the weight_age= directive by the most recent of the following ages: access age, modify age, and residence change age. The access age is defined as the current time minus the file's last access time. The weight_age directive has the following format:

  weight_age=weight_age_value

  Sets the weight factor for the overall age of the file to weight_age_value. The weight_age_value is multiplied by the most recent of the file's access, modify or residence change age to arrive at the age component of the file's release priority. Specify a floating-point number in the following range:
  0.0 < weight_age_value < 1.0. The default is 1.0.

  If you specify a weight_age= directive for a given file system, you cannot specify weight_age_access=, weight_age_modify=, or weight_age_residence= directives for the same file system.

- The second method allows you to specify separate weights for the access, modify, and residence ages. The ages are calculated in units of 60-second minutes.

  If you want to specify separate weights for the access, modify, and residence ages, use the following directives in the releaser.cmd file:

  weight_age_access=weight_age_access_value

  Sets the weight factor for the access age of the file to weight_age_access_value. Specify a floating-point number in the following range:

```
          0.0 < weight_age_access < 1.0.  The default is 1.0.

          The weight_age_access_value is multiplied by the
          file’s access age (expressed in minutes).  This
          product, added to the sum of the products of the
          modify and residence-change ages multiplied by their
          respective weights, becomes the age component of the
          file’s release priority.

          If you specify a weight_age= directive for a given
          file system, you cannot specify a weight_age_access=
          directive for the same file system.

     weight_age_modify=weight_age_modify_value
          Sets the weight factor for the modify age of the

          file to weight_age_modify_value.  Specify a
          floating-point number in the following range:
          0.0 < weight_age_modify < 1.0.  The default is 1.0.

          The weight_age_modify_value is multiplied by the
          file’s modify age (expressed in minutes).  This
          product, added to the sum of the products of the
          modify and residence-change ages multiplied by their
          respective weights, becomes the age component of the
          file’s release priority.

          If you specify a weight_age= directive for a given
          file system, you cannot specify a weight_age_modify=
          directive for the same file system.

     weight_age_residence=weight_age_residence_value
          Sets the weight factor for the residence-change age
          of the file to weight_age_residence_value.  Specify
          a floating-point number in the following range:
          0.0 < weight_age_residence < 1.0.  The default is
          1.0.

          The weight_age_residence_value is multiplied by the
          file’s residence-change age (expressed in minutes).
          This product, added to the sum of the products of
          the modify and residence-change ages multiplied by
          their respective weights, becomes the age component
          of the file’s release priority.

          If you specify a weight_age= directive for a given
          file system, you cannot specify a
          weight_age_residence= directive for the same file
          system.

MISCELLANEOUS DIRECTIVES
     The following miscellaneous directives can be specified in
     the releaser.cmd file:

     fs = file_system_family_set_name
          Specifies to the releaser that the subsequent
          directives apply to the indicated
          file_system_family_set_name only.
     list_size = number
          Sets the number of candidate files for release during
          one pass of the file system.  For number, specify an
          integer number in the following range:
          10 < number < 2,147,483,648
          The default is based on the size of the .inodes file.
          If there is enough space for one million inodes (512-
          bytes/inode), number is 100000, otherwise it is 30000.
          If you have many small files in your file system you

          may want to increase this number.

     no_release
          Prevents the releaser from releasing any files.  This
          directive is useful when you are tuning the priority
          weights.  Also see the display_all_candidates
          directive.  By default, files are released.

     rearch_no_release
          Prevents the releaser from releasing files marked to be
          rearchived.  By default, files marked for rearchive are
          released.

     logfile = filename
          Sets the name of the releaser's log file to filename.
          By default, no log file is written.

     display_all_candidates
          Writes the releaser priority for each file, as it is
          encountered, to the log file.  This can be useful in
          tuning when used in conjunction with the no_release
          directive.  This directive allows you to judge the
          effect of changing the priority weights.  By default
          file priority is not displayed in any way.

     min_residence_age = time
          Sets the minimum residency age to time seconds.  This
          is the minimum time a file must be online before it is
          considered to be a release candidate.  The default is
          600 seconds (10 minutes).

EXAMPLES
     Example 1.  This example file sets the weight_age= and
     weight_size= directives for the samfs1 file system.  No
     releaser log is produced.

               fs = samfs1
               weight_age = .45
               weight_size = 0.3

     Example 2.  This example provides weights for all file
     systems.  All file system releaser runs are logged to
     /var/adm/releaser.log.

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

     Example 3.  This example specifies weights and log files for
```

     each file system.

```
          logfile = /var/adm/default.releaser.log

          fs = samfs1

          weight_age = 1.0
          weight_size = 0.0
          logfile = /var/adm/samfs1.releaser.log

          fs = samfs2

          weight_age_modify = 0.3
          weight_age_access = 0.03
          weight_age_residence = 1.0
          weight_size = 0.0
          logfile = /var/adm/samfs2.releaser.log
```

     Example 4. This example is identical in function to example
     3, but it specifies the weight_size= and list_size=
     directives globally.

```
          logfile = /var/adm/default.releaser.log
          weight_size = 0.0
          list_size = 100000

          fs = samfs1

          weight_age = 1.0
          logfile = /var/adm/samfs1.releaser.log

          fs = samfs2

          weight_age_modify = 0.3
          weight_age_access = 0.03
          weight_age_residence = 1.0
          logfile = /var/adm/samfs2.releaser.log
```

SEE ALSO
     release(1).
     sam-releaser(1M).

rft.cmd(4)

NAME
     rft.cmd - SAM-QFS file transfer server directives file (was
     ftp.cmd)

SYNOPSIS
     /etc/opt/SUNWsamfs/rft.cmd

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     Directives for controlling the SAM-QFS file transfer server
     are read from /etc/opt/SUNWsamfs/rft.cmd. In the rft.cmd

file, each directive must appear on its own line. Each directive has the following format:

```
keyword = value
```

Comment lines can appear in the rft.cmd file. A pound sign (#) in column 1 indicates a comment line.

The rft.cmd file accepts the following directives:

logfile = filename
: Sets the name of the rft log file to filename, specified as an absolute pathname. By default, no log file is written.

  The rft log file contains a line for each file transferred. The line contains the date, time, and the name of the file.

tcpwindow = size
: Sets the TCP window size for the data connection. size may be specified with the suffixes ’b’, ’k’, ’M’, ’G’, and ’T’, for bytes, kilobytes, megabytes, gigabytes, and terabytes. The default unit size is bytes. The default value is 0.

blksize = size
: Sets the amount of data to send down the socket at a time. size may be specified with the suffixes ’b’, ’k’, ’M’, ’G’, and ’T’, for bytes, kilobytes, megabytes, gigabytes, and terabytes. The default unit size is bytes. The default value is 1024K bytes. This parameter is used by the remote archive client to set its write block size. Increasing this value may improve archiving performance over high latency WANs.

EXAMPLES

The following is an example /etc/opt/SUNWsamfs/rft.cmd file:

```
logfile = /var/opt/SUNWsamfs/log/rft
```

The results of the rft file transfer daemon’s operations are found in /var/opt/SUNWsamfs/log/rft.

FILES

The following files are used by the file transfer server:

/etc/opt/SUNWsamfs/rft.cmd	File transfer server command file.

SEE ALSO

sam-rftd(1M).

samdb.conf(4)

```
NAME
     samdb.conf - SAM-QFS mySQL database access file

SYNOPSIS
     /etc/opt/SUNWsamfs/samdb.conf

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The  file  /etc/opt/SUNWsamfs/samdb.conf   contains   access
     parameters  to  the  mySQL  database for each SAM-QFS family
     set.

     Each entry is a single line of the form:

     family_set_name:host:user:
     password:database name:port:
     :client_flag:mount_point

     where

     family_set_name is the SAM-QFS family set name.

                     The family_set_name field  must  contain  at
                     least  one  character and must not contain a
                     colon (:) or a newline (\n).

     host            is  the  hostname  of  the  mySQL   database
                     server.   It  may be either a hostname or an
                     IP address. If host is NULL  or  the  string
                     "localhost",  a connection to the local host
                     is assumed.

     user            is the MySQL login ID.

     password        is the MySQL password corresponding with the
                     user name.

     database_name   is the MySQL database name.

     port            is the TCP/IP port being used by the SAM-QFS
                     database  server.   If  the value of port is
                     blank or 0, the default  of  3306  is  used.
                     Note,  the  port  field  is  inoperative for
                     localhost databases.

     client_flag     is the value of the client flag.  See  mySQL
                     function mysql-real-connect() for details.

     mount_point     is the file system   mount  point  for  this
                     family set.

     Blank lines are treated as malformed entries and will  cause
     consumers of the file to fail.
```

EXAMPLES

The following is a sample samdb.conf file:

```
samfs1:db.oracle.com:3ksnn64:secret:samfs1:7009::/sam/sam1
samfs2:localhost:laura:secret:samfs2test:::/sam/sam2
```

In this example, two family sets are represented. The first line shows samfs1 which connects to the database on db.oracle.com via TCP port 7009 with database name samfs1. The second line shows samfs2 which connects to the database on localhost with database name samfs2test.

samfs.cmd(4)

NAME

samfs.cmd - Defines mount parameters for Sun QFS and SAM-QFS file systems

SYNOPSIS

/etc/opt/SUNWsamfs/samfs.cmd

AVAILABILITY

SUNWqfs

SUNWsamfs

DESCRIPTION

Commands for controlling samfs mount parameters are read from /etc/opt/SUNWsamfs/samfs.cmd. These commands serve as defaults, and can be superseded by parameters on the mount command. See mount_samfs(1M). The /etc/opt/SUNWsamfs/samfs.cmd file is read when sam-fsd is started. You can change it at any time while sam-fsd is running. The changes take effect when sam-fsd is restarted, or sent the signal SIGHUP via the samd config command.

When changing mount options in this file, you must unmount and mount the file system in order for the new mount options to take effect.

These commands are given one per line. Comments begin with a # and extend through the end of the line. Commands given before any "fs =" line apply in general to all file systems; "fs =" introduces commands which are specific to the mentioned file system only. File system-specific commands override general commands.

COMMANDS

See mount_samfs(1M) under OPTIONS for the list of supported commands. The following additional command is available as well.

fs = fs_name

This command specifies the following commands apply only to the indicated file system with family set name fs_name.

```
EXAMPLE
     This example file sets high and low  for  2  different  file
     systems, samfs1 and samfs2.

               fs = samfs1
                 high = 90
                 low = 80
               fs = samfs2
                 high = 80

                 low = 75

SEE ALSO
     release(1), setfa(1).
     mount_samfs(1M), sam-fsd(1M), sam_releaser(1M).
     sam_advise(3), sam_setfa(3).
     directio(3C).
     mcf(4).
```

sefdata(4)

```
NAME
     sefdata - Collects System Error Facility (SEF) data for
     SAM-QFS file systems

SYNOPSIS
     /var/opt/SUNWsamfs/sef/sefdata

     #include "/opt/SUNWsamfs/include/sefvals.h"

     #include "/opt/SUNWsamfs/include/sefstructs.h"

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The sefdata file contains the data gathered from the log
     sense pages of peripheral tape devices used by SAM-QFS file
     systems.  Each time the SAM-QFS software unloads a cartridge
     from a drive, pertinent log sense pages are obtained from
     the device, and a record is written to the sefdata file.
     Each record consists of a header followed by some number of
     log sense pages.

     The record header has the format of a sef_hdr structure.
     This structure is defined in
     /opt/SUNWsamfs/include/sefstructs.h, and it has the
     following components:

     struct sef_hdr {
         uint_t      sef_magic;          /* magic # for app to sync file posn */
         uint_t      sef_version;        /* version number */
         uint_t      sef_size;           /* size of this record, excl. header */
         uint16_t    sef_eq;             /* equipment number of this device */
         char        sef_devname[128];   /* pathname of device */
```

```
            uchar_t     sef_vendor_id[9];   /* vendor id from inquiry */
            uchar_t     sef_product_id[17]; /* product id from inquiry */
            uchar_t     sef_revision[5];    /* revision level from inquiry */
            uchar_t     sef_scsi_type;      /* device type from inquiry */
            vsn_t       sef_vsn;            /* vsn of media that was mounted */
            time_t      sef_timestamp;      /* timestamp of this record */
        }
```

The fields of the sef_hdr structure have the following meanings:

Field	Content
sef_magic	Has the value SEFMAGIC, as defined in /opt/SUNWsamfs/include/sefvals.h.
sef_version	Has the value SEFVERSION, as defined in /opt/SUNWsamfs/include/sefvals.h.
sef_size	The size of this record, excluding the header.
sef_eq	The equipment number of the device, as configured in the mcf file. For more information, see the mcf(4) man page.
sef_devname	A character string containing the path name of the device.
sef_vendor_id	The vendor identification of the device, as obtained from inquiry.
sef_product_id	The product identification of the device, as obtained from inquiry.
sef_revision	The revision level of the device, as obtained from inquiry.
sef_scsi_type	The device type, as obtained from inquiry.
sef_vsn	Volume Serial Name (VSN) of the volume mounted in the device when the data was generated.
sef_timestamp	Time that this record as written to the data file.

Following the header in each record is some number of log sense pages. Each log sense page consists of a SCSI-standard header followed by triplets of parameter codes, control values, and parameter values. For the exact format of the log sense pages returned by the devices in use at your site, consult the documentation provided with those devices.

FILES

File	Purpose
/var/opt/SUNWsamfs/sef/sefdata	

```
                         Contains SEF information.

     /opt/SUNWsamfs/include/sefvals.h
                         Contains values, such as those for
                         SEFMAGIC and SEFVERSION.

     /opt/SUNWsamfs/include/sefstructs.h
                         Contains include files for the SEF
                         header, the SCSI-standard header, and
                         other structures.

SEE ALSO
     Sun Storage Archive Manager Configuration and Administration
     Guide.
     sefreport(1M), sefsysevent(4).
```

sefsysevent(4)

```
NAME
     sefsysevent - SEF sysevent

SYNOPSIS
     /etc/sysevent/config/SUNW,SUNWsamfs,Device,sysevent.conf

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     In SAM-QFS environments, tape drive SCSI log sense error
     counter pages 2 and 3 for media analysis are available to
     the user via a Solaris sysevent.  SEF (System Error
     Facility) sysevents are enabled by default with a default
     polling interval of once before unload.  SEF sysevents
     behavior is controlled by defaults.conf and samset.

     How to add a sysevent handler for SEF sysevents:

     A simple SEF sysevent handler should be executable and may
     look like this:

     #!/bin/ksh
     echo "$@" >> /var/tmp/xx.dat
     exit 0

     To add the SEF sysevent handler to the syseventd(1M):

     # syseventadm add -vSUNW -pSUNWsamfs -cDevice -sSEF
     /var/tmp/xx \"\$VENDOR\" \"\$PRODUCT\" \"\$USN\" \"\$REV\"
     \$TOD \$EQ_ORD \"\$NAME\" \$INQ_TYPE \"\$MEDIA_TYPE\"
     \"\$VSN\" \$LABEL_TIME \$LP2_PC0 \$LP2_PC1 \$LP2_PC2
     \$LP2_PC3 \$LP2_PC4 \$LP2_PC5 \$LP2_PC6 \$LP3_PC0 \$LP3_PC1
     \$LP3_PC2 \$LP3_PC3 \$LP3_PC4 \$LP3_PC5 \$LP3_PC6 \$WHERE
     \$sequence

     The syseventadm(1M) add command above creates the
     /etc/sysevent/config/SUNW,SUNWsamfs,Device,sysevent.conf
```

file and a path to your SEF sysevent handler /var/tmp/xx.
Note the double quotes are required when using the
syseventadm(1M) command because the strings can be empty and
the data is positional.

To load the SEF sysevent handler:

```
# syseventadm restart
```

SEF sysevent event handler data looks like this:

```
# cat /var/tmp/xx.dat
"HP      " "Ultrium 2-SCSI  " "HUL2M00585" "F45H" 1094048112 82
"/dev/rmt/2cbn" 0x1 "li" "000750" 1091738029 0x0 0x0 0x0 0x0 0x0 0x70b1

0x0 0x0 0x0 0x0 0x322 0x322 0x4645 0x0 0x1 0x282
"HP      " "Ultrium 2-SCSI  " "HUL2M00617" "F5AH" 1094048116 81
"/dev/rmt/1cbn" 0x1 "li" "NAB975" 1092691221 0x0 0x0 0x0 0x0 0x0 0x35c
0x0 0x0 0x0 0x0 0x0 0x0 0x4a 0x0 0x1 0x283
```

A C language program to convert time of day $TOD and
$LABEL_TIME sysevent macros from digits to text:

```
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>
#include <time.h>

void main(int argc, char **argv)
{
         char str[100];
         time_t tm = atol(argv[1]);

         cftime(str, "%C", "tm);
         printf("%s0, str);
}
```

The complied sefsysevent_time program then can be used to
determine the label time of $VSN 000750.

```
# ./sefsysevent_time 1091738029
Thu Aug  5 14:33:49 MDT 2004
```

To change the default polling cycle from once at unload to
once every five minutes use:

```
# samset sef all on 300
# samset
device 80: tapealert on and supported, sef not applicable
device 81: tapealert on and supported, sef on and supported 300s
device 82: tapealert on and supported, sef on and supported 300s
device 90: tapealert on and supported, sef on and supported 300s
```

Or use /etc/opt/SUNWsamfs/defaults.conf to change the default behavior:

```
# cat defaults.conf
sef=all on 300
```

The SEF sysevent macros are available in the
/opt/SUNWsamfs/include/sefvals.h file. The following is a

description of the variables:

Field	Value
Class	Device
Subclass	SEF
Vendor	SUNW
Publisher	SUNWsamfs

SEF sysevent handler macros about SAM-QFS configuration and the SCSI Log Sense Error Counters for pages 2 and 3 and parameters 0-6.

Name	Value and Data Type
VENDOR	Inquiry vendor. Data type is string.
PRODUCT	Inquiry product. Data type is string.
REV	Inquiry revision. Data type is string.
USN	Inquiry unit serial number. Data type is string.
TOD	Time of day. Data type is int32.
EQ_ORD	mcf file Equipment Number. Data type is int16.
NAME	Device name. Data type is string.
VERSION	Inquiry version. Data type is byte.
INQ_TYPE	Inquiry peripheral device type. Data type is byte.
MEDIA_TYPE	SAM-QFS media type. Data type is string.
VSN	Volume serial name. Data type is string.
LABEL_TIME	VSN label timestamp. Data type is integer.
SET	mcf file Family Set. Data type is string.
FSEQ	mcf file Family Set Equipment Number. Data type is int16.
WHERE	SEF location poll=1 or unload=0. Data type is byte.

Write log sense page 2:

Name   Value and Data Type

LP2_PC0	Errors corrected without substantial delay. Data type is uint32.
LP2_PC1	Errors corrected with possible delays. Data type is uint32.
LP2_PC2	Total rewrites. Data type is uint32.
LP2_PC3	Total errors corrected. Data type is uint32.
LP2_PC4	Total times correction algorithm processed. Data type is uint32.
LP2_PC5	Total bytes processed. Data type is uint64.
LP2_PC6	Total uncorrected errors. Data type is uint32.

Read log sense page 3:

Name	Value and Data Type
LP3_PC0	Errors corrected without substantial delay. Data type is uint32.
LP3_PC1	Errors corrected with possible delays. Data type is uint32.
LP3_PC2	Total rereads. Data type is uint32.
LP3_PC3	Total errors corrected. Data type is uint32.
LP3_PC4	Total times correction algorithm processed. Data type is uint32.
LP3_PC5	Total bytes processed. Data type is uint64.
LP3_PC6	Total uncorrected errors. Data type is uint32.

To do simple media analysis, the captured sef data in the /var/tmp/xx.dat file can be formatted for StarOffice spreadsheet analysis and graphing.

SEE ALSO

samset(1M), defaults.conf(4), sefdata(4), sefreport(1M), tapealert(1M).

shrink.cmd(4)

```
NAME
     shrink.cmd - SAM-QFS shrink command file

SYNOPSIS
     /etc/opt/SUNWsamfs/shrink.cmd

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     Directives for controlling the shrink can be read from the
     /etc/opt/SUNWsamfs/shrink.cmd file.  The directives must
     appear one per line.

     Comment lines are permitted.  Comment lines must begin with
     a pound character (#), and the comment can extend through
     the rest of the line.

     Directives that appear prior to any fs= directive are
     applied to all file systems.  Directives that appear after a
     fs= directive are applied to the specified file system only.
     Directives that are specific to a file system override
     general directives.

     The miscellaneous directives control whether a log file is
     written, whether to stage back on-line files that were
     released, other aspects of shrinking.

MISCELLANEOUS DIRECTIVES
     The following miscellaneous directives can be specified in
     the shrink.cmd file:

     block_size =  n
          Sets the buffer size to read the .inodes file in units
          of megabytes.  For n, specify an integer such that 1 <
          n < 16.  The default n=1MB.

     display_all_files
          Writes the name for each file, as it is encountered, to
          the log file.  This directive allows you to see the
          result of executing the remove or release command.  By
          default, the file names are not displayed to the log
          file.

     do_not_execute
          Writes the name for each file, as it is encountered, to
          the log file.  This directive allows you to judge the
          effects of executing the remove or release command,
          without actually executing the command.  By default,
          the command is executed.

     fs = file_system_family_set_name

          Specifies to the shrink that the subsequent directives
          apply to the indicated file_system_family_set_name
          only.
     logfile = filename
          Sets the name of the shrink's log file to filename.  By
          default, no log file is written.

     stage_files
          The files released are staged back on-line.  By
          default, released files are not staged back on-line.
          See stage.

     stage_partial
          The partial size released for files is staged back on-
          line.  By default, the partial size is not staged back
          on-line.  See stage -p.

     streams =  n
          Sets the number of threads to be used to shrink the
          equipment.  For n, specify an integer such that 1 < n <
          128.  The default n=8.

EXAMPLES
     Example 1.  This example file sets the streams directive for
     the samfs1 file system. A shrink log is produced.

               fs = samfs1
               streams = 64
               logfile = /var/adm/shrink.log

     Example 2.  This example specifies stage parameters and log
     files for each file system.

               display_all_files
               logfile = /var/adm/default.shrink.log

               fs = samfs1

               stage_files
               stage_partial
               logfile = /var/adm/samfs1.shrink.log

               fs = samfs2

               stage_partial
               logfile = /var/adm/samfs2.shrink.log

SEE ALSO
     release(1), stage(1).

     mount_samfs(1M), sam-shrink(1M).
```

stager.cmd(4)

```
NAME
     stager.cmd - Defines SAM-QFS stager directives

SYNOPSIS
     /etc/opt/SUNWsamfs/stager.cmd

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     Directives for controlling the SAM-QFS stager are read from
     /etc/opt/SUNWsamfs/stager.cmd.  In the stager.cmd file, each
     directive must appear on its own line.  Each directive has
     the following format:

     keyword = value

     Comment lines can appear in the stager.cmd file.  A pound
     sign (#) in column 1 indicates a comment line.

     The stager.cmd file accepts the following directives:

     directio = on|off
               Set the file reading method for staging. The
               directive on will set direct I/O for all staging
               if file size is equal to or greater than
               dio_min_size. off will cause paged I/O to be used.
               The default is on.

               NOTE: Staging on a shared QFS file system always
               uses direct I/O.

     dio_min_size = n
               If the file size is < n megabytes, the stager will
               use paged I/O for non-shared QFS file systems. The
               default is 8 megabytes.  If directio = off all
               stage io is paged.

               NOTE: dio_min_size is ignored for shared QFS file
               systems which always use direct I/O.

     drives = library count
               Sets the number of drives to use for staging on
               media library library to a number specified by
               count.  The default value is the actual number of
               drives in library.

               The library specified must be the family set name
               of a media library as defined in the mcf file.  If
               this directive is specified, the stager uses only
               count number of drives in the media library to
               stage archive copies.  This directive prevents the

               stager from using all drives in a media library
               and possibly interfering with archiving.
```

```
                          For example, the following directive specifies
                          that 3 drives should be used for staging in an
                          ADIC/Grau media library.

                          drives = gr50 3

               bufsize = media buffer_size [ lock ]
                          Sets the stage buffer size for a specific media
                          type.

                          For media, specify a media type from the mcf(4)
                          man page.

                          For buffer_size, specify an integer value in the
                          range 2 < buffer_size < 8192.  The default is 16.
                          The buffer_size specified is multiplied by the
                          default block size for media.  For more
                          information on default block sizes, see the
                          dev_blksize description on the defaults.conf(4)
                          man page.

                          If lock is specified, the stager locks the stage
                          buffer in memory.  If the stage buffer is locked,
                          system CPU time can be reduced.

               logfile = filename [event]
                          Sets the name of the stager log file to filename,
                          specified as an absolute pathname.  By default, no
                          log file is written.  event is start, finish,
                          cancel, error, or all.  The default is finish,
                          cancel, and error.

                          The stager log file contains a line for each file
                          staged.  The line contains the event type, date,
                          time, media, VSN, inode generation number of the
                          file, position and offset of where the file is
                          stored, name of the file, copy number, user id,
                          group id, requestor's user id, equipment number of
                          the drive upon which the file was staged, and the
                          type of stage, 'V' for data verify and '-' for
                          others.

               maxactive = number
                          Sets the maximum number of stage requests that can
                          be active at one time in the stager to an integer
                          number.  The minimum number is 1.  The default
                          number is based on memory size, 5000 per gigabyte.
                          The maximum number is 500000.

                          The number of outstanding stage requests has a
                          direct impact on incore inode usage, since each
                          request requires an incore inode for the duration
                          of the stage.  Sites may wish to increase the
                          default number of incore inodes if they greatly
                          increase the maximum number of stage requests.
                          This can be done by setting ninodes in the
                          /etc/system file, as shown in the following
                          example.
```

```
set samfs:ninodes=100000
```

For more information on ninodes, see the Sun QFS File System Configuration and Administration Guide.

maxretries = number
: Sets the maximum number of stage retries attempted per archive copy when certain errors are encountered to an integer number. The minimum number is 0. The default number is 3. The maximum number is 20.

copysel = n1:n2:n3:n4
: Sets the copy selection sequence for staging. n? must be a range of 1 <= n? <= 4. By default, 1:2:3:4 is defined, so copy number 1, 2, 3 then 4 is selected for staging if stage is not initiated by stage(1) and copy number is not specified by -c option.

  Four copies, n1 to n4, must be defined, even if there are less than four copies available.

fs = file_system_family_set_name
: Specifies that the subsequent directives apply to the indicated file_system_family_set_name only until stream definition is met. File system specific directives override general directives.

  NOTE: Currently, only copysel definition can be defined for the specific file system.

STREAM DEFINITIONS SECTION

The streams and endstreams directives delimit this section of the stager.cmd file.

Each line begins with the media type followed by the definitions. The syntax for this line is as follows:

media definitions

where:

media   The media type. Currently, only ’dk’ is supported.

definitions

-maxsize size
: Set the maximum size of the stream to size. size may be specified with the suffixes ’b’, ’k’, ’M’, ’G’, and ’T’, for bytes, kilobytes, megabytes, gigabytes, and terabytes. The default unit size is bytes. The default value is 1G bytes. Files to be staged from the same stream will be added to the same stream. If size of the stream hit the size, new stream will be created for the VSN.

-maxcount count

Set the maximum file count to count for the each stream. The default value is 0. Files to be staged from the same VSN will be added to the same stream. If count of the file hit the count, new stream will be created for the VSN.

If more than one of -maxsize or -maxcount are specified, the first condition encountered creates the new stream for VSN.

EXAMPLES

The following is an example /etc/opt/SUNWsamfs/stager.cmd file:

```
logfile = /var/opt/SUNWsamfs/log/stager
drives= hp30 1
copysel = 4:3:2:1
fs = samfs1
copysel = 3:1:4:2
streams
dk -maxsize 2G -maxcount 10000
endstreams
```

The results of the stager's operations are found in /var/opt/SUNWsamfs/log/stager. For the media library specified as hp30, the stager is allowed to use only 1 drive for staging files. The stager selects copy number 4, 3, 2 then 1 for staging files by default. The stager selects copy number 3, 1, 4 then 2 for staging files for the file system samfs1. The size of stream is limited to 2G bytes, and the maiximum file count for the each stream is limited to 10000 for the media type dk.

FILES

The following files are used by the stager:

/etc/opt/SUNWsamfs/stager.cmd Stager command file.

SEE ALSO

sam-stagerd(1M).

defaults.conf(4), mcf(4).

◆ ◆ ◆ CHAPTER 6

Standards, Environment, and Macros (Man Pages Section 5)

This chapter provides section 5 man pages for Sun QFS and Sun Storage Archive Manager.

media(5)

```
NAME
     media - List of media supported by SAM-QFS

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     This man page is obsolete.  All  information  maintained  on
     this  man  page  prior to the Sun QFS or SAM-QFS 4.0 release
     has been moved to the mcf(4) man page.

     This man page will be removed in a future major release.
```

sam_dtrace(5)

```
NAME
     sam_dtrace - SAM-QFS DTrace probes

AVAILABILITY
     SUNWqfs SUNWsamfs

DESCRIPTION
     The SAM-QFS filesystem supports dynamic tracing with  DTrace
     probes.   More  information  on  the  usage of DTrace can be
     found                                                     at
     https://wikis.oracle.com/display/DTrace/Documentation.

     The following is a description of the currently  implemented
     DTrace probes with their arguments.  A quick list of SAM-QFS
     DTrace probes can be found online by typing:
```

```
     dtrace -l -m sdt:samfs

sdt:samfs::sam-open-ret
   The sam-open-ret probe triggers at the end of a  file  open.
   The arguments returned are:
   arg0 - Equipment number << 32 & Inode number
   arg1 - ip->di.status.bits
   arg2 - Open count (after open)
   arg3 - errno (after open)

sdt:samfs::sam-close-ret
   The sam-close-ret probe triggers at the end of a file close.
   The arguments returned are:
   arg0 - Equipment number << 32 & Inode number
   arg1 - ip->di.status.bits
   arg2 - Open count (after close)
   arg3 - last_close_flag << 32 & errno

sdt:samfs::sam-read-ent
   The sam-read-ent probe triggers at the start of a file read.
   The arguments returned are:
   arg0 - Equipment number << 32 & Inode number
   arg1 - uiop->uio_loffset (Offset in file at start of read)
   arg2 - uiop->uio_resid (Amount to read)

sdt:samfs::sam-read-ret
   The sam-read-ret probe triggers at the end of a  file  read.
   More  information is returned here.  You may want to trigger
   only the return call.  The arguments returned are:
   arg0 - Equipment number << 32 & Inode number
   arg1 - uiop->uio_loffset (Offset in file at end of read)
   arg2 - Number of bytes read.
   arg3 - Directio flag << 32 & errno

sdt:samfs::sam-write-ent
   The sam-write-ent probe triggers at  the  start  of  a  file

   write.  The arguments returned are:
   arg0 - Equipment number << 32 & Inode number
   arg1 - uiop->uio_loffset (Offset in file at start of write)
   arg2 - uiop->uio_resid (Amount to write)

sdt:samfs::sam-write-ret
   The sam-write-ret probe triggers at the end of a file write.
   More  information is returned here.  You may want to trigger
   only the return call.  The arguments returned are:
   arg0 - Equipment number << 32 & Inode number
   arg1 - uiop->uio_loffset (Offset in file at end of write)
   arg2 - Number of bytes written.
   arg3 - Directio flag << 32 & errno

sdt:samfs::sam-syscall-ent
   The sam-syscall-ent probe triggers at the entry of a sam-qfs
   syscall.  The arguments returned are:
   arg0 - syscall command (see include/sam/syscall.h)
   arg1 - pointer to syscall argument struct (use copyout to read it)
   arg2 - size of syscall argument

sdt:samfs::sam-syscall-ret
```

The sam-syscall-ent probe triggers at the end of a sam-qfs syscall. The arguments returned are:
arg0 - syscall command (see include/sam/syscall.h)
arg1 - pointer to returned argument (may be NULL, use copyout to read it)
arg2 - errno (after syscall)

sdt:samfs::sam-msgread-client
The sam-msgread-client probe triggers when the shared client receives a message from the metadata server. The arguments returned are:
arg0 - hdr.operation << 32 & hdr.command
arg1 - hdr.hostid << 32 & client ordinal
arg2 - hdr.fsid (file system id from message)
arg3 - hdr.error (errno)

sdt:samfs::sam-msgread-server
The sam-msgread-client probe triggers when the shared metadata server receives a message from the shared client. The arguments returned are:
arg0 - hdr.operation << 32 & hdr.command
arg1 - hdr.hostid << 32 & client ordinal
arg2 - hdr.fsid (file system id from message)
arg3 - hdr.error (errno)

sdt:samfs::sam-lookup-name
The sam-lookup-name probe triggers when the metadata server needs to look up a file name. The arguments returned are:
arg0 - File name component string (use stringof(arg0))
arg1 - Nonzero if case insensitive lookup
arg2 - Internal flags field.

sdt:samfs::sam-find-component
The sam-find-component probe triggers when the metadata server searches a file name component in a directory. The arguments returned are:
arg0 - File name component string (use stringof(arg0))
arg1 - Nonzero if case insensitive lookup
arg2 - File name hash

EXAMPLES

Some examples of usage of the SAM-QFS DTrace probes can be found in /opt/SUNWsamfs/examples/dtrace. These examples are useful in their own right, but also serve as example scripts which can be modified for other purposes. The scripts are:

fs_mon eqid
This script will monitor a specific file system and print out once per second the top 5 bytes/second files being accessed. It will continue until interrupted. The file system is specified by the numeric equipment id of the file system from the mcf file (see man mcf for more details). The output of fs_mon looks like:

```
lake-mn### /opt/SUNWsamfs/examples/dtrace/fs_mon 10
        inode           bytes/sec
             1030         22020096
             1029         20971520
        inode           bytes/sec
             1029         28311552
```

```
                  1030         27874304

  ino_mon eqid inode
     This script will monitor a specific file in a specific  file
     system  and  print  out  once  per second data on read/write
     speed.  It will also gather and quantize I/O size  data  for
     that  file and report a summary of that when the script ends
     (via ^C interrupt).  The file  is  specified  via  equipment
     number  (from  mcf file) and inode number.  Inode number can
     be found from "ls -i" output.   Output  from  ino_mon  looks
     like:

     lake-mn### /opt/SUNWsamfs/examples/dtrace/ino_mon 10 1029
     Inode 1029 opened by pid 15155, tid 1
     Inode 1029,  51261 KB/sec read,      0 KB/sec written
     Inode 1029, 110238 KB/sec read,      0 KB/sec written
     Inode 1029, 110640 KB/sec read,      0 KB/sec written
     Inode 1029, 115298 KB/sec read,      0 KB/sec written
     Inode 1029, 114805 KB/sec read,      0 KB/sec written
     Inode 1029 closed by pid 15155, tid 1, last_close 1
     Inode 1029,  39851 KB/sec read,      0 KB/sec written
     ^C
     I/O size breakdown for eq 10, inode 1029

                value  ------------- Distribution ------------- count
                   -1 |                                         0
                    0 |                                         1
                    1 |                                         0
                    2 |                                         0
                    4 |                                         0
                    8 |                                         0
                   16 |                                         0
                   32 |                                         0
                   64 |                                         0
                  128 |                                         0
                  256 |                                         0
                  512 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 1720320
                 1024 |                                         0
```

sam_worm(5)

```
NAME
     sam_worm - SAM-QFS Write Once Read Many (WORM) features

AVAILABILITY
     SUNWqfs SUNWsamfs

DESCRIPTION
     The Write Once Read Many (WORM) feature is available in  the
     SAM-QFS  filesystem.   The WORM feature allows you to retain
     files for a specified period of time.  This period can be as
     long  as the life of the system or as short as 1 minute, and
     is stored in filesystem metadata.  A SAM-QFS  filesystem  is
     made "WORM-capable" by one of the following mount options:

       o worm_capable
```

  o worm_lite

  o worm_emul

  o emul_lite

These mount options control the actions needed to enable WORM files and directories and the actions that can be performed on these files and directories once they're made WORM-capable. A directory is made WORM capable when the WORM trigger is applied to it. Likewise, a file is made into a WORM file when the WORM trigger is applied to an ordinary file in a WORM capable directory. Note, the file must be fully populated before the trigger is applied as the file's contents can't be modified afterward. SAM-QFS supports two WORM triggers, they are:

  o chmod 4000 filename

  o chmod -w filename

The "chmod 4000" command is the WORM trigger when the "worm_capable" or "worm_lite" mount option is used. Removing the write permissions, that is, the transition from a writable to read-only file is the WORM trigger when the "worm_emul" or "emul_lite" mount option is used.

The WORM trigger and Retention Periods

When the WORM trigger is applied to a file or directory, a period of time, known as a retention period, is associated with it. The retention period is used to indicate how long a file will be protected from change. When a file has an active retention period, it can not be renamed, removed, or have its path altered. When the period expires, a file may only be removed or have its retention period extended.

The retention period can be set or extended by advancing the access time of the associated file or directory. The difference between the current time and the access time is used as the retention period. Setting the retention period on a directory sets the default retention period for files created in the directory. This period is applied to files if a retention period is not specified when the WORM trigger is used. In addition, this period is inherited by all subdirectories created in this directory. Note, a directory's default retention period can be increased or decreased, while a file's period can only be increased.

Default Retention Periods

A default retention period can also be set using the mount option "def_retention=n", where "n" is a simple integer value representing minutes or a variable format MyNdOhPm in which M, N, O, P are arbitrary non-negative integers and the characters y, d, h, m represent the number of years, days, hours, and minutes. The default retention period is applied to a directory if a retention period is not specified when the WORM trigger is applied. If no default period is given, the system default of 30 days is used.

```
  WORM Lite
     The WORM lite option relaxes some  of  the  restrictions  on
     WORM  files.   This  is  enabled  when  the  "worm_lite"  or
     "emul_lite"  mount  options  are  used.   Only  the   system
     administrator is allowed to carry out the following actions:

       o Shorten retention periods on files

       o Remove retained files before their period expires

       o Rebuild "lite" enabled volumes (using sammkfs)

     WORM Lite is a solution for document management  and  reten-
     tion  policies  requiring  data retention guarantees without
     the strict constraints that full WORM implementations  place
     on systems.
```

SUNW.qfs(5)

```
NAME
     SUNW.qfs - Resource type implementation for the Sun QFS
     shared file system in an Oracle Solaris Cluster environment

AVAILABILITY
     SUNWqfs

DESCRIPTION
     The SUNW.qfs resource type implementation supports the Sun
     QFS shared file system installed in an Oracle Solaris Cluster
     environment.  It defines a failover resource for the shared
     file system’s metadata server (MDS).

     The Sun QFS shared file system must be configured to use Sun
     Cluster did devices in the mcf(4) file.  Additionally, the
     Oracle Solaris Cluster node-private hostnames (see scconf(1M)) must be
     used in the shared hosts file (see hosts.fs(4)) for the host
     IP addresses.

  Standard Properties
     See r_properties(5) for a complete description of the
     following resource properties:

     Validate_timeout
     Minimum: 60
     Default: 180

     Boot_timeout
     Minimum: 30
     Default: 300

     Prenet_start_timeout
     Minimum: 60
     Default: 300

     Fini_timeout
```

```
   Minimum: 60
   Default: 120

   Update_timeout
   Minimum: 60
   Default: 120

   Retry_Count
   Maximum: 10
   Default: 2
   Tunable: Anytime

   Retry_Interval
   Maximum: 3600
   Default: 300
   Tunable: Anytime

   Thorough_probe_interval
   Maximum: 3600
   Default: 60
   Tunable: Anytime

   FailOver_Mode
   Default: SOFT
   Tunable: Anytime

   Thorough_Probe_Interval
   Maximum: 3600
   Default: 60
   Tunable: Anytime

Extension Properties
   QFSFileSystem
   Type: string array
   Default: none
   Tunable: Anytime
   This property lists the mount points of the file systems
   under the control of the resource type.  The mount points
   must appear in the /etc/vfstab file.  The associated file
   systems must be samfs file systems.

   Monitor_retry_count
   Default: 4
   Tunable: Anytime
   This property controls the restarts of the fault monitor.
   This property indicates the number of times that the fault
   monitor is restarted by the Process Monitor Facility (PMF)
   and corresponds to the -n option passed to the pmfadm(1M)
   command.  The number of restarts is counted in a specified
   time window (see the property Monitor_retry_interval).  Note
   that this property refers to the restarts of the fault
   monitor itself, not the Sun QFS daemons.

   Monitor_retry_interval
   Default: 2
   Tunable: Anytime
   This property indicates that the failures of the fault
   monitor are counted and corresponds to the -t option passed
   to the pmfadm(1M) command.  If the number of times the fault
```

monitor fails exceeds the extension property
Monitor_retry_count, the fault monitor is not restarted by
the process monitor facility.

Probe_timeout
Default: 120
Minimum: 2
Tunable: Anytime
This property indicates the time out interval (in seconds)
that the probe method is allowed.

Child_mon_level
Default: -1
Tunable: Anytime
This property indicates to the PMF monitor the level of
child process monitoring indicated. -1 disables child
monitoring; Values of zero or larger enable PMF -C
monitoring of the specified level.

EXAMPLES

Example 1. A shared hosts file.

This example shows a shared hosts file for file system
sqfs1, an already existing, unmounted, idle file system, and
replacing its shared hosts file with one suitable for use
with the SUNW.qfs agent.

```
example# samsharefs -R sqfs1
#
# Host file for family set ’sqfs1’
#
# Version: 4    Generation: 317     Count: 3
# Server = host 0/ash, length = 83
#
ash ash-qfe0 1 - server
elm elm-qfe0 2 -
oak oak-qfe0 3 -
example# cat /etc/opt/SUNWsamfs/hosts.sqfs1
#
# Host file for ’sqfs1’
#
ash clusternode1-priv 1 - server
elm clusternode2-priv 2 -
oak clusternode3-priv 3 -
example# samsharefs -uR sqfs1
#
# Host file for family set ’sqfs1’
#
# Version: 4    Generation: 318    Count: 3
# Server = host 0/ash, length = 110
#
ash clusternode1-priv 1 - server
elm clusternode2-priv 2 -
oak clusternode3-priv 3 -
example# samd config
```

Example 2. Instantiating a Failover MDS Resource.

This example assumes that the data service is installed.

This example instantiates a failover Sun QFS MDS resource named qfs-rs in a resource group named qfs-rg. The qfs-rg resource group is assumed to contain at least one logical host name resource, which identifies the logical host names

associated with the resource group.

```
example# scrgadm -a -t SUNW.qfs
example# scrgadm -a -g qfs-rg -j qfs-rs -t SUNW.qfs \
-x QFSFileSystem=/global/qfs1,/global/qfs2
```

The qfs-rg resource group must contain a valid Sun QFS mount point as its QFSFileSystem property.

NOTES

The mount point provided must be the mount point of a Sun QFS shared file system. The file system should be mounted on all resource nodes when the resource group is brought online.

Sun QFS file systems that are not shared must use the HAStoragePlus resource type.

SEE ALSO

scha_resource_get(1HA).

pmfadm(1M), samd(1M), samsharefs(1M), scconf(1M), scrgadm(1M), scswitch(1M).

hosts.fs(4), mcf(4), vfstab(4).

attributes(5), r_properties(5).

Oracle Solaris Cluster 3.1 Data Services Installation and Configuration Guide

CHAPTER 7

Device and Network Interfaces (Man Pages Section 7)

This chapter provides section 7 man pages for Sun QFS and Sun Storage Archive Manager.

acl2640(7)

```
NAME
     acl2640 - The ACL2640 Automated Tape Library

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The ACL2640 tape library supports 264  DLT  tape  cartridges
     and  3  DLT  tape  drives.  The library has a  import/export
     unit that may be used to import or  export  media  into  the
     library.  The  import unit takes one cartridge at a time and
     the export unit will hold up to 12 cartridge.

CONFIGURATION
     The ACL2640 should NOT be configured  with  auto-clean  when
     running Sun QFS and SAM-QFS software.

IMPORT/EXPORT MEDIA
     To import media the  door  on  the  ACL2640  must  first  be
     opened.   To  open  the  door  issue the import(1M) command,
     then follow  the  instructions  in  the  ACL2640  Operator’s
     Guide.

     To export media, use the export(1M) command  to  move  media
     into  the  export  unit  then follow the instructions in the
     ACL2640 Operator’s Guide.

FILES
     mcf                 The configuration file for Sun  QFS  and
                         SAM-QFS software.

SEE ALSO
     export(1M), import(1M), sam-robotsd(1M).

     mcf(4).
```

acl452(7)

```
NAME
     acl452 - The ACL 4/52 Automated Tape Library

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The ACL 4/52 tape library supports 48  DLT  tape  cartridges
     and   4   DLT  tape  drives.   The  library  has  a  4  slot
     import/export unit that may be  used  to  import  or  export
     media  into the library.  These import/export slots may also
     be used as storage slots thus extending the storage capacity
     to 52 slots.

CONFIGURATION
     The ACL 4/52 should NOT be  configured  with  auto-clean  or
     auto-load  when running Sun QFS and SAM-QFS software.  Auto-
     load may be used during initial  loading  of  cartridges  as
     long as the Sun QFS and SAM-QFS software is not running.

IMPORT/EXPORT MEDIA
     To import media, the door on the  ACL  4/52  must  first  be
     opened.   To  open  the  door, issue the export(1M) command,
     then push the OPEN button on the ACL 4/52 front panel.   The
     door  should  open  and  you  may  place media in any of the
     slots.  You may then close the door by  pressing  the  CLOSE
     button  then  manually  closing  the  door when the ACL 4/52
     display indicates that it is ready. The Sun QFS and  SAM-QFS
     software  will  not  recognize  the  new  media  until  the
     import(1M) command is issued.  Anytime you close  the  door,
     you must issue the import function.

     To export media, use the move(1M) command to move media into
     the  import/export  unit  then issue the export(1M) command.
     Push the OPEN button on the ACL 4/52 front panel to open the
     door.

     If the door is already open, you must  close  the  door  and
     issue  the  import  command  before attempting to move media
     into the import/export unit.

     The slot numbers for the import/export unit are 48,  49,  50
     and 51.

     Note: After opening or closing the door, the ACL  4/52  goes
     offline until it has re-initialized.  This will cause delays
     since the library must become online before any commands may
     be issued.

FILES
     mcf                 The configuration file for the  Sun  QFS
                         and SAM-QFS software

SEE ALSO
     export(1M), import(1M), move(1M), sam-robotsd(1M).

     mcf(4).
```

fujitsulmf(7)

```
NAME
     fujitsulmf - The Fujitsu LMF Automated Tape Library

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     fujitsulmf is the Sun QFS and SAM-QFS software interface  to
     the  Fujitsu  LMF  library.  This interface utilizes the LMF
     interface supplied by Fujitsu.  For more information on LMF,
     see  the  LMF  MTL  Server/Client  User’s  Guide supplied by
     Fujitsu.

CONFIGURATION
     It is assumed that the site has the  LMF  server  configured
     and operating with the LMF library.

     The "equipment identifier"  field  in  the  mcf  file,  (see
     mcf(4)),  is the full path name to a parameters file used by
     fujitsulmf.  This file consists of a list of keyword = value
     pairs  or  a  keyword  followed by a drivename = value pair.
     All keywords and  values  are  case-sensitive  and  must  be
     entered as shown.

     lmfdrive
          There is one lmfdrive line for every drive assigned  to
          this  client.   Following  the  lmfdrive  keyword  is a
          drivename = path, where:

          drivename
               is the drivename as configured in LMF.

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

EXAMPLE
     Here are sample parameters files and mcf entries for an  LMF
     library.

          #
          # This is file: /etc/opt/SUNWsamfs/lmf50
          #
          # the name "LIB001DRV000" is from the LMF configuration
          #
          lmfdrive LIB001DRV000 = /dev/rmt/0cbn   # a comment
          #
          # the name "LIB001DRV001" is from the LMF configuration
          #
          lmfdrive LIB001DRV001 = /dev/rmt/1cbn  # a comment

          The mcf file entries.

          #
          # Sample mcf file entries for an LMF library
          #
          /etc/opt/SUNWsamfs/lmf50 50   fj   fj50  -  /var/opt/SUNWsamfs/catalog/fj50_cat
          /dev/rmt/0cbn           51   fd   fj50  -  /dev/samst/c2t5u0
          /dev/rmt/1cbn           52   fd   fj50  -  /dev/samst/c2t6u0
```

IMPORT/EXPORT

Since the physical adding and removing of media in the LMF library is done with LMF utilities, the import/export commands will only affect the library catalog. The import command has an optional parameter (see import(1M)) (-v) for supplying the volser to be added. fujitsulmf will verify that LMF knows about the volser before updating the catalog with the new entry. The export command (see export(1M)) will remove the entry from the catalog.

CATALOG

There are two utilities used to maintain the library catalog used by LMF. build_cat (see build_cat(1M)) is used to build the catalog. dump_cat (see dump_cat(1M)) and build_cat together are used to change the size of the catalog.

To initialize a catalog with 1000 slots run:

```
build_cat /tmp/catalog_file < /dev/null
```

then move /tmp/catalog_file to the path pointed to in the mcf file for this media changer. Use import to populate the catalog with the volumes allowed by DAS. Or, you can create a file with the list of volumes and supply it as input to build_cat (see build_cat(1M)) for the format of the input file.

If the size of the catalog needs to be increased, execute something like:

```
dump_cat file1 | build_cat -s 2000 /tmp/file2
```

This would create a new catalog file (/tmp/file2) with room for 2000 entries and initialize it with the entries from file1. This should only be done when the Sun QFS and SAM-QFS software is not running and sam-amld has been shutdown (see sam-amld(1M)).

FILES

mcf	The configuration file for the Sun QFS and SAM-QFS software.
/opt/SUNWsamfs/lib/liblmf2.so	The LMF library supplied by Fujitsu.

SEE ALSO

build_cat(1M), dump_cat(1M), export(1M), import(1M), sam-robotsd(1M).

mcf(4).

grauaci(7)

NAME

grauaci - The ADIC/Grau Automated Tape Library through the ACI

AVAILABILITY

SUNWsamfs

DESCRIPTION

grauaci is the Sun QFS and SAM-QFS software interface to the ADIC/Grau Network-attached library. This interface utilizes the DAS/ACI 3.10E interface supplied by ADIC. For more information on DAS/ACI, see the DAS/ACI 3.10E Interfacing Guide and the DAS Administration Guide. Both manuals are supplied by ADIC.

CONFIGURATION

Sun assumes that your site has the DAS server configured and operating with the ADIC/Grau library. In the DAS configuration file for this client, the avc (avoid volume contention) and the dismount parameters should both be set to true.

The Equipment Identifier field in the mcf file is the full path name to a parameters file used by grauaci. This file consists of a list of keyword = value pairs or a keyword followed by a drivename = value pair. For more information on the mcf file, see the mcf(4) man page.

All keywords and values, including the following, are case sensitive and must be entered as shown:

Keyword	Value
client	This is the name of this client as defined in the DAS configuration file. This is a required parameter.
server	This is the hostname of the server running the DAS server code. This is a required parameter.
acidrive	There is one acidrive line for every drive assigned to this client. Following the acidrive keyword is a drivename = path, string that is as follows:

drivename	The drive name as configured in the DAS configuration file.
path	The path name to the device. This name must match the Equipment Identifier of an entry in the mcf file.

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

EXAMPLE

The following example shows sample parameters files and mcf entries for a ADIC/Grau library supporting DLT tape and HP optical drives. The catalog files are placed in the default directory, which is /var/opt/SUNWsamfs/catalog.

```
#
# This is file: /etc/opt/SUNWsamfs/gr50
#
client = grau50
server = DAS-server
#
# the name "drive1" is from the DAS configuration file
#
acidrive drive1 = /dev/rmt/0cbn         # a comment
#
# the name "drive2" is from the DAS configuration file
#
acidrive drive2 = /dev/rmt/1cbn         # a comment

#
# This is file: /etc/opt/SUNWsamfs/gr60
#
client = grau60
server = DAS-server
#
# the name "DH03" is from the DAS configuration file
#
acidrive DH03 = /dev/samst/c1t1u0
```

The mcf file entries.

```
#
# Sample mcf file entries for an ADIC/Grau library - DLT
#
/etc/opt/SUNWsamfs/gr50  50      gr      gr50   -  gr50cat
/dev/rmt/0cbn           51      lt      gr50   -  /dev/samst/c2t5u0
/dev/rmt/1cbn           52      lt      gr50   -  /dev/samst/c2t6u0

#
# Sample mcf file entries for an ADIC/Grau library - HP optical
#
/etc/opt/SUNWsamfs/gr60  60      gr      gr60   -  gr60cat
/dev/samst/c1t1u0       61      od      gr60   -
```

IMPORT/EXPORT

The physical adding and removing of cartridges in an ADIC/Grau network-attached library is accomplished using the DAS utilities. The import(1M) and export(1M) commands affect only the library catalog. Therefore, importing and exporting cartridges with the ADIC/Grau network-attached library consists of the following two-step process:

1) Physically import or export the cartridge using the DAS utilities.

2) Virtually update the automated library catalog using the Sun QFS or SAM-QFS import and export utilities.

The import(1M) command has an optional -v parameter for supplying the VSN to be added. The grauaci interface verifies that DAS knows about the VSN before updating the catalog with the new entry. The export(1M) command removes the entry from the catalog. For more information on importing and exporting, see the import and export(1M) man pages.

CATALOG

There are several methods for building a catalog for an ADIC/Grau network-attached library. You should use the method that best suits your system configuration, and this is typically determined by the size of the catalog that is needed.

Method 1: Create a catalog with existing VSN entries. (Please note this method only works for tapes. It does not work for barcoded optical media.) You can build a catalog that contains entries for many tapes by using the build_cat(1M) command. As input to build_cat(1M), you need to create a file that contains the slot number, VSN, barcode, and media type. For example, file input_vsns follows:

```
0 TAPE01  TAPE01    lt
1 TAPE02  TAPE02    lt
2 TAPE03  TAPE03    lt
```

The input_vsns file can be used as input to the build_cat(1M) command, as follows:

```
build_cat input_vsns /var/opt/SUNWsamfs/grau50cat
```

Method 2: Create a null catalog and import VSN entries. You can create an empty catalog and populate it. To create a catalog that will accommodate 1000 slots, use the build_cat command, as follows:

```
build_cat -s 1000 /dev/null /var/opt/SUNWsamfs/catalog/grau50cat
```

Use the import(1M) command to add VSNs to this catalog, as follows:

```
import -v TAPE01 50
```

For ADIC/Grau optical media, it is very important to import the A side of barcoded optical media. The Sun QFS and SAM-QFS software queries the ADIC/Grau database to find the barcode for the B side and fills in the catalog entry for the B side appropriately. The A side of optical media in the ADIC/Grau automated library is the left side of a slot as you face the slots.

Method 3: Use the default catalog and import VSN entries. If a catalog path name is not specified in the mcf file, a default catalog is created in /var/opt/SUNWsamfs/catalog/family_set_name when the Sun QFS or SAM-QFS software is initialized. Following initialization, you must import VSN entries to this catalog. Use the import(1M) command, as follows:

```
          import -v TAPE01 50
```

In the preceding import(1M) command, 50 is the Equipment Identifier of the automated library as specified in the mcf file.

FILES

mcf	The configuration file for the Sun QFS and SAM-QFS software.
/opt/SUNWsamfs/lib/libaci.so	The ACI library supplied by ADIC.

SEE ALSO

build_cat(1M), dump_cat(1M), export(1M), import(1M), sam-robotsd(1M).

mcf(4).

historian(7)

NAME

historian - The Sun QFS and SAM-QFS historian

AVAILABILITY

SUNWsamfs

DESCRIPTION

historian is a catalog that keeps track of volumes that have been exported from an automated library or that have been unloaded from manually loaded devices.

CONFIGURATION

The historian catalog is similar to the catalog for an automated library but since there are no devices associated with it, has no family set name. If there is no historian catalog configured in the mcf file (see mcf(4)) one will be created as:

```
    historian    n+1   hy   -   -   /var/opt/SUNWsamfs/catalog/historian
```

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

The historian catalog will be created with 32 entries when the catalog server initializes and can grow during execution. Each time the catalog fills, 32 entries of approximately 200 bytes each will be added. Make sure the historian's catalog resides on a file system large enough to hold the expected size. Since the catalog is needed before a sam file system can be mounted, DO NOT put the catalog on a Sun QFS or SAM-QFS file system.

Two configuration parameters in the defaults.conf file (see defaults.conf(4)) affect the way the system will react to requests for media or requests to add media to the historian catalog. If exported_media is set to unavailable, then any media exported from a media changer will be set to unavailable in the historian. Any request for media flagged as unavailable will receive an ESRCH error. If attended is set to "no" (operator is NOT available), then any request for media in the historian catalog will be sent back to the file system with an error (ESRCH). Any request for media currently loaded in a manually loaded drive will be accepted no matter what the state of the attended or unavailable flags are.

EFFECTS OF HISTORIAN

Whenever the file system receives the error ESRCH for a stage request, it will automatically generate a stage request for the next archived copy (unless the last stage request was for the last copy). For a removable media request, the error ESRCH will be returned to the user.

IMPORT/EXPORT

import (see import(1M)) is used to insert entries to the historian catalog.

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

CATALOG

The catalog server will create a new, empty catalog in the default file location if none exists or no catalog is specified in the mcf file. Alternately, the build_cat command (see build_cat(1M)) may be used to build the initial catalog.

To initialize a catalog with 32 slots run:

```
build_cat - /tmp/catalog_file < /dev/null
```

then move /tmp/catalog_file to the path pointed to in the mcf file for the historian. Or, you can create a file with the list of volumes and supply it as input to build_cat (see build_cat(1M)) for the format of the input file.

FILES

mcf	The configuration file for the Sun QFS and SAM-QFS software.
defaults.conf	Default information.
/var/opt/SUNWsamfs/catalog/historian	Default historian catalog file.

SEE ALSO

build_cat(1M), dump_cat(1M), export(1M), sam-robotsd(1M).

defaults.conf(4), mcf(4).

ibm3494(7)

NAME

ibm3494 - The IBM3494 interface through lmcpd

AVAILABILITY

SUNWsamfs

DESCRIPTION

ibm3494 is the Sun QFS and SAM-QFS interface to the IBM 3494 library. This interface utilizes the lmcpd interface supplied by IBM. For more information on configuration and interfacing the IBM libraries, see the documentation supplied with the IBM hardware and for lmcpd.

CONFIGURATION

It is assumed that the site has the lmcpd daemon configured and operating with the 3494 library.

The "equipment identifier" field in the mcf file, (see mcf(4)), is the full path name to a parameters file used by ibm3494. This file consists of keyword = value and path_name = value pairs. All keyword/path_name/values are case-sensitive.

The keywords are:

name
: This is the name assigned by the system administrator and configured in the /etc/ibmatl.conf file and the symbolic name of the library. This parameter must be supplied, there is no default.

category
: The category is a hex number between 0x0001 and 0xfeff. Media controlled by Sun QFS or SAM-QFS will have its category set to this value. The default for category is 4.

access
: Access to the library may be shared or private. If private, then any media imported into the library (category = 0xff00) will be added to the catalog and its category will be changed to that specified by category above. If shared, then the import command (see import(1M)) will have to be used to add media to the catalog. The default for access is private.

device_path_name
: There is one device_path_name entry for every drive in the library attached to this machine. This name must match the Equipment Identifier of an entry in the mcf file. Following the device_path_name is the "device number" as described in the IBM documentation. The system administrator can determine this number by running the IBM supplied utility mtlib.

  Following the device number is the shared parameter. This parameter is optional and is used to indicate the drive is shared with other Sun QFS or SAM-QFS servers. See examples below.

EXAMPLE

The example uses the following file and information obtained from the IBM supplied utility mtlib. Both are documented in the materials supplied by IBM.

```
#
# This is file: /etc/ibmatl.conf
# Set this file up according the documentation supplied by IBM.
3493a   198.174.196.50  test1
```

After lmcpd is running, run mtlib to get the device numbers.

```
     mtlib -l 3493a -D
       0, 00145340 003590B1A00
       1, 00145350 003590B1A01
```

Here is a sample parameters file and mcf entries for a IBM 3494 library.

```
     #
     # This is parameters file /etc/opt/SUNWsamfs/ibm60.
     #
     name = 3493a                     # From /etc/ibmatl.conf
     /dev/rmt/1bn = 00145340          # From mtlib output
     /dev/rmt/2bn = 00145350 shared   # From mtlib output
     access=private
     category = 5

     # These are the mcf file entries.
     #
     # IBM 3494 library
     #
     /etc/opt/SUNWsamfs/ibm60 60   im   ibm3494e - ibmcat
     /dev/rmt/1bn             61   tp   ibm3494e
     /dev/rmt/2bn             62   tp   ibm3494e
```

IMPORT/EXPORT

Import media into the library by placing the new media into the I/O slots and closing the door. The library will lock the door and move the media into the storage area. Only 100 volumes can be imported at one time. If you are running with access=private, the library will inform the daemon as the media is moved and the media will be added to the catalog. If running with access=shared, then an import (see import(1M)) command will need to be executed to add the media to the catalog.

Exporting media (in all modes) is performed by the export (see export(1M)) command. This command will move the media to the I/O area and the output mode light on the operator panel will light. The operator can then remove the media from the I/O area.

CATALOG

If running with access=shared then a catalog will need to be built before starting Sun QFS or SAM-QFS. There are two utilities used to maintain the library catalog. build_cat

```
     (see  build_cat(1M)) is used to build the catalog.  dump_cat
     (see dump_cat(1M)) and build_cat together are used to change
     the size of the catalog.

     To initialize a catalog with 1000 slots run:

          build_cat /tmp/catalog_file < /dev/null

     then move /tmp/catalog_file to the path pointed  to  in  the
     mcf    file    for    this    library.    (In   this   case
     /var/opt/SUNWsamfs/catalog/ibmcat).  Use import to  populate
     the  catalog  with  the  volumes.  Or, you can create a file
     with the list of volumes and supply it as input to build_cat
     (see build_cat(1M) for the format of the input file).

     If the size of the catalog needs to  be  increased,  execute
     something like:

          dump_cat file1 | build_cat -s 2000 /tmp/file2

     This would create a new catalog file (/tmp/file2) with  room
     for  2000  entries  and  initialize it with the entries from
     file1.  This can only be done when the Sun  QFS  or  SAM-QFS
     software  is not running and sam-amld has been shutdown (see
     sam-amld(1M)).

FILES
     mcf                            The configuration file for the
                                    Sun QFS and SAM-QFS software.

     /etc/ibmatl.conf               Configuration  file  used   by
                                    lcmpd.

     /opt/SUNWsamfs/lib/libibmlmcp.so
                                    A shared object version of the
                                    runtime  library  supplied  by
                                    IBM

SEE ALSO
     build_cat(1M), dump_cat(1M),  export(1M),  import(1M),  sam-
     robotsd(1M).
    (mcf(4).
```

ibm3584(7)

```
NAME
     ibm3584 - Describes using the IBM 3584 UltraScalable Tape
     Library with Sun QFS or SAM-QFS software

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The IBM 3584 UltraScalable tape library can be used by Sun
     QFS and SAM-QFS software with few modifications to its
     typical operations.  This man page describes the steps you
```

need to take to use this library effectively. The following topics are described:

o Cleaning

o Using the IBM 3584 tape library's partitioning feature

CLEANING

The IBM 3584 UltraScalable Tape Library can be used in a Sun QFS or SAM-QFS environment, but you need to disable automatic cleaning and enable hosted cleaning. No other modifications need to be made to this library's default configuration.

Host cleaning enables the host to detect the need to clean an Ultrium Tape Drive and to control the cleaning process. Host cleaning with a cleaning cartridge is only supported when you disable automatic cleaning and only for the logical library in which each cleaning cartridge is stored. When you enable automatic cleaning, or when the cleaning cartridge is stored in a different logical library, the host application does not have access to the cleaning cartridge.

When automatic cleaning is disabled, the library continues to detect the need to clean a tape drive. When the need is detected, the library displays the physical location of the drive in the following message:

```
          CLEAN [Fx,Rzz]
```

The preceding message is interpreted as follows:

o F represents the frame, and x represents its number

o R represents the row, and xx represents its number

The message clears after you clean the drive by using the supported cleaning method. The cleaning cycle takes less than 2 minutes.

When you enable or disable automatic cleaning, the selected setting is stored in nonvolatile memory and becomes the default during later power-on cycles.

To disable automatic cleaning, perform the following steps:

1. Ensure that a cleaning cartridge is loaded in the library.

2. From the library's activity screen, press MENU. The main menu displays, as follows:

```
   ---------------------------------------------------------------------------
   Main Menu           Panel 0002

   Library Status
   Manual Operations
   Settings
   Usage Statistics
   Vital Product Data
   Service

   [BACK]  [UP]  [DOWN]  [ENTER]
   --------------------------------------------------------------------------

3. Press UP and DOWN to highlight Settings.  Press ENTER.
   The Settings menu displays, as follows:
   --------------------------------------------------------------------------
   Settings             Panel 0100

   Configuration
   Cleaning Mode
   Display/Change SCSI IDs
   Add/Remove Control Paths
   Date/Time
   Sounds

   [BACK]  [UP]  [DOWN]  [ENTER]
   --------------------------------------------------------------------------

4. Press UP or DOWN to highlight Cleaning Mode.  Press
   ENTER.  The Cleaning Mode screen displays and indicates
   whether automatic cleaning is currently enabled or
   disabled.
   --------------------------------------------------------------------------
   Cleaning Mode        Panel 0110

   Automatic Cleaning is ENABLED
   Disable Automatic Cleaning

   [BACK]  [ENTER]
   --------------------------------------------------------------------------

5. The ENTER key acts as a toggle switch for the two
   choices.  Press ENTER until Disable Automatic Cleaning is
   highlighted.  You should receive the following message:

   If you continue you will set the Automatic Cleaning Mode
   to DISABLED.  If you disable automatic cleaning you
   should ensure that each logical library has at least one
   cleaning cartridge since host-initiated cleaning can not
   use a cleaning cartridge located in a different logical
   library.  Do you want to continue?

6. Press YES to disable automatic cleaning.  The Cleaning
   Mode screen redisplays with the new setting.

7. Press BACK until you return to the Activity screen from
   step 1.
```

PARTITIONING

If your IBM 3584 tape library contains 2 or more drives, it can be partitioned into 2 or more logical libraries. If you have partitioned this library, make sure that it is operating as you configured it prior to installing any Sun QFS or SAM-QFS software. For more information on partitioning this library, see your IBM documentation. This subsection describes aspects of using the partitioning feature with the Sun QFS and SAM-QFS software.

When a cartridge is exported (as opposed to being placed in the drawer by a human), only the partition from which it was exported can access that drawer slot. If the cartridge is removed and re-inserted by a human, it is accessable to any/all partitions. The act of removal referred to in this subsection consists of the following steps:

1. Open door.

2. Remove cartridge(s).

3. Close door.

4. Wait for door to lock and then unlock.

5. Open door.

6. Replace cartridge(s).

7. Close door.

NOTES

Much of the text on this man page was derived from the IBM 3584 UltraScalable Tape Library Planning and Operator Guide, IBM publication GA32-0408-01, copyright IBM Corporation 2000.

SEE ALSO

IBM 3584 UltraScalable Tape Library Planning and Operator Guide, IBM publication GA32-0408-01.

http://www.ibm.com/storage/hardsoft/tape/lto/3584

sam-remote(7)

NAME

sam-remote, sam-clientd, sam-serverd - Describes the Sun SAM-Remote interface and daemons

SYNOPSIS

```
/opt/SUNWsamfs/sbin/sam-serverd mshmid pshmid equip

/opt/SUNWsamfs/sbin/sam-clientd mshmid pshmid equip
```

AVAILABILITY

SUNWsamfs

DESCRIPTION

The Sun SAM-Remote client and server software allows automated libraries to be shared among the Solaris systems in a SAM-QFS environment. Sun SAM-Remote allows you to configure multiple storage clients that archive and stage files from a centralized optical and/or tape library. This environment also allows you to make multiple archive copies on various media housed in multiple libraries.

DAEMONS

The Sun SAM-Remote daemons, sam-serverd and sam-clientd, control Sun SAM-Remote. The sam-robotsd daemon starts the sam-serverd and sam-clientd daemons. The identifiers associated with these daemons are as follows:

mshmid	The identifier of the master shared memory segment created by sam-amld.
pshmid	The identifier of the preview shared memory segment created by sam-amld.
equip	The equipment number of the device.

For more information on the sam-robotsd or sam-amld daemons, see the sam-robotsd(1M) or sam-amld(1M) man pages.

CONFIGURATION

Configuring the Sun SAM-Remote client and server software involves adding lines to the mcf file on both the system to be used as the Sun SAM-Remote client and on the system to be used as the Sun SAM-Remote server.

In addition, a client configuration file must be created on the Sun SAM-Remote client, and a server configuration file must be created on the Sun SAM-Remote server.

Each entry in mcf file can configure up to ten clients per server. Use more mcf entries to configure more than ten clients.

In the mcf file, the Equipment Type field contains sc to define a Sun SAM-Remote client or ss to define a Sun SAM-Remote server.

The server configuration file defines the disk buffer characteristics and media to be used for each client. For a client named portland for example:

```
portland
          media
          100 at (000031|000032|000034|000035|000037|000038)
          endmedia
```

The media definitions must be indented with white space or tab characters. The regex data must be enclosed by parentheses.

For a complete description of the Sun SAM-Remote configuration process, see the SAM-QFS Configuration and Administration Guide.

FILES

mcf	The master configuration file for SAM-QFS, Sun QFS, the Sun SAM-Remote client, and the Sun SAM-Remote server.

```
     /opt/SUNWsamfs/lib/librmtsam.so
                         The Sun SAM-Remote shared object
                         library.

SEE ALSO
     sam-amld(1M), sam-robotsd(1M).

     mcf(4).

     SAM-QFS Configuration and Administration Guide.
```

samaio(7)

```
NAME
     samaio - Pseudo Device Driver for AIO

AVAILABILITY
     SUNWqfsr SUNWsamfsr

DESCRIPTION
     The pseudo driver, samaio, allows you to attach a  QFS  file
     to  a  character  device, which can then be accessed through
     that device.  setfa -q attaches a QFS file to  samaio.  When
     you open a file with the -q attribute set, you use the char-
     acter device.  Samaio translates  access  to  the  character
     device  into  I/O on the underlying QFS file. This is useful
     for aio because raw device I/O is faster  than  file  system
     aio.

     Samaio is controlled through /dev/samaioctl -  this  is  the
     only  device  exported during attach, and is minor number 0.
     QFS communicates with samaio through ioctls on this  device.
     When  a  file  is  attached to samaio, character devices are
     exported in /dev/rsamaio. These devices  are  identified  by
     their  minor  number.  Minor  devices are tracked with state
     structures handled with ddi_soft_state(9F).

     The command ls displays a character device  for  files  with
     the  -q  attribute  set,  The command sls displays a regular
     file with its current length for files with the -q attribute
     set,

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  sup-
                    port the requested ioctl function.
     ENXIO          During opening, the device did not exist.

FILES
     /dev/samaioctl      Master control device
     /dev/rsamaio/n      Character device for file n
     /kernel/drv/samaio  32-bit driver
     /kernel/drv/samaio.conf
                         Driver configuration file.  (Should  not
                         be altered.)
     /kernel/drv/sparcv9/samaio
                         64-bit driver
SEE ALSO
     driver.conf(4), devfsadm(1M), setfa(1), sam_setfa(3)
```

samst(7)

```
NAME
     samst - Driver for SCSI media changers and optical drives

SYNOPSIS
     samst@target,lun:a

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     This driver handles embedded SCSI-2 and CCS-compatible  SCSI
     media changers, optical drives, CD-ROM drives and non-motion
     I/O for tape drives

     The type of device is determined using the SCSI inquiry com-
     mand.

     The only I/O supported for optical devices is "raw".   samst
     supports  512-, 1024-, 2048-, and 4096-byte sector sizes for
     optical media.  The names of the  raw  files  are  found  in
     /dev/samst.

  Special handling during open
     If O_NDELAY or O_NONBLOCK is specified on the open, then the
     device  does  not have to be in the ready state for the open
     to succeed.  This allows the opening of a  device  for  ini-
     tialization 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/cntnun   raw files

where:
     cn   controller n
     tn   SCSI target id n (0-6)
     un   SCSI LUN n (0-7)
```

SEE ALSO

samdev(1M).

driver.conf(4).

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  nor-
          mally.

     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 a nonzero value.  You can do this using  the
     set command in the /etc/system file (see system(4)).

     Whenever a new version of Sun QFS or SAM-QFS  is  installed,
     the  existing samst.conf file is copied to samst.conf.MMDDYY
     for reference and backup purposes.
```

sony(7)

```
NAME
     sony - Attaches a Sony network-attached tape library through
     the DZC-8000S interface

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     The SAM-QFS software package contains the Sun QFS and SAM-
     QFS interface to a Sony network-attached library.  This
     interface uses the DZC-8000S 3.01 interface supplied by
     Sony.  For more information on DZC-8000S, see the Sony
     PetaSite Application Interface DZC-8000S manual.  This
     manual is supplied by Sony.

CONFIGURATION
     It is assumed that the site has the PetaSite Controller
     (PSC) configured and operating with the Sony library.  In
     the Execute Mode of the PSC configuration, the following
     must be set to on:

     o  Thread With Load

     o  Unthread with Fast Unload

     o  Unthread with Eject

     o  Wait for Drive Use

     The Equipment Identifier field in the Sun QFS or SAM-QFS mcf
     file must be the full path name to a Sony parameters file.
     For more information on specifying a parameters file, see
     the mcf(4) man page.
     The parameters file consists of a list of keyword = value
     pairs.  All keyword and value specifications are
     case-sensitive and must be entered as shown on this man
     page.  The keyword and value specifications are as follows:

     userid = userid
               Identifies the user during initialization of the
               Sony library functions.  The userid values can be
               specified in hexadecimal or decimal.  The valid
               range is from 0 to PSCUSERIDMAX(0xfff), which is 0
               <= userid <= 65535 (decimal) or 0 <= userid <=
               0xffff (hexadecimal).  This is a required
               parameter.

     server = serverid
               Specifies the host name of the server running the
               PSC server code.  This is a required parameter.

     sonydrive binnum = path [ shared ]
               Specifies characteristics of the tape drive.
               There must be one sonydrive line for every drive
               assigned to Sun QFS or SAM-QFS in the mcf file.
               This name must match the Equipment Identifier of
               an entry in the mcf file.

               The following arguments follow the sonydrive
               keyword:

               binnum    Specifies the bin number assigned to the
                         drive in the PSC configuration. The bin
                         number can be identified using the PSC
                         Monitoring and Maintenance terminal.
                         This is a required argument.

               path      Specifies the Solaris /dev/rmt/ path
                         name to the device.  The path must match
                         the Equipment Identifier of an entry in
                         the mcf file.  This is a required
                         argument.

               shared    Specifies that this drive is shared with
                         other processes.  For example, this
                         drive can be shared between multiple Sun
                         QFS or SAM-QFS servers.  This is an
                         optional argument.
```

EXAMPLE

```
     The following example shows the configuration files for a
     network-attached Sony library with Sony DTF tapes.

     Here are the sample entries in the mcf file. The catalog
     file is placed in the default directory, which is
     /var/opt/SUNWsamfs/catalog.

     The mcf file is as follows:

     #
     # This is the file: /etc/opt/SUNWsamfs/mcf
     # This file shows sample mcf entries for a Sony network-attached
```

```
# robot with Sony DTF tapes.
#
/etc/opt/SUNWsamfs/sonyfile 50 pe sony50 on /var/opt/SUNWsamfs/sony50cat
/dev/rmt/0cbn              51 so sony50 on
/dev/rmt/1cbn              52 so sony50 on
```

The parameters file for a Sony library supporting Sony DTF tapes is as follows:

```
#
# This is file: /etc/opt/SUNWsamfs/sonyfile
#
# The userid identifies the user during initialization of
# the PetaSite library functions. Valid IDs are 0 to
# PSCUSERIDMAX(0xfff).
#
userid = 65533
#
# The server identifies the hostname for the server running
# the DZC-8000S server code.
#
server = europa
#
# The sonydrive bin number 1001 is from the PSC configuration file
#
sonydrive 1001 = /dev/rmt/0cbn shared  # a comment
#
# The sonydrive bin number 1002 is from the PSC configuration file
#
sonydrive 1002 = /dev/rmt/1cbn         # a comment
```

IMPORT/EXPORT

The physical adding and removing of cartridges in a Sony network-attached library is accomplished using the PSC utilities. The import(1M) and export(1M) commands affect only the library catalog. Therefore, importing and exporting cartridges with the Sony network-attached library proceeds according to the following two-step process:

1. Physically import or export the cartridge using the PSC software.

2. Virtually update the library catalog using the Sun QFS or SAM-QFS import/export utilities.

The import(1M) command has an optional -v option that allows you to specify the VSN to be added. The samsony package verifies that PSC knows about the VSN before updating the catalog with the new entry. The export(1M) command removes the entry from the catalog.

CATALOG

There are several methods for building a catalog for a Sony network-attached library. You should use the method that best suits your system configuration, typically depending on the size of the catalog that is needed.

Method 1: Create a catalog with existing VSN entries. You can build a catalog that contains entries for many tapes by using the build_cat(1M) command. As input to the build_cat(1M) command, you need to create a file that contains the slot number, VSN, bar code label, and media type. For example, the file input_vsns follows:

```
0  "SEG001"  "SEG001"  so
1  "SEG002"  "SEG002"  so
2  TEST1     TEST1     so
3  TEST2     TEST2     so
```

The input_vsns file can be used as input to the build_cat(1M) command as follows:

```
build_cat input_vsns /var/opt/SUNWsamfs/sony50cat
```

Method 2: Create a null catalog and import VSN entries. You can create an empty catalog and populate it. To create a catalog that will accommodate 1000 slots, use the build_cat(1M) command as follows:

```
build_cat -s 1000 /dev/null /var/opt/SUNWsamfs/catalog/sony50cat
```

Use the import(1M) command to add VSNs to this catalog, as follows:

```
import -v "SEG005" 50
```

Method 3: Use the default catalog and import VSN entries. If a catalog path name is not specified in the mcf file, a default catalog is created in /var/opt/SUNWsamfs/catalog/family_set_name when Sun QFS or SAM-QFS is initialized. Following initialization, you must import VSN entries to this catalog by using the import command as follows:

```
import -v "SEG005" 50
```

In the previous import(1M) command, 50 is the Equipment number of the library as specified in the mcf file.

FILES

mcf	The configuration file for the Sun QFS and SAM-QFS software.
/opt/SUNWsamfs/lib/libpsc.so	The PSC library supplied by Sony.
/opt/SUNWsamfs/sbin/sony_helper	A program to issue commands to the Sony PSC.

SEE ALSO

build_cat(1M), dump_cat(1M), export(1M), import(1M), sam-robotsd(1M).

mcf(4).

ssi.sh(7)

```
NAME
     ssi.sh - The configuration file  for  the  StorageTek  (STK)
     Client System Interface CSI.

AVAILABILITY
     SUNWsamfs

DESCRIPTION
     ssi.sh is a script that allows users to  select  values  for
     several  dynamic environment variables used by the CSI.  The
     STK API code defines default values for these  variables  if
     they  are  not defined dynamically. To allow the most flexi-
     bility  in  setting  these   variables,   a   shell   script
     /etc/opt/SUNWsamfs/scripts/ssi.sh,   is  used  by  the  sam-
     stkd(1M) daemon, to start the ssi_so. In general, most sites
     do not need to change the variables within this script.

CONFIGURATION
     An     example     script     can      be      found      in
     /opt/SUNWsamfs/examples/ssi.sh.   This  script,  or  a  user
     created      script,      must      be      copied       to
     /etc/opt/SUNWsamfs/scripts/ssi.sh

     It is assumed that the site has the server daemons (CSI  and
     ACSLM) configured and operating with the STK library.

     The following environment variables are defined in the shell
     script supplied in /opt/SUNWsamfs/examples/ssi.sh:

     CSI_TCP_RPCSERVICE
             This variable is used  to  define  whether  the  CSI
             operates  as  a  TCP  RPC Server. This environmental
             variable must be set to TRUE for the firewall-secure
             CSC.  The firewall-secure ACSLS applications packets
             are all sent using the TCP network transport.

     CSI_UDP_RPCSERVICE
             This variable is used  to  define  whether  the  CSI
             operates  as  a  UDP  RPC server. This environmental
             variable must be set  to  FALSE  for  the  firewall-
             secure  CSC.  The firewall-secure ACSLS applications
             packets are all sent using  the  TCP  network  tran-
             sport.

             The CSI can operate as a TCP and a UDP server simul-
             taneously.

     CSI_CONNECT_AGETIME
             This defines the value of the maximum age of pending
             requests  in  the CSI’s request queue. This variable
             is accessed  as  a  "C"  character  array  datatype,
             expressed  as  an integer number of seconds. A value
             of 172800 indicates two days.

             Messages older than this value are removed from  the
             queue  and  the  CSI sends an entry to the Event Log
```

```
                when this happens.

         CSI_RETRY_TIMEOUT
                This defines the minimum amount of time, in seconds,
                that the CSI will wait between attempts to establish
                a network connection.

         CSI_RETRY_TRIES
                This variable defines the number of attempts the CSI
                will  make  to  transmit a message. Pending messages
                are discarded if a connection cannot be  established
                within the defined number of tries.

                The CSI_RETRY_TIMEOUT and  CSI_RETRY_TRIES  together
                determine  the  minimum  total  time  the  CSI  will
                attempt to send a message.

     SEE ALSO
          sam-robotsd(1M).

          stk(7), ssi_so(7).
```

ssi_so(7)

```
     NAME
          ssi_so - The StorageTek (STK) ACSAPI client daemon.

     AVAILABILITY
          SUNWsamfs

     DESCRIPTION
          ssi_so is a shared object version of the SSI daemon supplied
          by  STK.   This daemon is the interface that 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 environ-
          ment variables.  To allow the most  flexibility  in  setting
          these         variables,        a        shell         script
          (/etc/opt/SUNWsamfs/scripts/ssi.sh) is used by the stk  dae-
          mon  (see  sam-stkd(1M)), to start a ssi_so daemon.  In gen-
          eral, most sites should not need  to  change  the  variables
          within this script.

     SEE ALSO
          sam-robotsd(1M).

          stk(7).
```

stk(7)

```
NAME
     stk - The StorageTek interface through ACSAPI

AVAILABILITY
     SUNWsamfs

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

CONFIGURATION
     It is assumed that the site has the server daemons (CSI  and
     ACSLM) configured and operating with the StorageTek library.

     The  Equipment  Identifier  field  in  the  mcf  file,  (see
     mcf(4)),  is the full path name to a parameters file used by
     stk.  This file consists of keyword = value and path_name  =
     value  pairs.  All keyword, path_name, and value entries are
     case-sensitive.

     The keywords are:

     access  This is the user_id used by this client  for  access
             control.   If  this  parameter  is not supplied, the
             access control string will  be  a  null  string  (no
             user_id).

     hostname
             This is the hostname for the server that is  running
             ACSLS.  If the hostname is not supplied, the default
             will be localhost.  All sites should set this value.

     ssihost This is the name used for the SAM-QFS server when  a
             multihomed  SAM-QFS  server  is  used.   The ssihost
             would be the name of the SAM-QFS server on  the  lan
             connecting  to  the  ACSLS host.  Only sites where a
             multihomed SAM-QFS server is used need to supply  an
             ssihost value. The default will be localhost.

     portnum This is the portnum for SSI services on  the  server
             that  is  running  ACSLS.  If the port number is not
             supplied, the default is 50004.  Please note that if
             you  are  running co-hosted ACSLS 5.3 or higher, the
             default value does  not  work  (try  a  higher  port
             number,  like  50014).   If you are running multiple
             connections to ACSLS servers, then the  port  number
             for  each  stk configuration file needs to be unique
             (for example, 50014  in  one,  50015  in  the  next,
             etc.).

     ssi_inet_port
```

This is the fixed port number for incoming responses and specifies the port the SSI will use for incoming ACSLS responses in a firewall environment. Valid values are 1024 - 65535, and 0. Setting this environmental variable to a non-zero value makes the SSI use this port for incoming ACSLS responses. This means that the firewall needs to allow incoming requests on that port in order for the ACSLS responses to be received by the SSI. Setting this value to zero or leaving it unset indicates that the previous behavior of allowing the port to be dynamically allocated will remain in effect.

csi_hostport
: This firewall environmental variable specifies the port to which the SSI will send its ACSLS requests on the ACSLS server. Setting this variable eliminate queries to the portmapper on the ACSLS server and instead, sends requests to this port on the ACSLS server. Valid values are 1024 - 65535, and 0. Setting this variable to zero or leaving it unset indicates that the previous behavior of querying the portmapper on the ACSLS server will continue to be used.

capid
: This specifies the CAP (Cartridge Access Port) to be used for exporting of volumes when the -f option is used with export command. Following the capid is the description of this CAP in terms of the StorageTek library. This description starts with an open parenthesis followed by 3 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 CAP as configured in the StorageTek library.

  lsm is the LSM number for this CAP as configured in the StorageTek library.

  cap is the CAP number for this CAP as configured in the StorageTek library.

capacity
: This is used to set the capacity of the media supported by the StorageTek. The parameter to capacity is a comma separated list of index = value pairs enclosed in parentheses. index is the index into the media_type file (supplied by StorageTek and located on the ACS system) and value is the capacity of that media type in units of 1024 bytes. You should only need to supply this entry if the ACS is not returning the correct media type or new media types have been added. Sun QFS and SAM-QFS have defaults for index values that were current at the time of release. Generally, it is necessary to supply an index only for new cartridge types. For the capacity of each cartridge type, see the SAM-QFS Storage and Archive Management Guide.

```
     device_path_name
          There is one device_path_name  entry  for  every  drive
          attached  to  this client.  The device_path_name is the
          path to the device on the client. This name must  match
          the  Equipment  Identifier of an entry in the mcf file.
          Following the device_path_name is  the  description  of
          this  drive  in  terms of the StorageTek library.  This
          description starts with an open parenthesis followed by
          4   keyword   =   value   pairs  followed  by  a  close
          parenthesis.  The keyword =  value  pairs  between  the
          parentheses  may  be  separated by a comma (,), a colon
          (:) or by white space. Following the close  parenthesis
          is  an  optional  keyword  used  by Sun QFS and SAM-QFS
          software to designate when a drive is shared with other
          Sun  QFS  and SAM-QFS servers.  The keyword identifiers
          and their meanings are as follows:

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

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

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

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

          shared
               The shared keyword follows the close  parenthesis.
               This  keyword  is optional and is used to indicate
               the drive is shared with other Sun QFS and SAM-QFS
               servers.

EXAMPLE
     Here is a sample parameters  file  and  mcf  entries  for  a
     StorageTek library:
          #
          # This is file: /etc/opt/SUNWsamfs/stk50
          #
          hostname = acsls_server_name
          portnum = 50004
          ssi_inet_port = 0
          csi_hostport = 0
          access = some_user  # No white space allowed in the user_id field
          capid = (acs=0, lsm=1, cap=0)
          /dev/rmt/0cbn = (acs=0, lsm=1, panel=0, drive=1)         #a comment
          /dev/rmt/1cbn = (acs=0, lsm=1, panel=0, drive=2) shared  #a comment
          capacity = (0=215040, 1=819200, 5=10485760)

     The mcf file entries that reference this configuration  file
     are:

          #
          # Sample mcf file entries for a StorageTek library
          #
          /etc/opt/SUNWsamfs/stk50      50  sk  sk50   - /var/opt/SUNWsamfs/catalog/sk50
          /dev/rmt/0cbn                 51  st  sk50   -
          /dev/rmt/1cbn                 52  st  sk50   -
```

IMPORT/EXPORT

Since the physical adding and removing of cartridges in the StorageTek library is done with ACSLM utilities, the import/export commands and GUI buttons will only affect the library catalog. The import command has optional parameters for supplying a single volume to be added or to add a number of volumes from a pool (see import(1M)). export (see export(1M)) will remove an entry from the catalog.

CATALOG

The Sun QFS and SAM-QFS systems automatically build a library catalog for a StorageTek automated library. However, you must populate the library catalog. For information on populating the library catalog, see the SAM-QFS Storage and Archive Management Guide.

FILES

mcf	The configuration file for the Sun QFS and SAM-QFS software.
/etc/opt/SUNWsamfs/scripts/ssi.sh	A shell script used to start ssi_so.
/opt/SUNWsamfs/sbin/ssi_so	A shared object version of the SSI daemon supplied by StorageTek.
/opt/SUNWsamfs/lib/stk/*	The libraries needed by the API interface supplied by StorageTek.
/opt/SUNWsamfs/sbin/stk_helper	A program to issue commands for the StorageTek ACSAPI

SEE ALSO

build_cat(1M), dump_cat(1M), export(1M), import(1M), sam-robotsd(1M).

mcf(4).

ssi_so(7).

SAM-QFS Configuration and Administration Guide.