Oracle® Hierarchical Storage Manager and StorageTek QFS Software

Maintenance and Administration Guide

Release 6.1.4

E42064-08

July 2019

ORACLE

Oracle Hierarchical Storage Manager and StorageTek QFS Software Maintenance and Administration Guide, Release 6.1.4

E42064-08

Contents

4 Managing Files and Directories

5 Managing Libraries, Media, and Drives

Preface

This document addresses the needs of system administrators, storage and network administrators, and service engineers who are tasked with monitoring, managing, and maintaining file systems and archiving solutions using Oracle Hierarchical Storage Manager (formerly StorageTek Storage Archive Manager) and Oracle StorageTek QFS Software.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support 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.

Prerequisites for Using this Document

This document assumes that you are already familiar with the administration of Oracle Solaris operating systems, disk and tape storage systems, and both local and storage area networks. Please refer to the Solaris documentation and man pages and to storage hardware documentation for information on relevant tasks, commands, and procedures.

Conventions

The following textual conventions are used in this document:

- *Italic* type represents book titles and emphasis.
- **Monospace** type represents commands and text displayed in a terminal window and the contents of configuration files, shell scripts, and source code files.
- **Monospace bold** type represents user inputs and significant changes to commandline output, terminal displays, or file contents. It may also be used to emphasize particularly relevant parts of a file or display.
- ***Monospace bold oblique*** type represents variable inputs and outputs in a terminal display or file.

- *Monospace oblique* type represents other variables in a terminal display or file.
- **...** (three-dot ellipsis marks) represent file contents or command output that is not relevant to the example and has thus been omitted for brevity or clarity.
- **[-]** (brackets surrounding values separated by a hyphen) delimit value ranges.
- **[ ]** (brackets) in command syntax descriptions indicate optional parameters.
- **root@***solaris-host***:~#** represents a Solaris command shell prompt.
- **[root@***linux-host* **~]#** represents Linux command shell prompts.

Available Documentation

The *Oracle Hierarchical Storage Manager and StorageTek QFS Software Maintenance and Administration Guide* is part of the multivolume *Oracle HSM Customer Documentation Library*, available from **http://docs.oracle.com/en/storage/#sw**.

Oracle Solaris operating system documentation is available at **http://docs.oracle.com/en/operating-systems/**.

For information on system requirements, new features, and bug fixes, consult the release notes, **README.txt**, in the download ZIP file or on your file-system server at **/opt/SUNWsamfs/doc/README.txt**.

1

Maintaining Oracle HSM Solutions

Throughout the planning and deployment process, we have stressed that QFS file systems and Oracle Hierarchical Storage Manager software are designed to hide the complexities of performance optimization, data protection, and archiving behind a simple, UNIX file-system interface. Users, applications, and, for the most part, administrators should be able to treat a fully optimized, Oracle HSM archiving system implemented on a mix of disk arrays and tape libraries as if it were an ordinary UFS file system on a single, local disk. Once installed and configured, Oracle HSM software should automatically manage your data and storage resources in the most efficient and reliable way possible, with minimal human intervention.

That said, as with any UNIX file system, there are still monitoring and periodic management tasks that need to be carried out. This book outlines these activities.

2

Monitoring File System Operations

Correctly configured Oracle HSM file systems need little administrative intervention, however you will need to monitor each system for abnormalities. In general, you monitor two things:

- Availability — The system can lose core functionality when key components such as a host system, network interface, file system, or storage subsystem become unavailable. The administrative interfaces and logs display availability alerts.
- Utilization — Even when the system is functioning normally, excessive usage can strain resources and hinder the archiving process. Monitor usage trends and rates to prevent usage issues.

Oracle HSM provides the following monitoring interfaces:

- Oracle HSM Manager
- samu
- Log and Trace Files

Oracle HSM Manager

Oracle HSM Manager is a browser-based GUI that lets administrators monitor and control all aspects of file-system operations. The GUI is available in for releases 6.1.3 and below. Refer to the help system for more information on the GUI.

samu

The **samu** operator utility is a text-based, menu-driven configuration and management interface that you launch from the command line. It helps monitor Oracle HSM devices, file system activity, and error messages.

The **samu** utility is in some respects similar to the UNIX **vi** editor. You select displays, set display options, navigate within and between displays, enter commands, refresh displays, and quit the utility using similar control key sequences. The last line of each display window displays error messages. Displays refresh automatically unless an error occurs, in which case the display halts until the operator takes further action. When desired, you can take snapshots of display windows for later reference.

The **h** command opens help screens that list all of the keyboard short cuts, commands. and parameters. You can also consult the **samu** man page and the *Oracle Hierarchical Storage Manager and StorageTek QFS* `samu` *Command Reference* in the *Oracle HSM Customer Documentation Library* (**http://docs.oracle.com/en/storage/#sw**) for additional information.

The following is a typical **samu** monitoring display:

```
Archiver status                  samu      5.4     12:24:10 Mar 19 2014

sam-archiverd: Waiting for resources

sam-arfind:  samma1 mounted at /samma1
Files waiting to start 0    schedule 70,524    archiving 0
Monitoring file system activity.

sam-arfind:  DISKVOL1 mounted at /diskvols/DISKVOL1
Files waiting to start 0    schedule 0    archiving 0
Monitoring file system activity.

                          samu on samqfshost1
```

Log and Trace Files

The Oracle HSM software performs comprehensive logging and, when configured, tracing. So you may wish to monitor the following files, particularly when problems arise:

- **/var/adm/messages**
- **/var/adm/sam-log**
- **/var/opt/SUNWsamfs/trace/** (holds trace files for the daemons and processes)
- **/var/opt/SUNWsamfs/devlog/** (holds logs for the devices configured in the **/etc/opt/SUNWsamfs/mcf** file)
- **/var/opt/SUNWsamfs/archiver.log**
- **/var/opt/SUNWsamfs/stager.log**
- **/var/opt/SUNWsamfs/recycler.log**
- additional archiving logs specific to file systems (if configured).

For information on configuring logging and tracing, see the *Oracle Hierarchical Storage Manager and StorageTek QFS Installation and Configuration Guide* in the *Oracle HSM Customer Documentation Library* (**http://docs.oracle.com/en/storage/#sw**).

3

Managing Oracle HSM File Systems

This chapter covers file-system maintenance and reconfiguration tasks. The first section, Managing Oracle HSM File Systems, addresses maintenance of all Oracle HSM file systems, archiving and non-archiving, shared and unshared (standalone). The second section, Managing Oracle HSM Shared File Systems, deals with special considerations that affect shared file systems.

Managing Oracle HSM File Systems

This section outlines the following tasks:

- administering file system quotas
- controlling archiving and staging operations
- renaming file systems
- repairing file systems
- adding devices to file systems
- removing data devices from a file system.

Administering File System Quotas

Set file system quotas to control the online and total storage space that a given user or collection of users can consume within the file system. You can set quotas by user ID, by group ID, or by an administrator-defined *admin set ID* that groups users by common characteristics, such as participation in a particular project. The admin set ID is especially useful when a project includes users from several groups and spans multiple directories and files.

You enable quotas by mounting a file system with the **`quota`** mount option (set by default) and disable quotas by mounting it with the **`noquota`** mount option. You define the quotas by placing one or more *quota files* in the file system root directory: **`.quota_u`**, **`.quota_g`**, and **`.quota_a`**, which set quotas for users, groups, and admin sets, respectively. The first record in each file, record **`0`**, sets the default values. Subsequent records set values specific to particular users, groups, or admin sets.

Quotas allocate usable file-system space, not simply storage space. They thus set upper limits on both the number of 512-byte blocks allocated on the media and the number of inodes allocated in the file system. The block count measures the storage space per se. The inode count measures the resources available for accessing that storage. A single file that used a great many blocks of storage space but only one inode might thus take up the same amount of file-system space as a great many empty, zero-length files that take up many inodes and no blocks.

Each quota can include both *soft* and *hard limits*. A hard limit defines the maximum file-system resources that all of a given owner's files can use temporarily. A soft limit defines the maximum file-system resources that the owner's files can use indefinitely. Resource usage can grow to amounts that lie between the soft and hard limits only for brief intervals, as defined by the *grace period* in the quota.

This section describes the following administrative tasks:

- characterizing the storage requirements of users, groups, and organizational units
- creating admin sets for projects and for directories used by multiple groups
- configuring a new file system to use quotas
- configuring quotas for an existing file system
- setting quotas for groups, projects, directories, and users
- repairing inconsistent quotas
- checking quotas
- temporarily extending or cancelling grace periods
- stopping new resource allocations
- removing quotas from a file system.

Characterize the Storage Requirements of Users, Groups, and Organizational Units

To set sustainable quotas, you have to set limits that accommodate user requirements in a way that is both manageable and scalable. So, before setting quotas, estimate the storage requirements of your users. To keep the process manageable, start by classifying user requirements as broadly as possible, so that you address the greatest number of requirements with the smallest amount of administrative effort. You can then specially assess a small number of user requirements that do not fit in to the broader categories. The results will provide the broad outlines of the quotas and types of limits that you will set.

The approach outlined below starts by identifying the file-system requirements of access-control groups, since most organizations already define these groups. Then it defines special sets of users whose needs do not align well with those of the standard groups. Then and only then does it begin to address any requirements that are unique to individual users. Proceed as follows:

1. Since your existing access-control groups already gather together users with similar resource requirements, start by defining the average storage requirements of any groups that will use the file system. Estimate both the *average amount of storage space* used (in 512-kilobyte blocks) and the *average number of files* stored, which is equivalent to the average number of inodes used.

   Since group members typically have similar organizational roles and work responsibilities, they frequently need access to the same directories and files and generally make similar demands on storage. In the example, we identify three groups that will use the file system **`/hsm/hqfs1`**: **`dev`** (Product Development), **`cit`** (Corporate Information Technologies), and **`pgmt`** (Program Management). We list the groups, the number of members in each, and their average individual and group requirements in a simple spreadsheet:

Group	Users	Average Blocks Per User	Average Files Per User	Average Blocks/Group	Average Files/Group
dev	30	67108864	500	2013265920	15000
cit	15	10485760	50	157286400	750
pmgt	6	20971520	200	125829120	1200
Total Blocks/ Files (Average)					

2. Next, carry out the same calculations for the *maximum amount of storage space* and the *maximum number of files* that group members will store at any given time. Record the results.

   In the example, we record the results in a new spreadsheet:

Group	Users	Maximum Blocks Per User	Maximum Files Per User	Maximum Blocks/Group	Maximum Files/Group
dev	30	100663296	1000	3019898880	30000
cit	15	15728640	100	235929600	1500
pmgt	6	31457280	400	188743680	2400
Total Blocks/ Files (Maximum)					

3. Now identify any sets of users that belong to different groups but share distinct storage requirements that cannot be not addressed on the basis of group membership. Make the same estimates and carry out the same calculations for each identified organization as you did for each access-control group.

   In the example, we identify two company projects that will require storage allocations, code named **portal** and **lockbox**. Members of the engineering, marketing, compliance, test, and documentation groups will work together on these projects and will use the same directories and many of the same files. So we add them to our requirements spreadsheets:

Group	Users	Average Blocks Per User	Average Files Per User	Average Blocks/Group	Average Files/Group
dev	30	67108864	500	2013265920	15000
cit	15	10485760	50	157286400	750
pmgt	6	20971520	200	125829120	1200
portal	10	31457280	400	314572800	4000
lockbox	12	31457280	500	377487360	6000
Total Blocks/ Files (Average)					

Group	Users	Maximum Blocks Per User	Maximum Files Per User	Maximum Blocks/Group	Maximum Files/Group
dev	30	100663296	1000	3019898880	30000
cit	15	15728640	100	235929600	1500
pmgt	6	31457280	400	188743680	2400
portal	10	37748736	700	377487360	7000

Group	Users	Maximum Blocks Per User	Maximum Files Per User	Maximum Blocks/Group	Maximum Files/Group
lockbox	12	45613056	600	547356672	7200

Total Blocks/ Files (Maximum)

4. Now identify any individual users whose requirements have not yet been addressed. Make the same estimates and carry out the same calculations for each user as you did for each access-control group and non-group organization.

   Where possible, address user requirements collectively, so that polices are uniform and management overhead is at a minimum. However, when individual requirements are unique, you need to address them individually. In the example, we identify **jr23547** in the **pgmt** group as a user whose special responsibilities require special storage allocations. So we add him to our requirements spreadsheets:

Group	Users Per Set	Average Blocks Per User	Average Files Per User	Average Blocks	Average Files
dev	30	67108864	500	2013265920	15000
cit	15	10485760	50	157286400	750
pmgt	6	20971520	200	125829120	1200
portal	10	31457280	400	314572800	4000
lockbox	12	31457280	500	377487360	6000
jr23547	1	10485760	600	10485760	600

Total Blocks/ Files (Average)

Group	Users	Maximum Blocks Per User	Maximum Files Per User	Maximum Blocks/Group	Maximum Files/Group
dev	30	100663296	1000	3019898880	30000
cit	15	15728640	100	235929600	1500
pmgt	6	31457280	400	188743680	2400
portal	10	37748736	700	377487360	7000
lockbox	12	45613056	600	547356672	7200
jr23547	1	100663296	2000	100663296	2000

Total Blocks/ Files (Maximum)

5. Finally, calculate the average and maximum blocks and files that all users require.

Group	Users	Average Blocks Per User	Average Files Per User	Average Blocks/Group	Average Files/Group
dev	30	67108864	500	2013265920	15000
cit	15	10485760	50	157286400	750
pmgt	6	20971520	200	125829120	1200
portal	10	31457280	400	314572800	4000

Group	Users	Average Blocks Per User	Average Files Per User	Average Blocks/Group	Average Files/Group
lockbox	12	31457280	500	377487360	6000
jr23547	1	10485760	600	10485760	600
Total Blocks/ Files (Average)				2998927360	27550

Group	Users	Maximum Blocks Per User	Maximum Files Per User	Maximum Blocks/Group	Maximum Files/Group
dev	30	100663296	1000	3019898880	30000
cit	15	15728640	100	235929600	1500
pmgt	6	31457280	400	188743680	2400
portal	10	37748736	700	377487360	7000
lockbox	12	45613056	600	547356672	7200
jr23547	1	100663296	2000	100663296	2000
Total Blocks/ Files (Average)				4470079488	50100

6. If you need to administer project-based quotas or other quotas that cannot be defined by access-control group and user IDs, create admin sets for projects and for directories used by multiple groups.
7. If you need to set quotas on newly created file systems that do not yet hold files, enable quotas on these new file systems now.
8. Otherwise, if you need to set quotas on existing file systems that already hold files, enable quotas on these older file systems now.

Create Admin Sets for Projects and for Directories Used by Multiple Groups

An admin set is a directory hierarchy or an individual directory or file that is identified for quota purposes by an *admin set ID*. All files created with a specified admin set ID or stored in a directory with a specified admin set ID have the same quotas, regardless of the user or group IDs that actually own the files. To define admin sets, proceed as follows:

1. Log in to the file-system server as **root**.

   In the example, the server is named **mds**:

   ```
   root@mds:~#
   ```

2. If you are using an admin set to configure storage quotas for a new project or team, create a new directory somewhere within the file system for this project or team.

   In the example, we create the directory in the **/hsm/hqfs1** file system and name it **portalproject/** for the project of the same name:

   ```
   root@mds:~# mkdir /hsm/hqfs1/portalproject
   ```

3. Assign an admin set ID to the directory or file on which you need to set a quota. Use the command **samchaid [-fhR]** *admin-set-id directory-or-file-name*, where:

   - **-f** forces the assignment and does not report errors.

- **-h** assigns the admin set ID to symbolic links. Without this option, the group of the file referenced by the symbolic link is changed.
- **-R** assigns the admin set ID recursively to subdirectories and files.
- *admin-set-id* is a unique integer value.
- *directory-or-file-name* is the name of the directory or file to which you are assigning the admin set ID.

In the example, we assign the admin ID **1** to the directory **/hsm/hqfs1/portalproject/** and all of its subdirectories and files.

```
root@mds:~# samchaid -R 1 /hsm/hqfs1/portalproject/
```

4. You can check the assignment, if desired. Use the **command sls -D** *directory-path*, where the **-D** specifies a detailed Oracle HSM directory listing for files and directories in *directory-path*:

```
root@mds:~# sls -D /hsm/hqfs1/
/portalproject:
  mode: drwxr-xr-x    links:   2  owner: root        group: root
  length:      4096  admin id:    1  inode:     1047.1
  project: user.root(1)
  access:      Feb 24 12:49  modification: Feb 24 12:44
  changed:     Feb 24 12:49  attributes:   Feb 24 12:44
  creation:    Feb 24 12:44  residence:    Feb 24 12:44
```

5. If you need to set quotas on newly created file systems that do not yet hold files, enable quotas on these new file systems now.
6. Otherwise, if you need to set quotas on existing file systems that already hold files, enable quotas on these older file systems now.

Enable Quotas on New File Systems

Use this procedure if you are creating a new file system and if no files currently reside in the file system.

1. Log in to the file-system server as **root**.

   In the example, the server is named **mds**:

   ```
   root@mds:~#
   ```

2. If the new file system is not currently mounted, mount it before proceeding.

   ```
   root@mds:~# mount /hsm/newfs
   root@mds:~#
   ```

3. If you have to set up quotas for groups, create a group quota file in the file-system root directory, **.quota_g**. Use the Solaris command **dd if=/dev/zero of=***mountpoint***/.quota_g bs=4096 count=***number-blocks*, where:

   - **if=/dev/zero** specifies null characters from the UNIX special file **/dev/zero** as the input.
   - **of=***mountpoint***/.quota_g** specifies the output file, where *mountpoint* is the mount point directory for the file system.
   - **bs=4096** sets the block size for the write to **4096** bytes.

- **count=***number-blocks* specifies the number of blocks to write. This value depends on the number of records that the file will hold. There is one 128-byte record for each specified quota, so one block can accommodate 32 records.

In the example, we create the group quota file for the file system **newfs** mounted at **/newfs**. During the requirements-gathering phase, we identified three groups that need quotas on the file system, **dev**, **cit**, and **pgmt**. We do not anticipate adding any other group quotas, so we size the file at one block:

```
root@mds:~# dd if=/dev/zero of=/hsm/newfs/.quota_g bs=4096 count=1
```

4. If you have to set up quotas for admin sets, create an admin set quota file in the file-system root directory, **.quota_a**. Use the Solaris command **dd if=/dev/zero of=***mountpoint***/.quota_a bs=4096**, where:

   - *mountpoint* is the mount point directory for the file system.
   - **.quota_a** is the name of the output file.
   - **4096** is the block size for the write in bytes.
   - *number-blocks* is the number of blocks to write.

   In the example, we create the admin sets quota file for the file system **newfs** mounted at **/newfs**. During the requirements-gathering phase, we identified two projects that need quotas on the file system, **portal** (admin set ID **1**) and **lockbox** (admin set ID **2**). We do not anticipate adding any other admin set quotas, so we size the file at one block:

   ```
   root@mds:~# dd if=/dev/zero of=/hsm/newfs/.quota_a bs=4096 count=1
   ```

5. If you have to set up quotas for users, create a user quota file, **.quota_u**, in the file-system root directory. Use the Solaris command **dd if=/dev/zero of=***mountpoint***/.quota_u bs=4096 count=***number-blocks*, where:

   - *mountpoint* is the mount point directory for the file system.
   - **.quota_u** is the name of the output file.
   - **4096** is the block size for the write in bytes.
   - *number-blocks* is the number of blocks to write.

   In the example, we create the user quota file for the file system **newfs** mounted at **/newfs**. During the requirements-gathering phase, we identified one user that needed specific quotas on the file system, **jr23547**. We do not anticipate adding any other individual user quotas, so we size the file at one block:

   ```
   root@mds:~# dd if=/dev/zero of=/hsm/newfs/.quota_u bs=4096 count=1
   ```

6. Unmount the file system.

   You must unmount the file system before you can remount it and enable the quota files.

   ```
   root@mds:~# umount /hsm/newfs
   ```

7. Perform a file system check.

   ```
   root@mds:~# samfsck -F newfs
   samfsck: Configuring file system
   samfsck: Enabling the sam-fsd service.
   name:     newfs1       version:     2A
   First pass
   Second pass
   ```

```
Third pass
...
root@mds:~#
```

8. Remount the file system.

   The system enables quotas when it detects one or more quota files in the root directory of the file system.

   You do not need to include the **quota** mount option in the **/etc/vfstab** or **samfs.cmd** file, because file systems are mounted with quotas enabled by default.

```
root@mds:~# mount /hsm/newfs
```

9. If you need to set quotas on existing file systems that already hold files, enable quotas on these older file systems now.
10. Otherwise, set quotas as needed.

Enable Quotas on Existing File Systems

Use this procedure if you are creating quotas for a file system that already holds files.

1. Log in to the file-system server as **root**.

   In the example, the server is named **mds**:

```
root@mds:~#
```

2. Open the **/etc/vfstab** file in a text editor, and make sure that the **noquota** mount option has *not* been set.

   In the example, we open the file in the **vi** text editor. The **noquota** mount option has been set:

```
root@mds:~# vi /etc/vfstab
#File
#Device        Device   Mount             System  fsck  Mount    Mount
#to Mount      to fsck  Point             Type    Pass  at Boot  Options
#------------  -------  ---------------   ------  ----  -------  ------------
/devices       -        /devices          devfs   -     no       -
/proc          -        /proc             proc    -     no       -
...
hqfs1          -        /hsm/hqfs1        samfs   -     no       noquota
```

3. If the **noquota** mount option has been set in the **/etc/vfstab** file, delete it and save the file.

```
root@mds:~# vi /etc/vfstab
#File
#Device        Device   Mount             System  fsck  Mount    Mount
#to Mount      to fsck  Point             Type    Pass  at Boot  Options
#------------  -------  ---------------   ------  ----  -------  ------------
/devices       -        /devices          devfs   -     no       -
/proc          -        /proc             proc    -     no       -
...
hqfs1          -        /hsm/hqfs1        samfs   -     no       -
:wq
root@mds:~#
```

4. If a **samfs.cmd** file exists in the directory **/etc/opt/SUNWsamfs/**, open it in a text editor. In the section **fs =** *filesystem*, where *filesystem* is the name of the file

system in which you are enabling quotas, search for the mount option **noquota**, which may also be written **\0 quota**.

In the example, the **noquota** mount option has been applied to the **hqfs** file system:

```
root@mds:~# vi /etc/opt/SUNWsamfs/samfs.cmd
#inodes = 0
fs = hqfs1
#  forcedirectio (default no forcedirectio)
  high = 80
  low = 70
#  weight_size = 1.
#  weight_age = 1.
#  readahead = 128
...
   noquota
#  stripe = 1    (ms filesystem or ma filesystem with mr disks)
#  stripe = 0    (ma filesystem with striped groups)
#  dio_rd_form_min = 256
```

5. If the **noquota** mount option has been set in the **/etc/opt/SUNWsamfs/samfs.cmd** file, comment it out or delete it. Save the file, and close the editor.

   In the example, we comment out the line:

```
root@mds:~# vi /etc/opt/SUNWsamfs/samfs.cmd
#inodes = 0
fs = hqfs1
#  forcedirectio (default no forcedirectio)
  high = 80
  low = 70
#  weight_size = 1.
#  weight_age = 1.
#  readahead = 128
...
#  noquota
#  stripe = 1    (ms filesystem or ma filesystem with mr disks)
#  stripe = 0    (ma filesystem with striped groups)
#  dio_rd_form_min = 256
:wq
root@mds:~#
```

6. If you made changes to the **/etc/vfstab** and/or **/etc/opt/SUNWsamfs/samfs.cmd** files, unmount the file system and then remount it.

   When you remove or disable the **noquota** mount option in a configuration file, you must unmount and remount the affected file system to enable quotas.

```
root@mds:~# umount /hsm/hqfs1
root@mds:~# mount /hsm/hqfs1
root@mds:~#
```

7. Change to the root directory of the file system and check for any existing quota files. Use the Solaris command **ls -a** and look for the files **.quota_g**, **.quota_a**, and/or **.quota_u**.

   In the example, no quota files currently exist.

```
root@mds:~# cd /hsm/hqfs1
root@mds:~# ls -a /hsm/hqfs1
.              .archive       .fuid          .stage         portalproject
..             .domain        .inodes        lost+found
```

```
root@mds:~#
```

8. If quota files exist, do not modify them.
9. If you have to set up quotas for groups and the group quota file, **.quota_g**, does not already exist in the file-system root directory, create the file now. Use the Solaris command **dd if=/dev/zero of=***mountpoint***/.quota_g bs=4096 count=***number-blocks*, where:

   - **if=/dev/zero** specifies null characters from the UNIX special file **/dev/zero** as the input.
   - **of=***mountpoint***/.quota_g** specifies the output file, where *mountpoint* is the mount point directory for the file system.
   - **bs=4096** sets the block size for the write to **4096** bytes.
   - **count=***number-blocks* specifies the number of blocks to write. This value depends on the number of records that the file will hold. There is one 128-byte record for each specified quota, so one block can accommodate 32 records.

   In the example, we create the group quota file for the file system **/hsm/hqfs1** mounted at **/hsm/hqfs1**. During the requirements-gathering phase, we identified three groups that need quotas on the file system, **dev**, **cit**, and **pgmt**. We do not anticipate adding any other group quotas, so we size the file at one block:

```
root@mds:~# dd if=/dev/zero of=/hsm/hqfs1/.quota_g bs=4096 count=1
1+0 records in
1+0 records out
root@mds:~#
```

10. If you have to set up quotas for admin sets and the admin sets quota file, **.quota_a**, does not already exist in the file-system root directory, create the file now. Use the Solaris command **dd if=/dev/zero of=***mountpoint***/.quota_a bs=4096 count=***number-blocks*, where:

    - *mountpoint* is the mount point directory for the file system.
    - **.quota_a** is the name of the output file.
    - **4096** is the block size for the write in bytes.
    - *number-blocks* is the number of blocks to write.

    In the example, we create the admin sets quota file for the file system **/hsm/hqfs1** mounted at **/hsm/hqfs1**. During the requirements-gathering phase, we identified two projects that need quotas on the file system, **portal** (admin set ID **1**) and **lockbox** (admin set ID **2**). We do not anticipate adding any other admin set quotas, so we size the file at one block:

```
root@mds:~# dd if=/dev/zero of=/hsm/hqfs1/.quota_a bs=4096 count=1
1+0 records in
1+0 records out
root@mds:~#
```

11. If you have to set up quotas for users and the user quota file, **.quota_u**, does not already exist in the file-system root directory, create the file now. Use the Solaris command **dd if=/dev/zero of=***mountpoint***/.quota_u bs=4096 count=***number-blocks*, where:

    - *mountpoint* is the mount point directory for the file system.
    - **.quota_u** is the name of the output file.

- **4096** is the block size for the write in bytes.
- *number-blocks* is the number of blocks to write.

In the example, we create the user quota file for the file system **/hsm/hqfs1** mounted at **/hsm/hqfs1**. During the requirements-gathering phase, we identified one user that needed specific quotas on the file system, **jr23547**. We do not anticipate adding any other individual user quotas, so we size the file at one block:

```
root@mds:~# dd if=/dev/zero of=/hsm/hqfs1/.quota_u bs=4096 count=1
1+0 records in
1+0 records out
root@mds:~#
```

12. Unmount the file system.

```
root@mds:~# umount /hsm/hqfs1
root@mds:~#
```

13. Perform a file system check.

```
root@mds:~# samfsck -F /hsm/hqfs1
samfsck: Configuring file system
samfsck: Enabling the sam-fsd service.
name:     hqfs1       version:     2A
First pass
Second pass
Third pass
...
root@mds:~#
```

14. Remount the file system.

    The system enables quotas when it detects one or more quota files in the root directory of the file system.

    You do not need to include the **quota** mount option in the **/etc/vfstab** or **samfs.cmd** file, because file systems are mounted with quotas enabled by default.

```
root@mds:~# mount /hsm/hqfs1
root@mds:~#
```

15. Otherwise, set quotas as needed.

Set Quotas for Groups, Projects, Directories, and Users

You set new quotas and adjust existing ones using the **samquota** command. Follow the procedure below:

1. Once you have characterized storage requirements, decide on the appropriate quotas for each group, user, and non-group organization. Consider the following factors and make adjustments as necessary:
   - the size of the file system compared to the average and maximum number of blocks that all users require
   - the number of inodes in the file system compared to the average and maximum number of inodes that all users require
   - the numbers and types of users that are likely to be close to their maximum requirement at any given time.
2. Log in to the file-system server as **root**.

In the example, the server is named **mds**:

```
root@mds:~#
```

3. Set limits for each group that requires them. Use the command **samquota -b** *number-blocks*:*type*[:*scope*] **-f** *number-files*:*type*[:*scope*] **-t** *interval*[:*scope*] **-G** *groupID* [*directory-or-file*], where:

   - **-b** *number-blocks* sets the maximum number of 512-kilobyte blocks that can be stored in the file system to *number-blocks*, an integer (see the **samquota** man page for alternative ways of specifying size). A value of **0** (zero) specifies an unlimited number of blocks.
   - **:** is a field separator.
   - *type* specifies the kind of limit, either **h** for a hard limit or **s** for a soft limit.
   - *scope* (optional) identifies the type of storage that is subject to the limit. It can be either **o** for online (disk-cache) storage only or **t** for total storage, which includes both disk-cache and archival storage (the default).
   - **-f** *number-files* sets the maximum number of files that can be stored in the file system to *number-files*, an integer. A value of **0** (zero) specifies an unlimited number of files.
   - **-t** *number-seconds* sets the *grace period*, the time during which soft limits can be exceeded, to *number-seconds*, an integer representing a number of seconds (see the **samquota** man page for alternative ways of specifying time).
   - **-G** *groupID* specifies a group name or integer identifier for the group. A value of **0** (zero) sets the default limits for all groups.
   - *directory-or-file* (optional) is the mount point directory for a specific file system or a specific directory or file on which you need to set a quota.

   In the example, we use our estimates from the requirements-gathering phase to set both hard and soft limits on both the amount of storage space in the **/hsm/mds** file system that group **dev** can use and the numbers of files that it can store. We set the grace period to **43200** seconds (twelve hours) for online storage only:

   ```
   root@mds:~# samquota -b 3019898880:h:t -f 30000:h:t -G dev /hsm/mds
   root@mds:~# samquota -b 2013265920:s:t -f 15000:s:t -t 43200:o -G dev /hsm/mds
   root@mds:~#
   ```

4. Set limits for each admin set that requires them. Use the command **samquota -b** *number-blocks*:*type*[:*scope*] **-f** *number-files*:*type*[:*scope*] **-t** *interval*[:*scope*] **-A** *adminsetID* [*directory-or-file*], where **-A** *adminsetID* is the integer value that uniquely identifies the admin set.

   Setting *adminsetID* to **0** (zero) sets the default limits for all admin sets.

   In the example, we use our estimates from the requirements-gathering phase to set both hard and soft limits on both the amount of storage space in the **/hsm/mds** file system that the **portal** project (admin set ID **1**) can use and the numbers of files that it can store. We set the grace period to **43200** seconds (twelve hours) for total storage used, which is the default scope:

   ```
   root@mds:~# samquota -b 377487360:h:t -f 7000:h:t -A 1 /hsm/mds
   root@mds:~# samquota -b 314572800:s:t -f 4000:s:t -t 43200 -A 1 /hsm/mds
   root@mds:~#
   ```

5. Set limits for each individual user that requires them. Use the command **samquota -b** *number-blocks*:*type*[:*scope*] **-f** *number-files*:*type*[:*scope*]

-t *interval*[**:***scope*] **-U** *userID* [*directory-or-file*], where **-U** *userID* is a user name or integer identifier for the user.

Setting *userID* to **0** (zero) sets the default limits for all users.

In the example, we use our estimates from the requirements-gathering phase to set both hard and soft limits on both the amount of storage space in the **/hsm/mds** file system that user **jr23547** can use and the numbers of files that **jr23547** can store. We set the grace period to **1209600** seconds (two weeks) for total storage used, which is the default scope:

```
root@mds:~# samquota -b 100663296:h:t -f 600:h:t -U jr23547 /hsm/mds
root@mds:~# samquota -b 10485760:s:t -f 2000:s:t -t 1209600 -U jr23547 /hsm/mds
root@mds:~#
```

6. Stop here.

Repair Inconsistent Quotas

If you mount an Oracle HSM file system with the **noquota** mount option when there are quota files in the root directory, quota records become inconsistent as blocks or files are allocated or freed. In this situation, proceed as follows:

1. Log in to the file-system server as **root**.

   In the example, the server is named **mds**:

   ```
   root@mds:~#
   ```

2. Unmount the affected file system.

   In the example, we unmount file system **hqfs1**:

   ```
   root@mds:~# umount /hsm/hqfs1
   root@mds:~#
   ```

3. Open the **/etc/vfstab** file in a text editor, and make sure that the **noquota** mount option has not been set.

   In the example, we open the file in the **vi** text editor. The **noquota** mount option has been set:

   ```
   root@mds:~# vi /etc/vfstab
   #File
   #Device        Device   Mount             System  fsck  Mount    Mount
   #to Mount      to fsck  Point             Type    Pass  at Boot  Options
   #------------  -------  ---------------   ------  ----  -------  ------------
   /devices       -        /devices          devfs   -     no       -
   /proc          -        /proc             proc    -     no       -
   ...
   hqfs1          -        /hsm/hqfs1        samfs   -     no       noquota
   ```

4. If the **noquota** mount option has been set in the **/etc/vfstab** file, delete it and save the file.

   ```
   root@mds:~# vi /etc/vfstab
   #File
   #Device        Device   Mount             System  fsck  Mount    Mount
   #to Mount      to fsck  Point             Type    Pass  at Boot  Options
   #------------  -------  ---------------   ------  ----  -------  ------------
   /devices       -        /devices          devfs   -     no       -
   /proc          -        /proc             proc    -     no       -
   ...
   ```

```
hqfs1          -       /hsm/hqfs1        samfs   -     no       -
:wq
root@mds:~#
```

5. Open the **/etc/opt/SUNWsamfs/samfs.cmd** file in a text editor, and make sure that the **noquota** mount option has not been set.

   In the example, we open the file in the **vi** text editor. The **noquota** mount option has not been set:

```
root@mds:~# vi /etc/opt/SUNWsamfs/samfs.cmd
# These are the defaults.  To change the default behavior, uncomment the
# appropriate line (remove the '#' character from the beginning of the line)
# and change the value.
#
#inodes = 0
#fs = mds
#  forcedirectio (default no forcedirectio)
#  high = 80
#  low = 70
#  weight_size = 1.
#  weight_age = 1.
#  readahead = 128
...
#  dio_wr_ill_min = 0
#  dio_wr_consec = 3
#  qwrite        (ma filesystem, default no qwrite)
#  shared_writer (ma filesystem, default no shared_writer)
#  shared_reader (ma filesystem, default no shared_reader)
```

6. If the **noquota** mount option has been set in the **/etc/opt/SUNWsamfs/samfs.cmd** file, delete it and save the file.

7. Repair the inconsistent quota records. Use the command **samfsck -F** *family-set-name*, where *family-set-name* is the family set name for the file system in the **/etc/opt/SUNWsamfs/mcf** file.

```
root@mds:~# samfsck -F hqfs1
samfsck: Configuring file system
samfsck: Enabling the sam-fsd service.
name:     hqfs1       version:     2A
First pass
Second pass
Third pass
...
root@mds:~#
```

8. Remount the file system.

   The system enables quotas when it detects one or more quota files in the root directory of the file system.

   You do not need to include the **quota** mount option in the **/etc/vfstab** or **samfs.cmd** file, because file systems are mounted with quotas enabled by default.

```
root@mds:~# mount /hsm/hqfs1
root@mds:~#
```

9. Stop here.

Check Quotas

Both administrators and users can monitor quotas and resource usage. The **root** user can generate quota reports on users, groups, or admin sets with the **samquota** command. File-system users can check their own quotas using the **squota** command:

Monitor Quotas as the File-System Administrator

1. Log in to the file-system server as **root**.

   In the example, the server is named **mds**:

   ```
   root@mds:~#
   ```

2. To display quota statistics for all groups, use the command **samquota -g** [*directory-or-file*], where the optional *directory-or-file* parameter limits the scope of the report to the file system mounted on the specified directory, the specified directory itself, or the specified file.

   In the example, we request a report for the **hqfs1** file system, which is mounted at **/hsm/hqfs1**:

   ```
   root@mds:~# samquota -g /hsm/hqfs1
   ```

3. To display quota statistics for all admin sets, use the command **samquota -a** [*directory-or-file*], where the optional *directory-or-file* parameter limits the scope of the report to the file system mounted on the specified directory, the specified directory itself, or the specified file.

   In the example, we request a report for the **hqfs1** file system, which is mounted at **/hsm/hqfs1**:

   ```
   root@mds:~# samquota -a /hsm/hqfs1
   ```

4. To display quota statistics for all users, use the command **samquota -u** [*directory-or-file*], where the optional *directory-or-file* parameter limits the scope of the report to the file system mounted on the specified directory, the specified directory itself, or the specified file.

   In the example, we request a report for the **hqfs1** file system, which is mounted at **/hsm/hqfs1**:

   ```
   root@mds:~# samquota -u /hsm/hqfs1
   ```

5. To display quota statistics for a specific group, use the command **samquota -G** *groupID* [*directory-or-file*], where *groupID* specifies a group name or integer identifier for the group and where the optional *directory-or-file* parameter limits the scope of the report to the file system mounted on the specified directory, the specified directory itself, or the specified file.

   In the example, we request a report on quotas for the **dev** group in the **hqfs1** file system, which is mounted at **/hsm/hqfs1**:

   ```
   root@mds:~# samquota -G dev /hsm/hqfs1
   ```

6. To display quota statistics for a specific admin set, use the command **samquota -A** *adminsetID* [*directory-or-file*], where *adminsetID* specifies an integer identifier for the admin set and where the optional *directory-or-file* parameter limits the scope of the report to the file system mounted on the specified directory, the specified directory itself, or the specified file.

   In the example, we request a report on quotas for admin set **1** in the **hqfs1** file system, which is mounted at **/hsm/hqfs1**:

```
root@mds:~# samquota -A 1 /hsm/hqfs1
```

7. To display quota statistics for a specific user, use the command **samquota -U** *userID* **[***directory-or-file***]**, where *userID* specifies a user name or integer identifier for the user and where the optional *directory-or-file* parameter limits the scope of the report to the file system mounted on the specified directory, the specified directory itself, or the specified file.

   In the example, we request a report on the quotas for user **jr23547** in the **hqfs1** file system, which is mounted at **/hsm/hqfs1**:

   ```
   root@mds:~# samquota -U jr23547 /hsm/hqfs1
   ```

8. Stop here.

Monitor Your Own User Quota

1. Log in to a file-system host using your user ID.

   In the example, we log in to host **mds** as user **od447**:

   ```
   od447@mds:~#
   ```

2. To display quota statistics for all groups, use the command **squota [***directory-or-file***]**, where the optional *directory-or-file* parameter limits the scope of the report to the file system mounted on the specified directory, the specified directory itself, or the specified file.

   In the example, we request a report for all file systems:

   ```
   od447@mds:~# squota
                                        Limits
              Type    ID    In Use     Soft     Hard
       /hsm/hqfs1
       Files  group   101         1     1000     1200
       Blocks group   101         8    20000    30000
       Grace period                    25920
       No user quota entry.
   od447@mds:~#
   ```

3. Stop here.

Temporarily Extend or Cancel Grace Periods

When you need to, you can do any of the following:

- extend a grace period by a specified amount
- restart a grace period
- end a grace period early.

Extend a Grace Period by a Specified Amount

If a group, user, or admin set has exceeded the specified soft limit for its quota and needs to remain above the soft limit temporarily but for a period that is longer than the current grace period allows, you can grant the extension as follows:

1. Log in to the file-system server as **root**.

   In the example, we log in to host **mds**:

   ```
   root@mds:~#
   ```

2. Check the quota that requires an extension. Use the command **samquota -***quota-type ID* **[***directory-or-file***]** where:

   - *quota-type ID* is **G** plus a group name or ID number, **A** plus an admin set ID number, or **U** plus a user name or ID number.
   - *directory-or-file* (optional) is either the mount point directory for a specific file system or the specific directory or file for which you need to extend a grace period.

   In the example, the **dev** group is significantly over the soft limit and has only a couple of hours left in its grace period:

```
root@mds:~# samquota -G dev /hsm/hqfs1
                                Online Limits                    Total Limits
       Type   ID    In Use       Soft      Hard       In Use     Soft  Hard
/hsm/hqfs1
Files  group  101       323      15000      30000          323    15000  30000
Blocks group  101 3109330961 2013265920 3019898880 3109330961 2013265920
3019898880
Grace period                 4320                             4320
---> Warning:  soft limits to be enforced in 2h21m16s
root@mds:~#
```

3. Extend the grace period, if warranted. Use the command **samquota -***quota-type ID* **-x** *number-seconds* **[***directory-or-file***]**, where:

   - *quota-type ID* is **G** plus a group name or ID number, **A** plus an admin set ID number, or **U** plus a user name or ID number.
   - *directory-or-file* (optional) is either the mount point directory for a specific file system or the specific directory or file for which you need to extend a grace period.
   - *number-seconds* is an integer representing the number of seconds in the extension (see the **samquota** man page for alternative ways of specifying time).

   Enter **y** (yes) when prompted to continue.

   In the example, we extend the grace period for the **dev** group to **2678400** seconds (31 days) for files in the **hqfs1** file system:

```
root@mds:~# samquota -G dev -x 2678400 /hsm/hqfs1
Setting Grace Timer:  continue? y
```

   When we recheck the **dev** group quota, the grace period has been extended:

```
root@mds:~# samquota -G dev /hsm/hqfs1
                                Online Limits                    Total Limits
       Type    ID    In Use       Soft      Hard   In Use       Soft   Hard
/hsm/hqfs1
Files  group   101       323      15000      30000     323      15000   30000
Blocks group   101     43208 2013265920 3019898880   43208 2013265920
3019898880
Grace period                 2678400                          2678400
---> Warning:  soft limits to be enforced in 31d
root@mds:~#
```

4. If a group, admin set, or user regularly needs extensions, re-evaluate storage requirements and consider setting new quotas.
5. Stop here.

Restart the Grace Period

If a group, user, or admin set has exceeded the specified soft limit for its quota and cannot free space quickly enough to get below the soft limit before the current grace period expires, you can restart the grace period. Proceed as follows:

1. Log in to the file-system server as **root**.

   In the example, we log into host **mds**:

   ```
   root@mds:~#
   ```

2. Check the quota that requires an extension. Use the command **samquota -***quota-type ID* **[***directory-or-file***]** where:

   - *quota-type ID* is **G** plus a group name or ID number, **A** plus an admin set ID number, or **U** plus a user name or ID number.
   - *directory-or-file* (optional) is either the mount point directory for a specific file system or a specific directory or file for which you need to extend a grace period.

   In the example, the **cit** group is over the soft limit for the **hqfs1** file system and has just over an hour left in its grace period:

   ```
   root@mds:~# samquota -G cit /hsm/hqfs1
                                  Online Limits                    Total Limits
          Type   ID    In Use        Soft       Hard     In Use     Soft  Hard
   /hsm/hqfs1
   Files  group  119       762         750       1500        762      750  1500
   Blocks group  119 3109330961 2013265920 3019898880  120096782  157286400
   235929600
   Grace period                4320                              4320
   ---> Warning:  soft limits to be enforced in 1h11m23s
   root@mds:~#
   ```

3. To reset the grace period to its full starting size the next time that a file or block is allocated, *clear* the grace period timer. Use the command **samquota -***quota-type ID* **-x clear [***directory-or-file***]**, where:

   - *quota-type ID* is **G** plus a group name or ID number, **A** plus an admin set ID number, or **U** plus a user name or ID number.
   - *directory-or-file* (optional) is either the mount point directory for a specific file system or the specific directory or file for which you need to extend a grace period.

   Enter **y** (yes) when prompted to continue.

   In the example, we clear the grace-period timer for the **cit** group's quota on the **hqfs1** file system.

   ```
   root@mds:~# samquota -G cit -x clear /hsm/hqfs1
   Setting Grace Timer:  continue? y
   root@mds:~#
   ```

   When we recheck the **cit** group quota, a file has been allocated and the grace period has been reset to **12h**, 12 hours (**4320** seconds):

   ```
   root@mds:~# samquota -G cit /hsm/hqfs1
                                  Online Limits                    Total Limits
          Type   ID    In Use        Soft       Hard     In Use     Soft  Hard
   /hsm/hqfs1
   Files  group  119       763         750       1500        763      750  1500
   ```

```
Blocks group  119 3109330961 2013265920 3019898880  120096782  157286400
235929600
Grace period                 4320                              4320
---> Warning:  soft limits to be enforced in 12h
root@mds:~#
```

4. Alternatively, to reset the grace period to its full starting size immediately, *reset* the grace period timer. Use the command **samquota -***quota-type ID* **-x reset [***directory-or-file***]**.

   - *quota-type ID* is **G** plus a group name or ID number, **A** plus an admin set ID number, or **U** plus a user name or ID number.
   - *directory-or-file* (optional) is either the mount point directory for a specific file system or the specific directory or file for which you need to extend a grace period.

   Enter **y** (yes) when prompted to continue.

   In the example, we clear the grace-period timer for the **cit** group's quota on the **hqfs1** file system.

```
root@mds:~# samquota -G cit -x reset /hsm/hqfs1
Setting Grace Timer:  continue? y
root@mds:~#
```

   When we recheck the **cit** group quota, the grace period has been reset to **12h**, 12 hours (**4320** seconds):

```
root@mds:~# samquota -G cit /hsm/hqfs1
                                Online Limits                    Total Limits
       Type   ID    In Use        Soft       Hard     In Use     Soft  Hard
/hsm/hqfs1
Files  group  119       762         750       1500        762      750  1500
Blocks group  119 3109330961 2013265920 3019898880  120096782  157286400
235929600
Grace period                 4320                              4320
---> Warning:  soft limits to be enforced in 12h
root@mds:~#
```

5. Stop here.

End a Grace Period Early

1. Log in to the file-system server as **root**.

   In the example, we log into host **mds**:

```
root@mds:~#
```

2. Check the grace period that you need to cut short. Use the command **samquota -***quota-type ID* **[***directory-or-file***]** where:

   - *quota-type ID* is **G** plus a group name or ID number, **A** plus an admin set ID number, or **U** plus a user name or ID number.
   - *directory-or-file* (optional) is either the mount point directory for a specific file system or a specific directory or file for which you need to extend a grace period.

   In the example, the **cit** group is over the soft limit and has eleven hours left in its grace period, but we need to end the grace period early:

```
root@mds:~# samquota -G cit /hsm/hqfs1
```

```
                                Online Limits                    Total Limits
        Type   ID    In Use        Soft       Hard     In Use       Soft
Hard
/hsm/hqfs1
Files  group  119       822         750       1500        822        750
1500
Blocks group  119 3109330961 2013265920 3019898880  120096782  157286400
235929600
Grace period                4320                              4320
---> Warning:  soft limits to be enforced in 11h
root@mds:~#
```

3. *Expire* the grace period. Use the command **samquota -***quota-type ID* **-x expire [***directory-or-file***]**, where:

   - *quota-type ID* is **G** plus a group name or ID number, **A** plus an admin set ID number, or **U** plus a user name or ID number.
   - *directory-or-file* (optional) is either the mount point directory for a specific file system or a specific directory or file for which you need to extend a grace period.

   In the example, we expire the grace period for the **cit** group:

   ```
   root@mds:~# samquota -G cit -x expire /hsm/hqfs1
   Setting Grace Timer:  continue? y
   ```

   When we re-check quotas, soft limits for the **cit** group are being enforced as hard limits:

   ```
   root@mds:~# samquota -G cit /hsm/hqfs1
                                   Online Limits                    Total Limits
           Type   ID    In Use        Soft       Hard     In Use     Soft  Hard
   /hsm/hqfs1
   Files  group  119       762         750       1500        762      750  1500
   Blocks group  119 3109330961 2013265920 3019898880  120096782  157286400
   235929600
   Grace period                4320                              4320
   ---> Online soft limits under enforcement (since 6s ago)
   root@mds:~#
   ```

4. Stop here.

Stop New Resource Allocations

You can inhibit file-system resource allocations by creating inconsistent quota values. When the file system detects that quota values are not consistent for a user, group, or admin set, it prevents that user, group, or admin set from using any more system resources. So setting the hard limit for a quota lower than the corresponding soft limit stops further allocations. To use this technique, proceed as follows:

1. Log in to the file-system server as **root**.

   In the example, we log into host **mds**:

   ```
   root@mds:~#
   ```

2. Back up the quota so that you can restore it later. Export the current configuration, and redirect the information to a file. Use the command **samquota -***quota-type ID* **[***directory-or-file***] >** *file* where:

- *quota-type ID* is **G** plus a group name or ID number, **A** plus an admin set ID number, or **U** plus a user name or ID number.
- *directory-or-file* (optional) is either the mount point directory for a specific file system or a specific directory or file for which you need to extend a grace period.
- *file* is the name of the output file.

In the example, we export the quota for the **cit** group to the file **restore.hqfs1.quota_g.cit** in the **root** user's home directory:

```
root@mds:~# samquota -G cit -e /hsm/hqfs1 > /root/restore.hqfs1.quota_g.cit
root@mds:~#
```

3. Check the output. Use the Solaris command **more <** *file*, where *file* is the name of the output file.

```
root@mds:~# more < /root/restore.hqfs1.quota_g.cit
# Type  ID
#                  Online Limits                          Total   Limits
#              soft            hard                   soft       hard
# Files
# Blocks
# Grace Periods
samquota -G 119 \
    -f       750:s:o  -f      1500:h:o        -f       750:s:t -f
1500:h:t \
    -b 157286400:s:o  -b
235929600:h:o        -b 157286400:s:t -b 235929600:h:t \
            -t 4320:o                               -t 4320:t
root@mds:~#
```

4. Set the hard limits for the quota to **0** (zero) and set the soft limits to **1** (or any non-zero value). Use the command **samquota -***quota-type ID* **-f 1:s -f 0:h -b 1:s -b 0:h [***directory-or-file***]**.

   - *quota-type ID* is **G** plus a group name or ID number, **A** plus an admin set ID number, or **U** plus a user name or ID number.
   - *directory-or-file* (optional) is the mount point directory for a specific file system or a specific directory or file for which you need to extend a grace period.

   In the example, we make the quota settings for the **cit** group in the **/hsm/hqfs1** file system inconsistent, and thereby stop new resource allocations.

```
root@mds:~# samquota -G cit -f 1:s -f 0:h -b 1:s -b 0:h /hsm/hqfs1
root@mds:~#
```

   When we check the quota for the **cit** group, zero quotas are in effect. The exclamation point characters (**!**) show all current use as over-quota, so no further allocations will be made:

```
root@mds:~# samquota -G cit /hsm/hqfs1
Online Limits                Total Limits
Type   ID              In Use    Soft     Hard         In Use     Soft     Hard
/sam6
Files  group 119        822!       1        0         822!       1        0
Blocks group 119 3109330961!       1        0  3109330961!       1        0
Grace period                      4320                           4320
---> Quota values inconsistent; zero quotas in effect.
root@mds:~#
```

5. When you are ready resume normal allocations by restoring the modified quota to its original state. Execute the backup file that you created as a shell script. Use the Solaris command **sh** *file*, where *file* is the name of the backup file.

   In the example, we restore the quota for the **cit** group by executing the file **/root/restore.hqfs1.quota_g.cit**

```
root@mds:~# sh /root/restore.hqfs1.quota_g.cit
Setting Grace Timer:  continue? y
Setting Grace Timer:  continue? y
root@mds:~#
```

   When we check the quota, normal limits have been restored and allocations are no longer blocked:

```
root@mds:~# samquota -G cit /hsm/hqfs1
                                Online Limits                    Total Limits
        Type   ID    In Use        Soft       Hard     In Use     Soft  Hard
/hsm/hqfs1
Files  group  119       822         750       1500        822     750   1500
Blocks group  119 3109330961 2013265920 3019898880  120096782  157286400
235929600
Grace period                       4320                              4320
---> Warning:  soft limits to be enforced in 11h
root@mds:~#
```

6. Stop here.

Remove the Quotas for a File System

To remove or disable quotas for a file system, disable quotas in the mount process.

1. Log in to the file-system server as **root**.

   In the example, we log in to host **mds**:

```
root@mds:~#
```

2. Open the **/etc/vfstab** file in a text editor, add the **noquota** mount option to the mount options column of the file system row, and save the file.

   In the example, we open the file in the **vi** text editor, and set the **noquota** mount option for the **hqfs1** file system:

```
root@mds:~# vi /etc/vfstab
#File
#Device        Device   Mount             System  fsck  Mount    Mount
#to Mount      to fsck  Point             Type    Pass  at Boot  Options
#------------  -------  ----------------  ------  ----  -------  ------------
/devices       -        /devices          devfs   -     no       -
/proc          -        /proc             proc    -     no       -
...
hqfs1          -        /hsm/hqfs1        samfs   -     no       noquota
:wq
root@mds:~#
```

3. If the file system is mounted, unmount it.

   You must unmount and then remount a file system so that the operating system reloads the **/etc/vfstab** file and makes the specified changes. In the example, we unmount the **hqfs1** file system:

```
root@mds:~# umount /hsm/hqfs1
root@mds:~#
```

4. Mount the file system.

   In the example, we mount the **hqfs1** file system:

```
root@mds:~# mount /hsm/hqfs1
root@mds:~#
```

5. If you expect to reinstate quotas later, leave the quota files in pace.
6. If you do not expect to reinstate quotas or if you need to reclaim the space consumed by quota files, use the Solaris command **rm** to delete the files **.quota_g**, **.quota_a**, and/or **.quota_u** from the root directory of the file system.

   In the example, we remove all quota files from the **/hsm/hqfs1** file system root directory:

```
root@mds:~# rm /hsm/hqfs1/.quota_g
root@mds:~# rm /hsm/hqfs1/.quota_a
root@mds:~# rm /hsm/hqfs1/.quota_u
root@mds:~#
```

7. Stop here.

Controlling Archiving and Staging Operations

In general, you manage archiving file systems in much the same was as you would non-archiving file systems. However, you must stop the archiving process before carrying out most file-system management tasks. When active, the archiving processes make changes to the file-system's primary disk cache. So you must quiesce these processes before you do maintenance work on the disk cache. This section covers the following tasks:

- idling archiving and staging processes
- stopping archiving and staging processes
- restarting archiving and staging processes.

Idle Archiving and Staging Processes

1. Log in to the file system host as **root**.

   In the example, we log in to host **mds**:

```
root@mds:~#
```

2. Idle all archiving processes. Use the command **samcmd aridle**.

   This command will allow current archiving and staging to complete, but will not start any new jobs:

```
root@mds:~# samcmd aridle
root@mds:~#
```

3. Idle all staging processes. Use the command **samcmd stidle**.

   This command will allow current archiving and staging to complete, but will not start any new jobs:

```
root@mds:~# samcmd stidle
root@mds:~#
```

4. Wait for active archiving jobs to complete. Check on the status of the archiving processes using the command **samcmd a**.

   When archiving processes are **Waiting for :arrun**, the archiving process is idle:

```
root@mds:~# samcmd a
Archiver status samcmd     5.4 10:20:34 May 20 2014
samcmd on samfs-mds
sam-archiverd:  Waiting for :arrun
sam-arfind: ...
Waiting for :arrun
```

5. Wait for active staging jobs to complete. Check on the status of the staging processes using the command **samcmd u**.

   When staging processes are **Waiting for :strun**, the staging process is idle:

```
root@mds:~# samcmd u
Staging queue samcmd      5.4  10:20:34 May 20 2014
samcmd on solaris.demo.lan
Staging queue by media type: all
sam-stagerd:  Waiting for :strun
root@mds:~#
```

6. To fully quiesce the system, stop archiving and staging processes as well.

Stop Archiving and Staging Processes

1. If you have not already done so, idle archiving and staging processes.
2. If you have not already done so, log in to the file system host as **root**.

   In the example, we log in to host **mds**:

```
root@mds:~#
```

3. Idle all removable media drives before proceeding further. For each drive, use the command **samcmd** *equipment-number* **idle**, where *equipment-number* is the equipment ordinal number assigned to the drive in the **/etc/opt/SUNWsamfs/mcf** file.

   This command will allow current archiving and staging jobs to complete before turning drives **off**, but will not start any new work. In the example, we idle four drives, with ordinal numbers **801**, **802**, **803**, and **804**:

```
root@mds:~# samcmd 801 idle
root@mds:~# samcmd 802 idle
root@mds:~# samcmd 803 idle
root@mds:~# samcmd 804 idle
root@mds:~#
```

4. Wait for running jobs to complete.

   We can check on the status of the drives using the command **samcmd r**. When all drives are **notrdy** and **empty**, we are ready to proceed.

```
root@mds:~# samcmd r
Removable media samcmd     5.4 18:37:09 Feb 17 2014
samcmd on hqfs1host
ty   eq   status      act  use  state  vsn
li  801   ---------p    0   0%  notrdy
          empty
li  802   ---------p    0   0%  notrdy
```

```
          empty
li  803   ---------p    0   0%  notrdy
          empty
li  804   ---------p    0   0%  notrdy
          empty
root@mds:~#
```

5. When the archiver and stager processes are idle and the tape drives are all **notrdy**, stop the library-control daemon. Use the command **samd stop**.

```
root@mds:~# samd stop
root@mds:~#
```

6. Proceed with file-system maintenance.
7. When maintenance is complete, restart archiving and staging processes.

   When you restart operations, pending stages are reissued and archiving is resumed.
8. Stop here.

Restart Archiving and Staging Processes

When you are ready, resume normal, automatic operation, proceed a follows:

1. Log in to the file system host as **root**.

   In the example, we log in to host **mds**:

```
root@mds:~#
```

2. Restart the Oracle HSM library-control daemon. Use the command **samd start**.

```
root@mds:~# samd start
root@mds:~#
```

3. Stop here.

Renaming File Systems

Renaming a file system is a two-step process. First you change the family set name for the file system by editing the **/etc/opt/SUNWsamfs/mcf** file. Then you have the **samfsck -R -F** command read the new name and update the superblock on the corresponding disk devices. To rename a file system, use the procedure below:

Rename a File System

1. Log in to the file-system server as **root**.

   In the example, we log in to host **mds**:

```
root@mds:~#
```

2. If you are renaming an archiving file system, finish active archiving and staging jobs and stop any new activity before proceeding further.
3. Unmount the file system that you need to rename.

   In the example, we unmount file system **hqfs1**:

```
root@mds:~# umount hqfs1
root@mds:~#
```

4. Open the **/etc/opt/SUNWsamfs/mcf** file in a text editor, land, in the first column of the file, change the equipment identifier of the file system to the new name.

   In the example, we use the **vi** editor to change the name of the file system **hqfs1** (equipment ordinal **100**) to **hpcc**:

```
root@mds:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment          Equipment  Equipment  Family        Device   Additional
# Identifier         Ordinal    Type       Set           State    Parameters
#----------------   ---------  ---------  ------------  ------   ----------
hpcc                100        ms         hqfs1         on
/dev/dsk/c1t3d0s3    101        md         hqfs1         on
/dev/dsk/c1t4d0s5    102        md         hqfs1         on
```

5. In the fourth column of the file, change the family set name of the file system to the new value. You may also change the file-system equipment identifier in the first column, but do not change anything else. Save the file and close the editor.

   In the example, we change both the family set name of the file system from **hqfs1** to **hpcc**:

```
root@mds:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment          Equipment  Equipment  Family        Device   Additional
# Identifier         Ordinal    Type       Set           State    Parameters
#----------------   ---------  ---------  ------------  ------   ----------
hpcc                100        ms         hpcc          on
/dev/dsk/c1t3d0s3    101        md         hpcc          on
/dev/dsk/c1t4d0s5    102        md         hpcc          on
:wq
root@mds:~#
```

6. If you are renaming an archiving file system, update the corresponding file system directive in the **archiver.cmd** file and, if configured, the **stager.cmd** file.

   In the example, we use the **vi** editor to to change the directive **fs = hqfs1** to **fs = hpcc**:

```
root@mds:~# vi /etc/opt/SUNWsamfs/archiver.cmd
# archiver.cmd: configuration file for archiving file systems
#----------------------------------------------------------------------
# General Directives
archivemeta = off                                                 # default
examine = noscan                                                  # default
#----------------------------------------------------------------------
# Archive Set Assignments
fs = hpcc
...
:wq
root@mds:~# vi /etc/opt/SUNWsamfs/archiver.cmd
# stager.cmd
logfile = /var/opt/SUNWsamfs/log/stager
drives= hp30 1
copysel = 4:3:2:1
fs = hpcc
...
:wq
root@mds:~#
```

7. Check the **mcf** file for errors by running the **sam-fsd** command, and correct any that are detected.

The **sam-fsd** is an initialization command that reads Oracle HSM configuration files. It will stop if it encounters an error. In the example, **sam-fsd** does not report any errors:

```
root@mds:~# sam-fsd
Trace file controls:
sam-amld      /var/opt/SUNWsamfs/trace/sam-amld
              cust err fatal ipc misc proc date
...
Would start sam-archiverd()
Would start sam-stagealld()
Would start sam-stagerd()
Would start sam-amld()
root@mds:~#
```

8. Rewrite the file-system super block to reflect the new family set name. Use the command **samfsck -R -F** *family-set-name*, where *family-set-name* is the family set name that you just specified in the **/etc/opt/SUNWsamfs/mcf** file.

   When issued with the **-R** and **-F** options, the **samfsck** command reads the new family set name and the corresponding disk-storage equipment identifiers from the **/etc/opt/SUNWsamfs/mcf** file. It then rewrites the super block on the specified disk devices with the new family set name. In the example, we run the command with the new **hpcc** family set name:

   ```
   root@mds:~# samfsck -R -F hpcc
   ...
   root@mds:~#
   ```

9. Open the **/etc/vfstab** file in a text editor, and locate the entry for the file system that you are renaming.

   In the example, we open the file in the **vi** text editor. We need to change the **hqfs1** file system entry to use the new name:

   ```
   root@mds:~# vi /etc/vfstab
   #File
   #Device        Device   Mount             System  fsck  Mount    Mount
   #to Mount      to fsck  Point             Type    Pass  at Boot  Options
   #------------  -------  ---------------   ------  ----  -------  ------------
   /devices       -        /devices          devfs   -     no       -
   /proc          -        /proc             proc    -     no       -
   ...
   hqfs1          -        /hsm/hqfs1        samfs   -     no       -
   ```

10. In the **/etc/vfstab** entry for the file system that you have renamed, change the file system name in the first column and the mount-point directory name in the third column (if required), and save the file.

    In the example, we change the name of the **hqfs1** file system to **hpcc** and change the mount point to match:

    ```
    root@mds:~# vi /etc/vfstab
    #File
    #Device        Device   Mount             System  fsck  Mount    Mount
    #to Mount      to fsck  Point             Type    Pass  at Boot  Options
    #------------  -------  ---------------   ------  ----  -------  ------------
    /devices       -        /devices          devfs   -     no       -
    /proc          -        /proc             proc    -     no       -
    ...
    hpcc           -        /hsm/hpcc         samfs   -     no       -
    :wq
    ```

```
root@mds:~#
```

11. Create the new mount-point directory for the new file system, if required, and set the access permissions for the mount point.

    Users must have execute (**x**) permission to change to the mount-point directory and access files in the mounted file system. In the example, we create the **/hsm/hpcc** mount-point directory and set permissions to **755** (**-rwxr-xr-x**):

```
root@mds:~# mkdir /hsm/hpcc
root@mds:~# chmod 755 /hsm/hpcc
root@mds:~#
```

12. Tell the Oracle HSM software to re-read the **mcf** file and reconfigure itself accordingly. Use the command **samd config**.

```
root@mds:~# samd config
Configuring SAM-FS ...
root@mds:~#
```

13. If **samd config** reports errors, correct them and re-issue the command until no errors are found.
14. Mount the file system.

    In the example, we use the new mount point directory:

```
root@mds:~# mount /hsm/hpcc
```

15. Stop here.

Repairing File Systems

When file systems report errors via **samu**, Oracle HSM Manager, or the **/var/adm/sam-log** file, follow the procedure below:

Repair a File System

1. Log in to the file-system server as **root**.

   In the example, we log in to host **mds**:

```
root@mds:~#
```

2. If you are repairing an archiving file system, finish active archiving and staging jobs and stop any new activity before proceeding further.
3. Unmount the affected file system.

   You may need to try more than once if you are waiting for archiving to stop. In the example, we unmount file system **hqfs1**:

```
root@mds:~# umount /hsm/hqfs1
samfs umount: /hsm/hqfs1: is busy
root@mds:~# umount /hsm/hqfs1
root@mds:~#
```

4. Repair the file system. Use the command **samfsck -F -V** *family-set-name*, where *family-set-name* is the family set name specified for the file system in the **/etc/opt/SUNWsamfs/mcf** file.

It is often a good idea to save the repair results to a date-stamped file for later reference and for diagnostic purposes, when necessary. So in the example, we save the results by piping the **samfsck** output to the command **tee /var/tmp/samfsck-FV.***family-set-name***.`date '+%Y%m%d.%H%M%S'`**:

```
root@mds:~# samfsck -F -V hqfs1 | tee /var/tmp/samfsck-FV.hqfs1.
`date '+%Y%m%d.%H%M%S'`
name:     /hsm/hqfs1       version:     2A
First pass
Second pass
Third pass
NOTICE: ino 2.2,  Repaired link count from 8 to 14
Inodes processed: 123392
total data kilobytes       = 1965952
total data kilobytes free  = 1047680
total meta kilobytes       = 131040
total meta kilobytes free  = 65568
INFO:  FS samma1 repaired:
        start:  May 19, 2014 10:57:13 AM MDT
        finish: May 19, 2014 10:57:37 AM MDT
NOTICE: Reclaimed 70057984 bytes
NOTICE: Reclaimed 9519104 meta bytes
root@mds:~#
```

5. Remount the file system.

```
root@mds:~# mount /hsm/hqfs1
root@mds:~#
```

6. Stop here.

Adding Devices to File Systems

Before you add devices to an existing file system, you should consider your requirements and your alternatives. Make sure that enlarging the existing file system is the best way to meet growing capacity requirements. If you need more physical storage space to accommodate new projects or user communities, creating one or more new Oracle HSM file systems may be a better choice. Multiple, smaller file systems will generally offer better performance than one much larger file system, and the smaller file systems may be easier to create and maintain.

Once you have decided that you need to enlarge a file system, you can take either of two approaches:

- You can add devices to the mounted file system (recommended)
- You can unmount the file system before adding devices.

Add Devices to a Mounted File System

Proceed as follows:

1. Log in to the file-system server as **root**.

   In the example, we log in to host **mds**:

```
root@mds:~#
```

2. Open the **/etc/opt/SUNWsamfs/mcf** file in a text editor, and locate the file system that you need to enlarge.

In the examples, we use the **vi** editor. We need to enlarge two file systems, the general-purpose **samqfsms** file system and the high-performance **samqfs2ma** file system:

```
root@mds:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment         Equipment  Equipment  Family     Device   Additional
# Identifier        Ordinal    Type       Set        State    Parameters
#-----------------  ---------  ---------  ---------  ------   ---------------
samqfsms            100        ms         samqfsms   on
/dev/dsk/c1t3d0s3   101        md         samqfsms   on
/dev/dsk/c1t4d0s5   102        md         samqfsms   on
samqfs2ma           200        ma         samqfs2ma  on
/dev/dsk/c1t3d0s3   201        mm         samqfs2ma  on
/dev/dsk/c1t3d0s5   202        md         samqfs2ma  on
/dev/dsk/c1t4d0s5   203        md         samqfs2ma  on
```

3. If you are adding devices to a general-purpose **ms** file system, add additional data/metadata devices to the end of the file system definition in the **mcf** file. Then save the file, and close the editor.

   You can add up to 252 logical devices. In the example, we add two devices, **103** and **104**, to the **samqfsms** file system:

```
root@mds:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment         Equipment  Equipment  Family     Device   Additional
# Identifier        Ordinal    Type       Set        State    Parameters
#-----------------  ---------  ---------  ---------  ------   ---------------
samqfsms            100        ms         samqfsms    on
/dev/dsk/c1t3d0s3   101        md         samqfsms    on
/dev/dsk/c1t4d0s5   102        md         samqfsms    on
/dev/dsk/c1t3d0s7   103        md         samqfsms    on
/dev/dsk/c1t4d0s7   104        md         samqfsms    on
:wq
root@mds:~#
```

4. If you are adding devices to a high-performance **ma** file system, add data devices and one or more **mm** disk devices to the end of the file system definition in the **mcf** file. Then save the file, and close the editor.

   Always add new devices at the end of the list of existing devices. You can add up to 252, adding metadata devices proportionately as you add data devices. In the example, we add one **mm** metadata device, **204**, and two **md** data devices, **205** and **206**, to the **samqfs2ma** file system:

```
root@mds:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment         Equipment  Equipment  Family     Device   Additional
# Identifier        Ordinal    Type       Set        State    Parameters
#-----------------  ---------  ---------  ---------  ------   ---------------
...
samqfs2ma           200        ma         samqfs2ma  on
/dev/dsk/c1t3d0s3   201        mm         samqfs2ma  on
/dev/dsk/c1t3d0s5   202        md         samqfs2ma  on
/dev/dsk/c1t4d0s5   203        md         samqfs2ma  on
/dev/dsk/c1t5d0s6   204        mm         samqfs2ma  on
/dev/dsk/c1t3d0s7   205        md         samqfs2ma  on
/dev/dsk/c1t4d0s7   206        md         samqfs2ma  on
:wq
root@mds:~#
```

5. Check the **mcf** file for errors by running the **sam-fsd** command, and correct any that are detected.

The **sam-fsd** is an initialization command that reads Oracle HSM configuration files. It will stop if it encounters an error:

```
root@mds:~# sam-fsd
```

6. If the **sam-fsd** command finds an error in the **mcf** file, edit the file to correct the error and recheck as described in the preceding step.

   In the example below, **sam-fsd** reports an unspecified problem with a device:

```
root@mds:~# sam-fsd
Problem in mcf file /etc/opt/SUNWsamfs/mcf for filesystem samqfsms
sam-fsd: Problem with file system devices.
```

   Usually, such errors are the result of inadvertent typing mistakes. Here, when we open the **mcf** file in an editor, we find that we have typed a letter **o** instead of a 0 in the equipment name for device **104**, the second new **md** device:

```
samqfsms                   100       ms         samqfsms    on
/dev/dsk/c1t3d0s3          101       md         samqfsms    on
/dev/dsk/c1t4d0s5          102       md         samqfsms    on
/dev/dsk/c1t3d0s7          103       md         samqfsms    on
/dev/dsk/c1t4dos7          104       md         samqfsms    on
                  ^
```

7. If the **sam-fsd** command runs without error, the **mcf** file is correct. Proceed to the next step.

   The example is a partial listing of error-free output:

```
root@mds:~# sam-fsd
Trace file controls:
sam-amld      /var/opt/SUNWsamfs/trace/sam-amld
              cust err fatal ipc misc proc date
...
Would start sam-archiverd()
Would start sam-stagealld()
Would start sam-stagerd()
Would start sam-amld()
root@mds:~#
```

8. Tell the Oracle HSM software to re-read the **mcf** file and reconfigure itself accordingly. Use the command **samd config**.

```
root@mds:~# samd config
Configuring SAM-FS
root@mds:~#
```

9. Make sure that **samd config** has updated the Oracle HSM file system configuration to include the new devices. Use the command **samcmd f**.

   The devices should be in the **off** state. In the example, **samcmd f** shows the new devices, **103** and **104**, and both are **off**:

```
root@mds:~# samcmd f
File systems samcmd     5.4 16:57:35 Feb 27 2014
samcmd on mds
ty      eq   state device_name       status      high low mountpoint server
ms     100   on    samqfsms          m----2-----  80% 70% /samqfsms
 md     101   on   /dev/dsk/c1t3d0s3
 md     102   on   /dev/dsk/c1t4d0s5
 md     103   off  /dev/dsk/c1t3d0s7
 md     104   off  /dev/dsk/c1t4d0s7
root@mds:~#
```

10. Enable the newly added devices. For each device, use the command **samcmd add** *equipment-number*, where *equipment-number* is the equipment ordinal number assigned to the device in the **/etc/opt/SUNWsamfs/mcf** file.

    In the example, we enable new devices, **103** and **104**:

```
root@mds:~# samcmd add 103
root@mds:~# samcmd add 104
```

11. If you are working on a shared file system, finish configuring the new devices using the procedure for shared file systems.

12. If you are working on an unshared, standalone file system, make sure that the devices were added and are ready for use by the file system. Use the command **samcmd m**, check the results.

    When the device is in the **on** state, it has been added successfully and is ready to use. In the example, we have successfully added devices **103** and **104**:

```
root@mds:~# samcmd f
Mass storage status samcmd     5.4 17:17:08 Feb 27 2014
samcmd on mds
ty   eq  status       use state  ord  capacity      free  ra  part high low
ms   100  m----2----- 13% on           3.840G    3.588G  1M    16  80% 70%
 md   101             31% on       0  959.938M  834.250M
 md   102             13% on       1  959.938M  834.250M
 md   103              0% on       2  959.938M  959.938M
 md   104              0% on       3  959.938M  959.938M
root@mds:~#
```

13. Stop here.

Finish Configuring New Devices Added to a Shared File System

When you add devices to a shared file system, you must carry out a few more steps before the devices are configured on all file-system hosts. Proceed as follows:

1. Log in to the file-system metadata server host as **root**.

    In the example, the metadata server host is named **mds1**:

```
root@mds1:~#
```

2. Make sure that the new devices were added to the metadata server. Use the command **samcmd m**.

    When the device is in the **unavail** state, it has been added successfully but is not yet ready for use. In the example, we have successfully added devices **103** and **104**:

```
root@mds1:~# samcmd f
Mass storage status samcmd     5.4 17:17:08 Feb 27 2014
samcmd on metadata-server
ty      eq  status       use state  ord   capacity     free    ra  part high
low
ms     100  m----2-----  13% on           3.840G    3.588G    1M    16  80%
70%
 md     101              31% on       0  959.938M  834.250M
 md     102              13% on       1  959.938M  834.250M
 md     103               0% unavail  2  959.938M  959.938M
 md     104               0% unavail  3  959.938M  959.938M
root@mds1:~#
```

3. Log in to each file-system client hosts as **root**.

   Remember to include potential metadata servers, since they are also clients. In the example, we need to log in to a potential metadata server, named **mds2**, and two clients, **clnt1** and **clnt2L** (a Linux host). So we open three terminal windows and use secure shell (**ssh**):

```
root@mds1:~# ssh root@mds2
Password:
root@mds2:~#

root@mds1:~# ssh root@clnt1
Password:
root@clnt1:~#

root@mds1:~# ssh root@clnt2L
Password:
[root@clnt2L ~]#
```

4. If the client is a Linux client, unmount the shared file system.

```
[root@clnt2L ~]# umount /hsm/shrfs1
[root@clnt2L ~]#
```

5. On each client, open the **/etc/opt/SUNWsamfs/mcf** file in a text editor, and add the new devices to the end of the file system definition, just as you did on the server.

   In the example, we add devices **103** and **104** to the **mcf** file on **clnt1**:

```
root@clnt1:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment          Equipment  Equipment  Family     Device   Additional
# Identifier         Ordinal    Type       Set        State    Parameters
#-----------------   ---------  ---------  ---------  ------   ----------
shrfs1              100        ms         shrfs1     on       shared
/dev/dsk/c1t3d0s3    101        md         shrfs1     on
/dev/dsk/c1t4d0s5    102        md         shrfs1     on
/dev/dsk/c1t3d0s7    103        md         shrfs1     on
/dev/dsk/c1t4d0s7    104        md         shrfs1     on
:wq
root@clnt1:~#
```

6. On each client, check the **mcf** file for errors by running the **sam-fsd** command, and correct any that are detected.

```
root@clnt1:~# sam-fsd
...
root@clnt1:~#

[root@clnt2L ~]# sam-fsd
...
[root@clnt2L ~]#
```

7. On each client, tell the Oracle HSM software to re-read the **mcf** file and reconfigure itself accordingly:

```
root@clnt1:~# samd config
...
root@clnt1:~#

[root@clnt2L ~]# samd config
...
[root@clnt2L ~]#
```

8. If the client is a Linux client, mount the shared file system.

```
[root@clnt2L ~]# mount /hsm/shrfs1
[root@clnt2L ~]#
```

9. Once all clients have been configured, return to the metadata server, and enable storage allocation on the new devices. For each device, use the command **samcmd alloc** *equipment-number*, where *equipment-number* is the equipment ordinal number assigned to the device in the **/etc/opt/SUNWsamfs/mcf** file.

   In the example, we enable storage allocation on devices **103** and **104**:

```
root@mds1:~# samcmd alloc 103
root@mds1:~# samcmd alloc 104
```

10. Finally, make sure that the devices are ready for use by the file system. Use the command **samcmd m**, and check the results.

    When the device is in the **on** state, it has been added successfully and is ready to use. In the example, we have successfully added devices **103** and **104**:

```
root@mds1:~# samcmd f
Mass storage status samcmd     5.4 17:17:08 Feb 27 2014
samcmd on metadata-server
ty      eq  status       use state  ord   capacity     free    ra  part high
low
ms     100  m----2-----  13% on            3.840G    3.588G    1M    16  80%
70%
 md     101              31% on       0  959.938M  834.250M
 md     102              13% on       1  959.938M  834.250M
 md     103               0% on       2  959.938M  959.938M
 md     104               0% on       3  959.938M  959.938M
root@mds1:~#
```

11. Stop here.

Add Devices to an Unmounted File System

Proceed as follows:

1. Log in to the file-system server host as **root**.

   In the example, the metadata server host is named **mds**:

```
root@mds:~#
```

2. If you are adding devices to an archiving file system, finish active archiving and staging jobs and stop any new activity before proceeding further.

3. Unmount the file system.

   Do not proceed until you have unmounted the file system. In the example, we unmount file system **hqfs1**:

```
root@mds:~# umount /hsm/hqfs1
root@mds:~#
```

4. Open the **/etc/opt/SUNWsamfs/mcf** file in a text editor, and locate the file system that you need to enlarge.

   In the example, we use the **vi** editor. We need to enlarge the **hqfs1** file system:

```
root@mds:~# vi /etc/opt/SUNWsamfs/mcf
```

```
# Equipment           Equipment  Equipment  Family     Device   Additional
# Identifier          Ordinal    Type       Set        State    Parameters
#-----------------    ---------  ---------  ---------  ------   ---------
hqfs1                 100        ms         hqfs1      on
/dev/dsk/c1t3d0s3     101        md         hqfs1      on
/dev/dsk/c1t4d0s5     102        md         hqfs1      on
```

5. If you are adding devices to a high-performance **ma** file system, you must add metadata storage along with the data storage. Add enough additional **mm** disk devices to store the metadata for the data devices that you add. Then save the file, and close the editor.

   You can add up to 252 logical devices. In the example, we add one **mm** metadata device and two data devices to the **hqfs1** file system:

```
root@mds:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment           Equipment  Equipment  Family     Device   Additional
# Identifier          Ordinal    Type       Set        State    Parameters
#-----------------    ---------  ---------  ---------  ------   ---------
hqfs1                 200        ma         hqfs1      on
/dev/dsk/c1t3d0s3     201        mm         hqfs1      on
/dev/dsk/c1t5d0s6     204        mm         hqfs1      on
/dev/dsk/c1t3d0s5     202        md         hqfs1      on
/dev/dsk/c1t4d0s5     203        md         hqfs1      on
/dev/dsk/c1t3d0s7     205        md         hqfs1      on
/dev/dsk/c1t4dos7     206        md         hqfs1      on
:wq
root@mds:~#
```

6. If you are adding devices to a general-purpose **ms** file system, add additional data/metadata devices to the file system definition in the **mcf** file. Then save the file, and close the editor.

   You can add up to 252 logical devices. In the example, we add two devices to the **hqfs1** file system:

```
root@mds:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment           Equipment  Equipment  Family     Device   Additional
# Identifier          Ordinal    Type       Set        State    Parameters
#-----------------    ---------  ---------  ---------  ------   ---------
hqfs1                 100        ms         hqfs1      on
/dev/dsk/c1t3d0s3     101        md         hqfs1      on
/dev/dsk/c1t4d0s5     102        md         hqfs1      on
/dev/dsk/c1t3d0s7     103        md         hqfs1      on
/dev/dsk/c1t4dos7     104        md         hqfs1      on
:wq
root@mds:~#
```

7. Check the **mcf** file for errors by running the **sam-fsd** command, and correct any that are detected.

   The **sam-fsd** is an initialization command that reads Oracle HSM configuration files. It will stop if it encounters an error:

```
root@mds:~# sam-fsd
Trace file controls:
sam-amld      /var/opt/SUNWsamfs/trace/sam-amld
...
Would start sam-archiverd()
Would start sam-stagealld()
Would start sam-stagerd()
Would start sam-amld()
```

```
root@mds:~#
```

8. Tell the Oracle HSM software to re-read the **mcf** file and reconfigure itself accordingly:

```
root@mds:~# samd config
...
root@mds:~#
```

9. Incorporate the new devices into file system. Use the command **samgrowfs** *family-set-name*, where *family-set-name* is the family set name specified for the file system in the **/etc/opt/SUNWsamfs/mcf** file.

   In the example, we grow the **hqfs1** file system:

```
root@mds:~# samgrowfs hqfs1
...
root@mds:~#
```

10. Remount the file system.

```
root@mds:~# mount /hsm/hqfs1
root@mds:~#
```

11. If you added devices to an archiving file system, restart the Oracle HSM library-management daemon. Use the command **samd start**.

```
root@mds:~# samd start
...
root@mds:~#
```

12. If you neglected to unmount the file system before making changes and if, consequently, the file system will not mount, restore the original **mcf** file by deleting references to the added devices. Then run **samd config** to restore the configuration, unmount the file system, and start over.

13. Stop here.

Removing Data Devices from a File System

When required, you can remove data devices from mounted Oracle HSM file systems. Typically this becomes necessary when you need to replace a failed unit or when you need to free up under-utilized devices for other uses. There are, however, some limitations.

You can only remove data devices. You cannot remove any devices used to hold metadata, since metadata defines the organization of the file system itself. This means that you can remove **md**, **mr**, and striped-group devices from high-performance **ma** file systems only. You cannot remove **mm** metadata devices from **ma** file systems. Nor can you remove **md** devices from general purpose **ms** file systems, since these devices store both data and metadata.

To remove devices, you must also have somewhere to move any valid data files that reside on the target device. This means that you cannot remove all the devices. One device must always remain available in the file system and it must have enough free capacity to hold all files residing on the devices that you remove. So, if you need to remove a striped group, you must have another available striped group configured with an identical number of member devices.

To remove devices, proceed as follows:

- Make sure that file-system metadata and data are backed up.

- Remove devices from the mounted high-performance file system.

Make Sure that File-System Metadata and Data are Backed Up

Carry out the following tasks:

- Run **samexplorer**.
- Create a recovery point file for the file system.

Run samexplorer

1. Log in to the file-system server host as **root**.

   In the example, the metadata server host is named **mds**:

   ```
   root@mds:~#
   ```

2. Create a **samexplorer** report. Use the command **samexplorer** *path/hostname*.*YYYYMMDD*.*hhmmz*.*tar.gz*, where:

   - *path* is the path to the chosen directory.
   - *hostname* is the name of the Oracle HSM file system host.
   - *YYYYMMDD*.*hhmmz* is a date and time stamp.

   By default, the file is called **/tmp/SAMreport.***hostname*.*YYYYMMDD*.*hhmmz*.**tar.gz**. In the example, we use the directory **/zfs1/hsmcfg/**, where **/zfs1** is a file system that has no components in common with the Oracle HSM file system:

   ```
   root@mds:~# samexplorer /zfs1/hsmcfg/SAMreport.mds.2016013.1659MST.tar.gz

        Report name:     /zfs1/hsmcfg/SAMreport.mds.2016013.1659MST.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.

        /zfs1/hsmcfg/SAMreport.mds.2016013.1659MST.tar.gz
   ```

3. Create a recovery point file for the file system.

Create a Recovery Point File for the File System

1. Log in to the file-system server host as **root**.

   In the example, the metadata server host is named **mds**:

   ```
   root@mds:~#
   ```

2. Select the location where the recovery point file will be stored. The selected location must share no devices with the file system that you are backing up and must have room to store an unusually large file.

   The devices that we intend to remove may contain files that have not been archived. Since such files exist only as single copies, we will have to create a recovery point file that stores at least some data as well as metadata. This can substantially increase the size of the recovery point file.

In the example, we create a subdirectory, **tmp/**, in a file system has no components in common with the Oracle HSM file system, **/zfs1**:

```
root@mds:~# mkdir /zfs1/tmp/
root@mds:~#
```

3. Change to the file system's root directory.

   In the example, we change to the mount-point directory **/hsm/hqfs1**:

```
root@mds:~# cd /hsm/hqfs1
root@mds:~#
```

4. Back up the file-system metadata and any unarchived data. Use the command **samfsdump -f -u** *recovery-point*, where *recovery-point* is the path and file name of the finished recovery point file.

   Note that the **-u** option adds the data portion of unarchived files to the recovery point. This can greatly increase the size of the file.

   In the example, we create a recovery point file for the **hqfs1** file system called **hqfs1-20140313.025215** in the directory **/zfs1/hsmcfg/**. We check the result using the command **ls -l**:

```
root@mds:~# cd /hsm/hqfs1
root@mds:~# samfsdump -f /zfs1/hsmcfg/hqfs1-`date '+%Y%m%d.%H%M%S'` -T
/hsm/hqfs1
samfsdump statistics:
    Files:              10010
    Directories:        2
    Symbolic links:     0
    Resource files:     0
    Files as members of hard links :    0
    Files as first hard link :  0
    File segments:      0
    File archives:      10010
    Damaged files:      0
    Files with data:    0
    File  warnings:     0
    Errors:             0
    Unprocessed dirs:   0
    File data bytes:    0
root@mds:~# ls -l /zfs1/hsmcfg/hqfs1*
-rw-r--r-- 1 root other 5376517 Mar 13 02:52 /zfs1/hsmcfg/hqfs1-20140313.025215
root@mds:~#
```

5. Now remove devices from the mounted, high-performance file system.

Remove Devices from a Mounted High-Performance File System

You must remove devices one at a time. For each device, proceed as follows:

1. Log in to the file-system server host as **root**.

   In the example, the metadata server host is named **mds**:

```
root@mds:~#
```

2. Open the **/etc/opt/SUNWsamfs/mcf** file, and note the equipment ordinal number for the device that you need to remove.

In the example, we use the **vi** editor. We need to remove device **/dev/dsk/c1t4d0s7** from the equipment list for the **hqfs1** file system. The equipment ordinal number is **104**:

```
root@mds:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment          Equipment  Equipment  Family     Device   Additional
# Identifier         Ordinal    Type       Set        State    Parameters
#----------------   ---------  ---------  ---------  ------   --------------
hqfs1                100        ms         hqfs1      on
/dev/dsk/c1t3d0s3    101        md         hqfs1      on
/dev/dsk/c1t4d0s5    102        md         hqfs1      on
/dev/dsk/c1t3d0s7    103        md         hqfs1      on
/dev/dsk/c1t4d0s7    104        md         hqfs1      on
:q
root@mds:~#
```

3. Before you try to remove a device, make sure that the remaining devices in the file system can accept any files that have to be moved from the device that you intend to delete.

   - Make sure that the remaining devices have adequate capacity.
   - If the device is a striped group, make sure that the file system contains another striped group with an equivalent configuration.

     For example, if the striped group that you plan to remove has four equipment numbers, you must have another striped group that is in the ON state and has four equipment numbers.

4. Make sure that the file system that you plan to modify has a version 2A superblock. Use the command **samfsinfo** *filesystem-name*, where *filesystem-name* is the name of the file system.

   In the example, file system **hqfs1** uses a **version:2A** superblock:

   ```
   root@mds:~# /opt/SUNWsamfs/sbin/samfsinfo hqfs1
   samfsinfo: filesystem hqfs1 is mounted.
   name:     hqfs1       version:     2A
   time:     Tuesday, June 28, 2011  6:07:36 AM MDT
   feature:  Aligned Maps
   count:    4
   ...
   root@mds:~#
   ```

5. If the file system does not have a version 2A superblock, stop here. You cannot remove devices while this file system is mounted.

6. If you are removing devices from an Oracle HSM archiving file system, release all archived files from the disk device that you are removing. Use the command **samcmd release** *equipment-number*, where *equipment-number* is the equipment ordinal number that identifies the device in the **/etc/opt/SUNWsamfs/mcf** file.

   If the device is a striped group, provide the equipment number of any device in the group.

   The Oracle HSM software changes the state of the specified device to **noalloc** (no allocations) so that no new files are stored on it, and starts releasing previously archived files. Once the device contains no unarchived files, the software removes the device from the file system configuration and changes its state to **off**.

   In the example, we release files from device **104** in the archiving file system **hqfs1**:

   ```
   root@mds:~# samcmd release 104
   ```

7. If you are removing a device from an Oracle HSM non-archiving file system, move all remaining valid files off the disk device that you are removing. Use the command **samcmd remove** *equipment-number*, where *equipment-number* is the equipment ordinal number that identifies the device in the **/etc/opt/SUNWsamfs/mcf** file.

   The Oracle HSM software changes the state of the specified device to **noalloc** (no allocations) so that no new files are stored on it, and starts moving files that contain valid data to the remaining devices in the file system. When all files have been moved, the software removes the device from the file system configuration and changes its state to **off**.

   In the example, we move files off of device **104**:

```
root@mds:~# samcmd remove 104
```

8. Monitor the progress of the selected process, **samcmd remove** or **samcmd release**. Use the command **samcmd m** and/or watch the log file and **/var/opt/SUNWsamfs/trace/sam-shrink** file.

   The **release** process completes fairly quickly if all files have been archived, because it merely releases space associated with files that have been copied to archival media. Depending on the amount of data and the number of files, the **remove** process takes considerably longer because it must move files between disk devices.

```
root@mds:~# samcmd m
ty     eq  status       use state  ord  capacity      free    ra  part high low
ms     100 m----2-----  27% on            3.691G    2.628G    1M    16  80% 70%
 md     101              27% on       0  959.938M  703.188M
 md     102              28% on       1  899.938M  646.625M
 md     103              13% on       2  959.938M  834.250M
 md     104               0% noalloc  3  959.938M  959.938M
root@mds:~#
```

9. If you are using **samcmd release** and the target device does not enter the **off** state, there are unarchived files on the device. Wait for the archiver to run and archiving to complete. Then use the command **samcmd release** again. You can check on the progress of archiving by using the command **samcmd a**.

   The **release** process cannot free the disk space until unarchived files are archived.

```
root@mds:~# samcmd a
Archiver status samcmd     5.4 14:12:14 Mar  1 2014
samcmd on mds
sam-archiverd:  Waiting for resources
sam-arfind:  hqfs1  mounted at /hsm/hqfs1
Files waiting to start          4   schedule          2   archiving          2
root@mds:~#
```

10. If **samcmd release** fails because one or more unarchived files cannot be archived, move the unarchived files to another device. Use the command **samcmd remove** *equipment-number*, just as you would when removing devices from a non-archiving, standalone file system.

    In the example, we move files off of device **104**:

```
root@mds:~# samcmd remove 104
```

11. Once the device state has been changed to **off**, open the **/etc/opt/SUNWsamfs/mcf** file in a text editor, locate the file system, and update the equipment list to reflect the changes. Save the file and close the editor.

    In the example, **samcmd m** shows that **104** is **off**. So we use the **vi** editor to open the **mcf** file. We remove the entry for device **104** from the equipment list for the **hqfs1** file system and save our changes:

```
root@mds:~# samcmd m
ty      eq  status       use state  ord  capacity      free    ra  part high
low
ms     100  m----2-----  27% on           3.691G    2.628G    1M    16  80%
70%
 md     101              27% on       0  959.938M  703.188M
 md     102              28% on       1  899.938M  646.625M
 md     103              13% on       2  959.938M  834.250M
 md     104               0% off      3  959.938M  959.938M
root@mds:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment          Equipment  Equipment  Family     Device   Additional
# Identifier         Ordinal    Type       Set        State    Parameters
#-----------------   ---------  ---------  ---------  ------   ---------
hqfs1                100        ms         hqfs1      on
/dev/dsk/c1t3d0s3    101        md         hqfs1      on
/dev/dsk/c1t4d0s5    102        md         hqfs1      on
/dev/dsk/c1t3d0s7    103        md         hqfs1      on
:wq
root@mds:~#
```

12. Check the modified **mcf** file for errors by running the **sam-fsd** command, and correct any errors that are detected.

    The **sam-fsd** command will stop if it encounters an error. In the example, it reports no errors:

```
root@mds:~# sam-fsd
...
Would start sam-archiverd()
Would start sam-stagealld()
Would start sam-stagerd()
Would start sam-amld()
root@mds:~#
```

13. Tell the Oracle HSM software to re-read the **mcf** file and reconfigure itself accordingly:

```
root@mds:~# samd config
```

14. Stop here.

Managing Oracle HSM Shared File Systems

This section outlines the following tasks:

- mounting and unmounting shared file systems
- changing the host configuration of a shared file system
- switching from the active metadata server to a potential metadata server
- converting an unshared file system to a shared file system
- converting a shared file system to an unshared file system.

Mounting and Unmounting Shared File Systems

When you mount or unmount a shared file system, the order in which you mount or unmount the metadata server and the clients is important.

For failover purposes, the mount options should be the same on the metadata server and all potential metadata servers. For example, you can create a **samfs.cmd** file that contains the mount options and copy that file to all of the hosts.

For more information about mounting shared file systems, see the **mount_samfs** man page.

Mount a Shared File System

1. Log in to the Oracle HSM metadata server and client hosts as **root**.

   In the example, we log in to the metadata server host for the **shrfs1** file system, **mds**. Then we open a terminal window for each client, **clnt1** and **clnt2**. We use **ssh** (Secure Shell) to log in:

   ```
   root@mds:~# ssh root@clnt1
   Password:
   root@clnt1:~#

   root@mds:~# ssh root@clnt2
   Password:
   root@clnt2:~#
   ```

2. If the file system has an entry in the Solaris **/etc/vfstab** file, mount the shared file system on the metadata server host using the command **mount** *mountpoint*, where *mountpoint* is the mount point directory on the host's root file system.

   Always mount the file system on the metadata server host first, before mounting the file system on clients.

   In the example, the **shrfs1** file system has the following entry in the **/etc/vfstab** file:

   ```
   shrfs1   -         /hsm/shrfs1  samfs   -     no       shared
   ```

   So we can mount the file system by supplying only the mount point parameter:

   ```
   root@mds:~# mount /hsm/shrfs1
   root@mds:~#
   ```

3. If the file system does not have an entry in the Solaris **/etc/vfstab** file, mount the shared file system on the metadata server host using the command **mount -F samfs -o shared** *mountpoint*, where *mountpoint* is the mount point directory on the host's root file system.

   Always mount the file system on the metadata server host first, before mounting the file system on clients.

   In the example, the **shrfs1** file system has no entry in the **/etc/vfstab** file:

   ```
   root@mds:~# mount -F samfs -o shared /shrfs1
   root@mds:~#
   ```

4. If the file system has an entry in the Solaris **/etc/vfstab** file, mount the shared file system on each client host using the command **mount** *mountpoint*, where *mountpoint* is the mount point directory on the host's root file system.

   You can mount the file system on the client hosts in any order.

```
root@clnt1:~# mount /shrfs1
root@clnt1:~#

root@clnt2:~# mount /shrfs1
root@clnt2:~#
```

5. If the file system does not have an entry in the Solaris **/etc/vfstab** file, mount the shared file system on each client host using the command **mount -F samfs -o shared** *mountpoint*, where *mountpoint* is the mount point directory on the host's root file system.

   You can mount the file system on the client hosts in any order.

```
root@clnt1:~# mount -F samfs -o shared /shrfs1
root@clnt1:~#

root@clnt2:~# mount -F samfs -o shared /shrfs1
root@clnt2:~#
```

6. Stop here.

Unmount a Shared File System

1. Log in to the Oracle HSM metadata server and client hosts as **root**.

   In the example, we log in to the metadata server host for the **shrfs1** file system, **mds**. Then we open a terminal window for each client, **shrfs1-clnt1** and **shrfs1-client2** and use **ssh** (Secure Shell) to log in:

```
root@mds:~# ssh root@shrfs1-clnt1
Password:
root@clnt1:~#

root@mds:~# ssh root@shrfs1-client2
Password:
root@clnt2:~#
```

2. If the file system is shared through NFS or SAMBA, unshare the file system before you unmount it. On the metadata server, use the command **unshare** *mount-point*, where *mount-point* is the mount point directory of the Oracle HSM file system.

```
root@mds:~# unshare /shrfs1
root@mds:~#
```

3. Unmount the Oracle HSM shared file system from each client. Use the command **umount** *mount-point*, where *mount-point* is the mount point directory of the Oracle HSM file system.

   See the **umount_samfs** man page for further details. In the example, we unmount **/sharedqfs1** from our two clients, **shrfs1-clnt1** and **shrfs1-client2**:

```
root@clnt1:~# umount /shrfs1
root@clnt1:~# exit
root@mds:~#

root@clnt2:~# umount /shrfs1
root@clnt1:~# exit
root@mds:~#
```

4. Unmount the Oracle HSM shared file system from the metadata server. Use the command **umount -o await_clients=***interval mount-point*, where *mount-point*

is the mount point directory of the Oracle HSM file system and *interval* is the number of seconds by which the **-o await_clients** option delays execution.

When the **umount** command is issued on the metadata server of an Oracle HSM shared file system, the **-o await_clients** option makes **umount** wait the specified number of seconds so that clients have time to unmount the share. It has no effect if you unmount an unshared file system or issue the command on an Oracle HSM client. See the **umount_samfs** man page for further details.

In the example, we unmount the **/shrfs1** file system from the server, allowing **60** seconds for clients to unmount:

```
root@mds:~# umount -o await_clients=60 /shrfs1
root@mds:~#
```

5. Stop here.

Changing the Host Configuration of a Shared File System

This section provides instructions for configuring additional hosts as clients of a shared file system and for de-configuring existing clients. It covers the following tasks:

- configuring additional file system clients
- removing a host from a shared file system configuration
- configuring datamover clients for distributed tape I/O
- connecting tape drives using persistent bindings.

Configuring Additional File System Clients

There are three parts to the process of adding a client host to a shared file system:

- First, you add the host information to the shared file system configuration.
- Then you configure the shared file system on the host, using the procedure specific to the host operating system, Solaris or Linux.
- Finally, you mount the shared file system on the host, using the procedure specific to the host operating system, Solaris or Linux.

Add the Host Information to the Shared File System Configuration

1. Log in to the Oracle HSM metadata server as **root**.

   In the example, the metadata server host is **mds**:

   ```
   root@mds1:~#
   ```

2. Back up the file **/etc/opt/SUNWsamfs/hosts.***filesystem*, where *filesystem* is the name of the file system to which you are adding the client host.

   In the example, the Oracle HSM shared file system is named **shrfs1**:

   ```
   root@mds1:~# cp /etc/opt/SUNWsamfs/hosts.shrfs1
   /etc/opt/SUNWsamfs/hosts.shrfs1.bak
   ```

3. If the shared file system is mounted, run the command **samsharefs** *filesystem* from the active metadata server, redirecting output to a file, **/etc/opt/SUNWsamfs/hosts.***filesystem*, where *filesystem* is the name of the file system to which you are adding the client host.

   The **samsharefs** command displays the host configuration for an Oracle HSM shared file system. Redirecting the output to a file creates a new hosts file:

```
root@mds1:~# samsharefs shrfs1 > /etc/opt/SUNWsamfs/hosts.shrfs1
```

4. If the shared file system is not mounted, run the command **samsharefs -R** *filesystem* from an active or potential metadata server, redirecting output to the file **/etc/opt/SUNWsamfs/hosts.***filesystem*, where *filesystem* is the name of the file system to which you are adding the client host.

   The **samsharefs -R** command can only be run from an active or potential metadata server (see the **samsharefs** man page for more details). The **samsharefs** command displays the host configuration for an Oracle HSM shared file system. Redirecting the output to a file creates a new hosts file. In the example, we run the command from the metadata server **mds**:

   ```
   root@mds1:~# samsharefs -R shrfs1 > /etc/opt/SUNWsamfs/hosts.shrfs1
   ```

5. Open the newly created hosts file in a text editor.

   In the example, we use the **vi** editor. The host configuration includes the active metadata server, **mds1**, one client that is also a potential metadata server, **mds2**, and two other clients, **clnt1** and **clnt2**:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/hosts.shrfs1
   #                                        Server   On/  Additional
   #Host Name            Network Interface  Ordinal  Off  Parameters
   #------------------   -----------------  -------  ---  ----------
   mds1                  10.79.213.117      1        0    server
   mds2                  10.79.213.217      2        0
   clnt1                 10.79.213.133      0        0
   clnt2                 10.79.213.47       0        0
   ```

6. In the hosts file, add a line for the new client host, save the file, and close the editor.

   In the example, we add an entry for the host **clnt3**:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/hosts.shrfs1
   #                                        Server   On/  Additional
   #Host Name            Network Interface  Ordinal  Off  Parameters
   #------------------   -----------------  -------  ---  ----------
   mds1                  10.79.213.117      1        0    server
   mds2                  10.79.213.217      2        0
   clnt1                 10.79.213.133      0        0
   clnt2                 10.79.213.47       0        0
   clnt3                 10.79.213.49       0        0
   :wq
   root@mds1:~#
   ```

7. If the file system is mounted, update the file-system from the active metadata server. Use the command **samsharefs -u** *filesystem*, where *filesystem* is the name of the file system to which you are adding the client host.

   The **samsharefs** command re-reads the revised hosts file and updates the configuration:

   ```
   root@mds1:~# samsharefs -u shrfs1
   ```

8. If the file system is not mounted, update the file-system from an active or potential metadata server. Use the command **samsharefs -R -u** *filesystem*, where *filesystem* is the name of the file system to which you are adding the client host.

   The **samsharefs** command re-reads the revised hosts file and updates the configuration:

```
root@mds1:~# samsharefs -R -u shrfs1
```

9. If you are adding a Solaris host, configure the shared file system as a Solaris client.
10. If you are adding a Linux host, configure the shared file system as a Linux client.

Configure a Shared File System on a Solaris Client

1. On the shared file-system client, log in as **root**.

   In the example, the Oracle HSM shared file system is **shrfs1**, and the client host is **clnt1**:

   ```
   root@clnt1:~#
   ```

2. In a terminal window, retrieve the configuration information for the shared file system. Use the command **samfsconfig** *device-path*, where *device-path* is the location where the command should start to search for file-system disk devices (such as **/dev/dsk/*** or **/dev/zvol/dsk/rpool/***).

   ```
   root@clnt1:~# samfsconfig /dev/dsk/*
   ```

3. If the host has access to the metadata devices for the file system and is thus suitable for use as a potential metadata server, the **samfsconfig** output closely resembles the **mcf** file that you created on the file-system metadata server.

   In our example, host **clnt1** has access to the metadata devices (equipment type **mm**), so the command output shows the same equipment listed in the **mcf** file on the server, **mds**. Only the host-assigned device controller numbers differ:

   ```
   root@clnt1:~# samfsconfig /dev/dsk/*
   # Family Set 'shrfs1' Created Thu Feb 21 07:17:00 2013
   # Generation 0 Eq count 4 Eq meta count 1
   shrfs1             300         ma         shrfs1   -
   /dev/dsk/c1t0d0s0    301         mm         shrfs1   -
   /dev/dsk/c1t3d0s0    302         mr         shrfs1   -
   /dev/dsk/c1t3d0s1    303         mr         shrfs1   -
   ```

4. If the host does not have access to the metadata devices for the file system, the **samfsconfig** command cannot find the metadata devices and thus cannot fit the Oracle HSM devices that it discovers into the file-system configuration. The command output lists **Ordinal 0**—the metadata device—under **Missing Slices**, fails to include the line that identifies the file-system family set, and comments out the listings for the data devices.

   In our example, host **shrfs1-client2** has access to the data devices only. So the **samfsconfig** output looks like this:

   ```
   root@clnt2:~# samfsconfig /dev/dsk/*
   # Family Set 'shrfs1' Created Thu Feb 21 07:17:00 2013
   # Missing slices
   # Ordinal 0
   # /dev/dsk/c4t3d0s0    302         mr         shrfs1   -
   # /dev/dsk/c4t3d0s1    303         mr         shrfs1   -
   ```

5. Copy the entries for the shared file system from the **samfsconfig** output. Then, in a second window, open the **/etc/opt/SUNWsamfs/mcf** file in a text editor, and paste the copied entries into the file.

   In our first example, the host, **clnt1**, has access to the metadata devices for the file system, so the **mcf** file starts out looking like this:

   ```
   root@clnt1:~# vi /etc/opt/SUNWsamfs/mcf
   ```

```
# Equipment         Equipment  Equipment  Family     Device   Additional
# Identifier        Ordinal    Type       Set        State    Parameters
#---------------   ---------  ---------  ---------  ------   ---------------
shrfs1             300        ma         shrfs1     -
/dev/dsk/c1t0d0s0   301        mm         shrfs1     -
/dev/dsk/c1t3d0s0   302        mr         shrfs1     -
/dev/dsk/c1t3d0s1   303        mr         shrfs1     -
```

In the second example, the host, **shrfs1-client2**, does not have access to the metadata devices for the file system, so the **mcf** file starts out looking like this:

```
root@clnt2:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment         Equipment  Equipment  Family     Device   Additional
# Identifier        Ordinal    Type       Set        State    Parameters
#---------------   ---------  ---------  ---------  ------   ---------------
# /dev/dsk/c4t3d0s0   302         mr         shrfs1   -
# /dev/dsk/c4t3d0s1   303         mr         shrfs1   -
```

6. If the host has access to the metadata devices for the file system, add the **shared** parameter to the **Additional Parameters** field of the entry for the shared file system.

   In the first example, the host, **clnt1**, has access to the metadata:

```
root@clnt1:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment         Equipment  Equipment  Family     Device   Additional
# Identifier        Ordinal    Type       Set        State    Parameters
#---------------   ---------  ---------  ---------  ------   ---------------
shrfs1             300        ma         shrfs1     -        shared
/dev/dsk/c1t0d0s0   301        mm         shrfs1     -
/dev/dsk/c1t3d0s0   302        mr         shrfs1     -
/dev/dsk/c1t3d0s1   303        mr         shrfs1     -
```

7. If the host does not have access to the metadata devices for the file-system, add a line for the shared file system and include the **shared** parameter

```
root@clnt2:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment         Equipment  Equipment  Family     Device   Additional
# Identifier        Ordinal    Type       Set        State    Parameters
#---------------   ---------  ---------  ---------  ------   ---------------
shrfs1             300        ma         shrfs1     -        shared
# /dev/dsk/c4t3d0s0   302         mr         shrfs1   -
# /dev/dsk/c4t3d0s1   303         mr         shrfs1   -
```

8. If the host does not have access to the metadata devices for the file system, add a line for the metadata device. Set the **Equipment Identifier** field to **nodev** (*no device*) and set the remaining fields to exactly the same values as they have on the metadata server:

```
root@clnt2:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment         Equipment  Equipment  Family     Device   Additional
# Identifier        Ordinal    Type       Set        State    Parameters
#---------------   ---------  ---------  ---------  ------   ---------------
shrfs1             300        ma         shrfs1     on       shared
nodev               301         mm         shrfs1     on
# /dev/dsk/c4t3d0s0   302         mr         shrfs1     -
# /dev/dsk/c4t3d0s1   303         mr         shrfs1     -
```

9. If the host does not have access to the metadata devices for the file system, uncomment the entries for the data devices.

```
root@clnt2:~# vi /etc/opt/SUNWsamfs/mcf
```

```
# Equipment         Equipment  Equipment  Family     Device   Additional
# Identifier        Ordinal    Type       Set        State    Parameters
#----------------   ---------  ---------  ---------  ------   ---------------
shrfs1             300        ma         shrfs1     on       shared
nodev               301        mm         shrfs1     on
/dev/dsk/c4t3d0s0   302        mr         shrfs1     -
/dev/dsk/c4t3d0s1   303        mr         shrfs1     -
```

10. Make sure that the **Device State** field is set to **on** for all devices, save the **mcf** file, and close the editor.

    In our first example, the host, **clnt1**, has access to the metadata devices for the file system, so the **mcf** file ends up looking like this:

```
root@clnt1:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment         Equipment  Equipment  Family     Device   Additional
# Identifier        Ordinal    Type       Set        State    Parameters
#----------------   ---------  ---------  ---------  ------   ---------------
shrfs1             300        ma         shrfs1     on       shared
/dev/dsk/c1t0d0s0   301        mm         shrfs1     on
/dev/dsk/c1t3d0s0   302        mr         shrfs1     on
/dev/dsk/c1t3d0s1   303        mr         shrfs1     on
:wq
root@clnt1:~#
```

    In the second example, the host, **shrfs1-client2**, does not have access to the metadata devices for the file system, so the **mcf** file starts ends up like this:

```
root@clnt2:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment         Equipment  Equipment  Family     Device   Additional
# Identifier        Ordinal    Type       Set        State    Parameters
#----------------   ---------  ---------  ---------  ------   ---------------
shrfs1               300          ma         shrfs1     on       shared
nodev                 301          mm         shrfs1     on
/dev/dsk/c4t3d0s0     302          mr         shrfs1     on
/dev/dsk/c4t3d0s1     303          mr         shrfs1     on
:wq
root@clnt2:~#
```

11. Check the **mcf** file for errors by running the **sam-fsd** command, and correct any errors found.

    The **sam-fsd** is an initialization command that reads Oracle HSM configuration files. It will stop if it encounters an error. In the example, we check the **mcf** file on **clnt1** and find no errors:

```
root@clnt1:~# sam-fsd
...
Would start sam-archiverd()
Would start sam-stagealld()
Would start sam-stagerd()
Would start sam-amld()
root@mds:~#
```

12. Next mount the shared file system on the Solaris host.

Mount the Shared File System on a Solaris Host

1. On the shared file-system host, log in as **root**.

   In the example, the Oracle HSM shared file system is **shrfs1**, and the host is a client named **clnt1**:

```
root@clnt1:~#
```

2. Back up the operating system's **/etc/vfstab** file.

```
root@clnt1:~# cp /etc/vfstab /etc/vfstab.backup
```

3. Open the **/etc/vfstab** file in a text editor, and add a line for the shared file system.

   In the example, we open the file in the **vi** text editor and add a line for the **shrfs1** family set device:

```
root@clnt1:~# vi /etc/vfstab
#File
#Device    Device   Mount        System  fsck  Mount    Mount
#to Mount  to fsck  Point        Type    Pass  at Boot  Options
#--------  -------  --------     ------  ----  -------  --------------------
/devices   -        /devices     devfs   -     no       -
/proc      -        /proc        proc    -     no       -
...
shrfs1     -        /hsm/shrfs1  samfs   -     no
```

4. To mount the file system on the client as a shared file system, enter the **shared** option in the **Mount Options** column of the **vfstab** entry for the shared file system.

   If we wanted the current client to mount the shared file system **shrfs1** read-only, we would edit the **vfstab** entry as shown in the example below:

```
#File
#Device    Device   Mount        System  fsck  Mount    Mount
#to Mount  to fsck  Point        Type    Pass  at Boot  Options
#--------  -------  --------     ------  ----  -------  --------------------
/devices   -        /devices     devfs   -     no       -
/proc      -        /proc        proc    -     no       -
...
shrfs1     -        /hsm/shrfs1  samfs   -     no       shared
```

5. Add any other desired mount options using commas as separators, and make any other desired changes to the **/etc/vfstab** file. Then save the **/etc/vfstab** file.

   In the example, we add no additional mount options:

```
#File
#Device    Device   Mount        System  fsck  Mount    Mount
#to Mount  to fsck  Point        Type    Pass  at Boot  Options
#--------  -------  --------     ------  ----  -------  --------------------
/devices   -        /devices     devfs   -     no       -
/proc      -        /proc        proc    -     no       -
...
shrfs1     -        /hsm/shrfs1  samfs   -     no       shared
:wq
root@clnt1:~#
```

6. Create the mount point specified in the **/etc/vfstab** file, and set the access permissions for the mount point.

   The mount-point permissions must be the same as on the metadata server and on all other clients. Users must have execute (**x**) permission to change to the mount-point directory and access files in the mounted file system. In the example, we create the **/hsm/shrfs1** mount-point directory and set permissions to **755** (**-rwxr-xr-x**):

```
root@clnt1:~# mkdir /hsm
```

```
root@clnt1:~# mkdir /hsm/shrfs1
root@clnt1:~# chmod 755 /hsm/shrfs1
root@clnt1:~#
```

7. Mount the shared file system:

```
root@clnt1:~# mount /hsm/shrfs1
root@clnt1:~#
```

8. If you the new file system client is connected to tape devices and is to serve as a datamover, configure the client for distributed I/O.
9. Otherwise, stop here.

Configure the Shared File System on a Linux Client Host

1. On the Linux client, log in as **root**.

   In the example, the Oracle HSM shared file system is **shrfs1**, and the host is a Linux client named **clnt2L**:

```
[root@clnt2L ~]#
```

2. In a terminal window, retrieve the configuration information for the shared file system using the **samfsconfig** *device-path* command, where *device-path* is the location where the command should start to search for file-system disk devices (such as **/dev/***).

   Since Linux hosts do not have access to the metadata devices for the file system, the **samfsconfig** command cannot find the metadata devices and thus cannot fit the Oracle HSM devices that it discovers into the file-system configuration. The command output lists **Ordinal 0**—the metadata device—under **Missing Slices**, fails to include the line that identifies the file-system family set, and comments out the listings for the data devices.

   In our example, the **samfsconfig** output for Linux host **clnt2L** looks like this:

```
[root@clnt2L ~]# samfsconfig /dev/*
# Family Set 'shrfs1' Created Thu Feb 21 07:17:00 2013
#
# Missing slices
# Ordinal 0
# /dev/sda4            302         mr         shrfs1   -
# /dev/sda5            303         mr         shrfs1   -
```

3. Copy the entries for the shared file system from the **samfsconfig** output. Then, in a second window, open the **/etc/opt/SUNWsamfs/mcf** file in a text editor, and paste the copied entries into the file.

   In the example, the **mcf** file for the Linux the host, **clnt2L**, starts out looking like this:

```
[root@clnt2L ~]# vi /etc/opt/SUNWsamfs/mcf
# Equipment           Equipment  Equipment  Family     Device   Additional
# Identifier          Ordinal    Type       Set        State    Parameters
#------------------   ---------  ---------  ---------  ------   -------------
#/dev/sda4             302        mr         shrfs1     -
#/dev/sda5             303        mr         shrfs1     -
```

4. In the **mcf** file, insert a line for the shared file system, and include the **shared** parameter.

```
# Equipment           Equipment  Equipment  Family     Device   Additional
```

```
# Identifier          Ordinal    Type       Set        State    Parameters
#------------------   ---------  ---------  ---------  ------   -------------
shrfs1                300        ma         shrfs1     -        shared
#/dev/sda4             302        mr          shrfs1     -
#/dev/sda5             303        mr          shrfs1     -
```

5. In the **mcf** file, insert lines for the file system's metadata devices. Since the Linux host does not have access to metadata devices, set the **Equipment Identifier** field to **nodev** (*no device*) and then set the remaining fields to exactly the same values as they have on the metadata server:

```
# Equipment           Equipment  Equipment  Family     Device   Additional
# Identifier          Ordinal    Type       Set        State    Parameters
#------------------   ---------  ---------  ---------  ------   -------------
shrfs1                300         ma         shrfs1     on       shared
nodev                 301         mm         shrfs1     on
#/dev/sda4             302         mr          shrfs1     -
#/dev/sda5             303         mr          shrfs1     -
```

6. In the **mcf** file, uncomment the entries for the Linux data devices.

```
# Equipment           Equipment  Equipment  Family     Device   Additional
# Identifier          Ordinal    Type       Set        State    Parameters
#------------------   ---------  ---------  ---------  ------   -------------
shrfs1                300        ma         shrfs1     on       shared
nodev                 301        mm         shrfs1     on
/dev/sda4             302        mr         shrfs1     -
/dev/sda5             303        mr         shrfs1     -
```

7. Make sure that the **Device State** field is set to **on** for all devices, and save the **mcf** file.

```
# Equipment           Equipment  Equipment  Family     Device   Additional
# Identifier          Ordinal    Type       Set        State    Parameters
#------------------   ---------  ---------  ---------  ------   -------------
shrfs1                300        ma         shrfs1     on       shared
nodev                 301        mm         shrfs1     on
/dev/sda4             302        mr         shrfs1     on
/dev/sda5             303        mr         shrfs1     on
:wq
[root@clnt2L ~]#
```

8. Check the **mcf** file for errors by running the **sam-fsd** command, and correct any errors found.

   The **sam-fsd** is an initialization command that reads Oracle HSM configuration files. It will stop if it encounters an error. In the example, we check the **mcf** file on the Linux client, **clnt2L** and find no errors:

```
[root@clnt2L ~]# sam-fsd
...
Would start sam-archiverd()
Would start sam-stagealld()
Would start sam-stagerd()
Would start sam-amld()
root@mds:~#
```

9. Now, mount the shared file system on the Linux host.

Mount the Shared File System on a Linux Client Host

1. On the Linux client, log in as **root**.

In the example, the Oracle HSM shared file system is **shrfs1**, and the host is a Linux client named **clnt2L**:

```
[root@clnt2L ~]#
```

2. Back up the operating system's **/etc/fstab** file.

```
[root@clnt2L ~]# cp /etc/fstab /etc/fstab.backup
```

3. Open the **/etc/fstab** file in a text editor, and start a line for the shared file system.

   In the example, we use the **vi** text editor and add a line for the **shrfs1** family set device:

```
[root@clnt2L ~]# vi /etc/fstab
#File
#Device    Mount        System    Mount                    Dump      Pass
#to Mount  Point        Type      Options                  Frequency Number
#--------  -------      --------  -----------------------  --------- ------
...
/proc      /proc        proc      defaults
shrfs1     /hsm/shrfs1  samfs
```

4. In the fourth column of the file, add the mandatory **shared** mount option.

```
[root@clnt2L ~]# vi /etc/fstab
#File
#Device    Mount        System    Mount                    Dump      Pass
#to Mount  Point        Type      Options                  Frequency Number
#--------  -------      --------  -----------------------  --------- ------
...
/proc      /proc        proc      defaults
shrfs1     /hsm/shrfs1  samfs     shared
```

5. In the fourth column of the file, add any other desired mount options using commas as separators.

   Linux clients support the following additional mount options:

   - **rw**, **ro**
   - **retry**
   - **meta_timeo**
   - **rdlease**, **wrlease**, **aplease**
   - **minallocsz**, **maxallocsz**
   - **noauto**, **auto**

   In the example, we add the option **noauto**:

```
#File
#Device    Mount        System    Mount                    Dump      Pass
#to Mount  Point        Type      Options                  Frequency Number
#--------  -------      --------  -----------------------  --------- ------
...
/proc      /proc        proc      defaults
shrfs1     /hsm/shrfs1  samfs     shared,noauto
```

6. Enter zero (**0**) in each of the two remaining columns in the file. Then save the **/etc/fstab** file.

```
#File
```

```
#Device    Mount        System    Mount                     Dump      Pass
#to Mount  Point        Type      Options                   Frequency Number
#--------  -------      --------  ------------------------  --------- ------
...
/proc      /proc        proc      defaults
shrfs1     /hsm/shrfs1  samfs     shared,noauto             0         0
:wq
[root@clnt2L ~]#
```

7. Create the mount point specified in the **/etc/fstab** file, and set the access permissions for the mount point.

   The mount-point permissions must be the same as on the metadata server and on all other clients. Users must have execute (**x**) permission to change to the mount-point directory and access files in the mounted file system. In the example, we create the **/shrfs1** mount-point directory and set permissions to **755** (**-rwxr-xr-x**):

   ```
   [root@clnt2L ~]# mkdir /hsm
   [root@clnt2L ~]# mkdir /hsm/shrfs1
   [root@clnt2L ~]# chmod 755 /hsm//shrfs1
   ```

8. Mount the shared file system. Use the command **mount** *mountpoint*, where *mountpoint* is the mount-point directory specified in the **/etc/fstab** file.

   As the example shows, the **mount** command generates a warning. This is normal and can be ignored:

   ```
   [root@clnt2L ~]# mount /hsm/shrfs1
   Warning: loading SUNWqfs will taint the kernel: SMI license
   See http://www.tux.org/lkml/#export-tainted for information
   about tainted modules. Module SUNWqfs loaded with warnings
   ```

9. Stop here.

Removing a Host from a Shared File System Configuration

Removing a host from a shared file system is simply a matter of the removing it from the server configuration, as described below (to fully deconfigure the host, uninstall the software and the configuration files):

Remove the Host from the File System Hosts File

1. Log in to the Oracle HSM metadata server as **root**.

   In the example, the Oracle HSM shared file system is **shrfs1**, and the metadata server host is **mds1**:

   ```
   root@mds1:~#
   ```

2. Log in to each client as **root**, and unmount the shared file system.

   Remember that potential metadata servers are themselves clients. In the example, we have three clients: **clnt1**, **clnt2**, and **mds2**, a potential metadata server. For each client, we log in using **ssh**, unmount the file system **shrfs1**, and close the **ssh** session:

   ```
   root@mds1:~# ssh root@clnt1
   Password:
   root@clnt1:~# umount /hsm/shrfs1
   root@clnt1:~# exit
   root@mds1:~# ssh root@clnt2
   ```

```
Password:
root@clnt2:~# umount /hsm/shrfs1
root@clnt2:~# exit
root@mds1:~# ssh root@mds2
Password:
root@mds2:~# umount /hsm/shrfs1
root@mds1:~# exit
root@mds1:~#
```

3. On the metadata server, unmount the shared file system.

```
root@mds1:~# umount /hsm/shrfs1
root@mds1:~#
```

4. On the metadata server, rename the file **/etc/opt/SUNWsamfs/hosts.***filesystem* to **/etc/opt/SUNWsamfs/hosts.***filesystem***.bak**, where *filesystem* is the name of the file system from which you are removing the client host.

```
root@mds1:~# mv /etc/opt/SUNWsamfs/hosts.shrfs1
/etc/opt/SUNWsamfs/hosts.shrfs1.bak
root@mds1:~#
```

5. Capture the current shared file system host configuration top a file. From the metadata server, run the command **samsharefs -R** *filesystem*, redirecting the output to the file **/etc/opt/SUNWsamfs/hosts.***filesystem*, where *filesystem* is the name of the file system to which you are adding the client host.

   The **samsharefs** command displays the host configuration for the specified Oracle HSM shared file system. Redirecting the output to a file creates a new hosts file. In the example, we run the command from the metadata server **mds**:

```
root@mds1:~# samsharefs -R shrfs1 > /etc/opt/SUNWsamfs/hosts.shrfs1
root@mds1:~#
```

6. Open the newly created hosts file in a text editor.

   In the example, we use the **vi** editor. We need to remove the client **shrfs1-clnt2**:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/hosts.shrfs1
#                                       Server   On/  Additional
#Host Name            Network Interface  Ordinal  Off  Parameters
#------------------   -----------------  -------  ---  ----------
mds                   10.79.213.117      1        0    server
mds2                  10.79.213.217      2        0
clnt1                 10.79.213.133      0        0
clnt2                 10.79.213.47       0        0
```

7. In the hosts file, delete the line that corresponds to the client host that you need to remove. Then save the file, and close the editor.

   In the example, we delete the entry for the host **clnt2**:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/hosts.shrfs1
#                                       Server   On/  Additional
#Host Name            Network Interface  Ordinal  Off  Parameters
#------------------   -----------------  -------  ---  ----------
mds                   10.79.213.117      1        0    server
mds2                  10.79.213.217      2        0
clnt1                 10.79.213.133      0        0
:wq
root@mds1:~#
```

8. Update the file-system with the revised hosts file. From the metadata server, use the command **samsharefs -R -u** *filesystem*, where *filesystem* is the name of the file system from which you are removing the client host.

```
root@mds1:~# samsharefs -u shrfs1
```

9. On the metadata server host, mount the shared file system.

   In the examples, the **/etc/vfstab** file contains an entry for the **shrfs1** file system, so we use the simple mounting syntax (see the **mount_samfs** man page for full information):

```
root@mds1:~# mount /hsm/shrfs1
```

10. On the each client host, mount the shared file system.

    Remember that potential metadata servers are themselves clients. In the example, we now have two clients: **clnt1** and **mds2**, a potential metadata server. For each client, we log in using **ssh**, mount the file system **shrfs1**, and close the **ssh** session:

```
root@mds1:~# ssh root@mds2
Password:
root@mds2:~# mount /hsm/shrfs1
root@mds2:~# exit
root@mds1:~# ssh root@clnt1
Password:
root@clnt1:~# mount /hsm/shrfs1
root@clnt1:~# exit
root@mds1:~#
```

11. Stop here.

Configuring Datamover Clients for Distributed Tape I/O

Starting with Oracle HSM Release 6.1.4, any client of a shared archiving file system that runs on Solaris 11 or higher can attach tape drives and carry out tape I/O on behalf of the file system. Distributing tape I/O across these *datamover* hosts greatly reduces server overhead, improves file-system performance, and allows significantly more flexibility when scaling Oracle HSM implementations. As your archiving needs increase, you now have the option of either replacing Oracle HSM metadata servers with more powerful systems (vertical scaling) or spreading the load across more clients (horizontal scaling).

Configure the Datamover Client To configure a client for distributed tape I/O, proceed as follows:

1. Connect all devices that will be used for distributed I/O to the client.
2. If you have not already done so, connect the tape devices using persistent bindings. Then return here.
3. Log in to the shared archiving file system's metadata server as **root**.

   In the example, the host name is **mds**:

```
root@mds:~#
```

4. Make sure that the metadata server is running Oracle HSM Solaris 11 or higher.

```
root@mds:~# uname -r
5.11
root@mds:~#
```

5. Make sure that all clients that serve as datamovers are running Oracle HSM Solaris 11 or higher.

   In the example, we open a terminal window for each client host, **clnt1** and **clnt2**, and log in remotely using **ssh**. The log-in banner displays the Solaris version:

```
root@mds:~# ssh root@clnt1
...
Oracle Corporation      SunOS 5.11      11.1    September 2013
root@clnt1:~#

root@mds:~# ssh root@clnt2
...
Oracle Corporation      SunOS 5.11      11.1    September 2013
root@clnt2:~#
```

6. On the metadata server, copy the file **/opt/SUNWsamfs/examples/defaults.conf** to the directory **/etc/opt/SUNWsamfs/**.

```
root@mds:~# cp /opt/SUNWsamfs/examples/defaults.conf /etc/opt/SUNWsamfs/
root@mds:~#
```

7. On the metadata server, open the file **/etc/opt/SUNWsamfs/defaults.conf** in a text editor.

   By default, **distio** is **off** (disabled). In the example, we open the file in the **vi** editor:

```
root@mds:~#  vi /etc/opt/SUNWsamfs/defaults.conf
# These are the defaults.  To change the default behavior, uncomment the
# appropriate line (remove the '#' character from the beginning of the line)
...
#distio = on
```

8. In the **defaults.conf** file, enable distributed I/O by uncommenting the line **distio = on**.

   By default, **distio** is **off** (disabled).

```
root@mds:~#  vi /etc/opt/SUNWsamfs/defaults.conf
# These are the defaults.  To change the default behavior, uncomment the
# appropriate line (remove the '#' character from the beginning of the line)
...
distio = on
```

9. Next, identify each device type that should participate in distributed I/O by adding a line to the **defaults.conf** file of the form *dev*_**distio = on**, where *dev* is one of the equipment type identifiers listed in Appendix A, "Glossary of Equipment Types". To exclude device type *dev* from distributed I/O, add the line *dev*_**distio = off**.

   By default, Oracle StorageTek T10000 drives and LTO drives are allowed to participate in distributed I/O, while all other types are excluded. In the example, we plan to use LTO drives, so we do not need to make any edits:

```
root@mds:~#  vi /etc/opt/SUNWsamfs/defaults.conf
# These are the defaults.  To change the default behavior, uncomment the
# appropriate line (remove the '#' character from the beginning of the line)
...
distio = on
```

10. Next, identify each device type that should not participate in distributed I/O by adding a line to the **defaults.conf** file of the form *dev*_**distio = off**, where *dev* is one of the equipment type identifiers listed in Appendix A, "Glossary of Equipment Types".

    In the example, we do not want to use Oracle StorageTek T10000 drives with distributed I/O. Since, by default, Oracle StorageTek T10000 drives are allowed to participate in distributed I/O, we have to add the line **ti_distio = off** to the **defaults.conf** file:

```
root@mds:~#  vi /etc/opt/SUNWsamfs/defaults.conf
# These are the defaults.  To change the default behavior, uncomment the
# appropriate line (remove the '#' character from the beginning of the line)
...
distio = on
```

11. Save the **defaults.conf** file, and close the editor.

```
root@mds:~#  vi /etc/opt/SUNWsamfs/defaults.conf
# These are the defaults.  To change the default behavior, uncomment the
# appropriate line (remove the '#' character from the beginning of the line)
...
distio = on
ti_distio = off
:wq
root@mds:~#
```

12. On each client that will serve as a datamover, edit the **defaults.conf** file so that it matches the file on the server.

    In the example, we use Secure Shell (**ssh**) to remotely log in to client **clnt1** and edit the **defaults.conf** file:

```
root@mds:~# ssh root@clnt1
Password:
root@clnt1:~# vi /etc/opt/SUNWsamfs/defaults.conf
# These are the defaults.  To change the default behavior, uncomment the
# appropriate line (remove the '#' character from the beginning of the line)
# and change the value.
...
distio = on
ti_distio = off
:wq
root@clnt1:~# exit
root@mds:~#
```

13. On each client that will serve as a datamover, open the **/etc/opt/SUNWsamfs/mcf** file in a text editor. Add all of the tape devices that the metadata server is using for distributed tape I/O. Make sure that the device order and equipment numbers are identical to those in the **mcf** file on the metadata server.

    In the example, we edit the **mcf** file on client **clnt1** using **vi**:

```
root@clnt1:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment              Equipment Equipment Family      Device Additional
# Identifier             Ordinal   Type      Set         State  Parameters
#---------------------- --------- --------- ----------  ------ -------------
shrfs1                   800       ms        samsharefs  on
...
# Archival storage for copies:
/dev/rmt/60cbn           901       li                    on
/dev/rmt/61cbn           902       li                    on
```

```
/dev/rmt/62cbn           903       li                  on
/dev/rmt/63cbn           904       li                  on
```

14. If the tape library listed in the **/etc/opt/SUNWsamfs/mcf** file on the metadata server is configured on the client that will serve as a datamover, specify the library family set as the family set name for the tape devices that are being used for distributed tape I/O. Save the file, and close the editor.

    In the example, the library is configured on the host, **clnt1**, so we use the family set name **lib1** for the tape devices:

```
root@clnt1:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment              Equipment Equipment Family      Device Additional
# Identifier             Ordinal   Type      Set         State  Parameters
#----------------------- --------- --------- ----------  ------ -------------
shrfs1                   800       ms        samsharefs  on
...
# Archival storage for copies:
/dev/scsi/changer/c1t0d5 900       rb        lib1        on
/dev/rmt/60cbn           901       li        lib1        on
/dev/rmt/61cbn           902       li        lib1        on
/dev/rmt/62cbn           903       li        lib1        on
/dev/rmt/63cbn           904       li        lib1        on
:wq
root@clnt1:~#
```

15. If the tape library listed in the **/etc/opt/SUNWsamfs/mcf** file on the metadata server is *not* configured on the client that will serve as a datamover, use a hyphen (**-**) as the family set name for the tape devices that are being used for distributed tape I/O. Save the file, and close the editor.

    In the example, the library is not configured on the host, **clnt1**, so we use the hyphen as the family set name for the drives:

```
root@clnt2:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment              Equipment Equipment Family      Device Additional
# Identifier             Ordinal   Type      Set         State  Parameters
#----------------------- --------- --------- ----------  ------ -------------
shrfs1                   800       ms        samsharefs  on
...
# Archival storage for copies:
/dev/rmt/60cbn           901       ti        -           on
/dev/rmt/61cbn           902       ti        -           on
/dev/rmt/62cbn           903       ti        -           on
/dev/rmt/63cbn           904       ti        -           on
:wq
root@clnt2:~#
```

16. If you need to enable or disable distributed tape I/O for particular archive set copies, open the server's **/etc/opt/SUNWsamfs/archiver.cmd** file in a text editor and add the **-distio** parameter to the copy directive. Set **-distio on** to enable or **off** to disable distributed I/O. Save the file, and close the editor.

    In the example, we use the vi editor to turn distributed I/O **off** for copy **1** and **on** for copy **2**:

```
root@mds:~# vi /etc/opt/SUNWsamfs/archiver.cmd
# archiver.cmd
# Generated by config api Mon Nov 22 14:31:39 2013
...
#
```

```
# Copy Parameters Directives
params
allsets -sort path -offline_copy stageahead
allsets.1 -startage 10m -startsize 500M -startcount 500000 -distio off
allsets.2 -startage 24h -startsize 20G  -startcount 500000 -distio on
:wq
root@mds:~#
```

17. On each host, check the **mcf** file for errors by running the **sam-fsd** command, and correct any errors found.

    The **sam-fsd** is an initialization command that reads Oracle HSM configuration files. It will stop if it encounters an error. In the example, we check the **mcf** file on clients **clnt1** and **clnt2**:

```
root@mds:~# ssh root@clnt1
Password:
root@clnt1:~# sam-fsd
...
root@clnt1:~# exit
root@mds:~# ssh root@clnt2
Password:
root@clnt2:~# sam-fsd
...
root@clnt2:~# exit
root@mds:~#
```

18. On the server, tell the Oracle HSM software to read the modified configuration files and reconfigure itself accordingly. Use the command **samd config**, and correct any errors found.

    In the example, we run the **samd config** command on the server, **mds**:

```
root@mds:~# samd config
Configuring SAM-FS ...
root@mds:~#
```

19. Stop here.

Connecting Tape Drives Using Persistent Bindings

When you add a host that serves as either a potential metadata server or a distributed I/O datamover client, you must configure removable media devices using persistent bindings. The Solaris operating system attaches drives to the system device tree in the order in which it discovers the devices at startup. This order may or may not reflect the order in which devices are discovered by other file system hosts or the order in which they are physically installed in the tape library. So you need to bind the devices to the new host in the same way that they are bound to the other hosts and in the same order in which they are installed in the removable media library.

The procedures below outline the required steps (for full information, see the **devfsadm** and **devlinks** man pages and the administration documentation for your version of the Solaris operating system):

- If you have moved, added, or removed drives in a library or replaced or reconfigured the library associated with an archiving Oracle HSM shared file system, update persistent bindings to reflect the changes.
- If you are adding a new metadata server or datamover client to an archiving Oracle HSM shared file system, persistently bind the new file system host to the removable media devices

Update Persistent Bindings to Reflect Changes to the Hardware Configuration

1. Log in to the active metadata server host as **root**.

```
root@mds1:~#
```

2. Create a new drive-mapping file as described in "Determining the Order in Which Drives are Installed in the Library" on page 5-11.

   In the example, the **device-mappings.txt** file looks like this:

```
root@mds1:~# vi /root/device-mappings.txt
LIBRARY SOLARIS          SOLARIS
DEVICE  LOGICAL          PHYSICAL
NUMBER  DEVICE           DEVICE
------- -------------    ------------------------------------------------------
   2    /dev/rmt/0cbn -> ../../devices/pci@8.../st@w500104f00093c438,0:cbn
   1    /dev/rmt/1cbn -> ../../devices/pci@8.../st@w500104f0008120fe,0:cbn
   3    /dev/rmt/2cbn -> ../../devices/pci@8.../st@w500104f000c086e1,0:cbn
   4    /dev/rmt/3cbn ->  ../../devices/pci@8.../st@w500104f000b6d98d,0:cbn
```

3. Open the **/etc/devlink.tab** file in a text editor.

   In the example, we use the **vi** editor:

```
root@mds1:~# vi /etc/devlink.tab
# Copyright (c) 1993, 2011, Oracle and/or its affiliates. All rights reserved.
# This is the table used by devlinks
# Each entry should have 2 fields; but may have 3.  Fields are separated
# by single tab ('\t') characters.
...
```

4. Using the **device-mappings.txt** file as a guide, remap a starting node in the Solaris tape device tree to the first drive in the library. In the **/etc/devlink.tab** file, add a line of the form **type=ddi_byte:tape; addr=***device_address***,0; rmt/node-number\M0**, where *device_address* is the physical address of the device and *node-number* is a position in the Solaris device tree that is high enough to avoid conflicts with any devices that Solaris configures automatically (Solaris starts from node **0**).

   In the example, we note that the device address for the first device in the library, **1**, is **w500104f0008120fe** and see that the device is currently attached to the host at **rmt/1**:

```
root@mds1:~# cat /root/device-mappings.txt
LIBRARY SOLARIS          SOLARIS
DEVICE  LOGICAL          PHYSICAL
NUMBER  DEVICE           DEVICE
------- -------------    ------------------------------------------------------
   2    /dev/rmt/0cbn -> ../../devices/pci@8.../st@w500104f00093c438,0:cbn
   1    /dev/rmt/1cbn -> ../../devices/pci@8.../st@w500104f0008120fe,0:cbn
   3    /dev/rmt/2cbn -> ../../devices/pci@8.../st@w500104f000c086e1,0:cbn
   4    /dev/rmt/3cbn -> ../../devices/pci@8.../st@w500104f000b6d98d,0:cbn
```

   So we create a line in **/etc/devlink.tab** that remaps **rmt/60** to the number **1** drive in the library, **w500104f0008120fe**:

```
root@mds1:~# vi /etc/devlink.tab
# Copyright (c) 1993, 2011, Oracle and/or its affiliates. All rights reserved.
...
type=ddi_byte:tape;addr=w500104f0008120fe,0;    rmt/60\M0
:w
```

5. Continue to add lines to the **/etc/devlink.tab** file for each tape device that is assigned for Oracle HSM archiving, so that the drive order in the device tree on the metadata server matches the installation order on the library. Save the file, and close the editor.

   In the example, we note the order and addresses of the three remaining devices—library drive **2** at **w500104f00093c438**, library drive **3** at **w500104f000c086e1**, and library drive **4** at **w500104f000c086e1**:

```
root@mds1:~# cat /root/device-mappings.txt
...
   2    /dev/rmt/0cbn -> ../../devices/pci@8\.../st@w500104f00093c438,0:cbn
   1    /dev/rmt/1cbn -> ../../devices/pci@8\.../st@w500104f0008120fe,0:cbn
   3    /dev/rmt/2cbn -> ../../devices/pci@8\.../st@w500104f000c086e1,0:cbn
   4    /dev/rmt/3cbn -> ../../devices/pci@8\.../st@w500104f000b6d98d,0:cbn
```

   Then we map the device addresses to next three Solaris device nodes, maintaining the same order as in the library:

```
root@mds1:~# vi /etc/devlink.tab
...
type=ddi_byte:tape;addr=w500104f0008120fe,0;    rmt/60\M0
type=ddi_byte:tape;addr=w500104f00093c438,0;    rmt/61\M0
type=ddi_byte:tape;addr=w500104f000c086e1,0;    rmt/62\M0
type=ddi_byte:tape;addr=w500104f000b6d98d,0;    rmt/63\M0
:wq
root@mds1:~#
```

6. Delete all existing links to the tape devices in **/dev/rmt**.

```
root@mds1:~# rm /dev/rmt/*
```

7. Create new, persistent tape-device links from the entries in the **/etc/devlink.tab** file. Use the command **devfsadm -c tape**.

   Each time that the **devfsadm** command runs, it creates new tape device links for devices specified in the **/etc/devlink.tab** file using the configuration specified by the file. The **-c tape** option restricts the command to creating new links for tape-class devices only:

```
root@mds1:~# devfsadm -c tape
```

8. Repeat the operation on each potential metadata server and datamover in the shared file system configuration. In each case, add the same lines to the **/etc/devlink.tab** file, delete the links in **/dev/rmt**, and run **devfsadm -c tape**.

   In the example, The file system configuration includes one potential metadata server, **mds2**, and one client, **clnt1**. We use **ssh** to log in to each host in turn, and configure the same four logical devices: **rmt/60\M0**, **rmt/61\M0**, **rmt/62\M0**, and **rmt/63\M0**.

```
root@mds1:~# ssh root@mds2
Password:
root@mds2:~# vi /etc/devlink.tab
...
type=ddi_byte:tape;addr=w500104f0008120fe,0;    rmt/60\M0
type=ddi_byte:tape;addr=w500104f00093c438,0;    rmt/61\M0
type=ddi_byte:tape;addr=w500104f000c086e1,0;    rmt/62\M0
type=ddi_byte:tape;addr=w500104f000b6d98d,0;    rmt/63\M0
:wq
root@mds2:~# rm /dev/rmt/*
root@mds2:~# devfsadm -c tape
```

```
root@mds2:~# exit
root@mds1:~# ssh root@clnt1
Password:
root@clnt1:~# vi /etc/devlink.tab
...
type=ddi_byte:tape;addr=w500104f0008120fe,0;    rmt/60\M0
type=ddi_byte:tape;addr=w500104f00093c438,0;    rmt/61\M0
type=ddi_byte:tape;addr=w500104f000c086e1,0;    rmt/62\M0
type=ddi_byte:tape;addr=w500104f000b6d98d,0;    rmt/63\M0
:wq
root@clnt1:~# rm /dev/rmt/*
root@clnt1:~# devfsadm -c tape
root@clnt1:~# exit
root@mds1:~#
```

9. Return to the task that you were performing: "Configuring Datamover Clients for Distributed Tape I/O" on page 3-55 or "Configuring Additional File System Clients" on page 3-44.

Persistently Bind a New File System Host to Removable Media Devices

1. Log in to the host as **root**.

```
root@mds1~#
```

2. If the physical order of the drives in the media library has changed since the existing file-system hosts were configured, create a new mapping file as described in "Determining the Order in Which Drives are Installed in the Library" on page 5-11.

   In the example, the **device-mappings.txt** file looks like this:

```
root@mds1~# cat /root/device-mappings.txt
LIBRARY SOLARIS          SOLARIS
DEVICE  LOGICAL          PHYSICAL
NUMBER  DEVICE           DEVICE
------- -------------    --------------------------------------------------
   2    /dev/rmt/0cbn -> ../../devices/pci@8.../st@w500104f00093c438,0:cbn
   1    /dev/rmt/1cbn -> ../../devices/pci@8.../st@w500104f0008120fe,0:cbn
   3    /dev/rmt/2cbn -> ../../devices/pci@8.../st@w500104f000c086e1,0:cbn
   4    /dev/rmt/3cbn -> ../../devices/pci@8.../st@w500104f000b6d98d,0:cbn
```

3. Open the **/etc/devlink.tab** file in a test editor.

   In the example, we use the **vi** editor:

```
root@mds1~# vi /etc/devlink.tab
# Copyright (c) 1993, 2011, Oracle and/or its affiliates. All rights reserved.
# This is the table used by devlinks
# Each entry should have 2 fields; but may have 3.  Fields are separated
# by single tab ('\t') characters.
...
```

4. Using the **device-mappings.txt** file as a guide, remap a starting node in the Solaris tape device tree, **rmt/***node-number*, to the first drive in the library. Add a line to the **/etc/devlink.tab** file of the form **type=ddi_byte:tape; addr=***device_address***,0; rmt/***node-number***\M0**, where: *device_address* is the physical address of the device and *node-number* is the device's position in the Solaris device tree. Choose a node number that is high enough to avoid conflicts with any devices that Solaris configures automatically (Solaris starts from node **0**).

In the example, we note that the device address for the first device in the library, **1**, is **w500104f0008120fe** and see that the device is currently attached to the host at **rmt/1**:

```
root@mds1~# cat /root/device-mappings.txt
LIBRARY SOLARIS          SOLARIS
DEVICE  LOGICAL          PHYSICAL
NUMBER  DEVICE           DEVICE
------- -------------    ----------------------------------------------------
   2    /dev/rmt/0cbn -> ../../devices/pci@8.../st@w500104f00093c438,0:cbn
   1    /dev/rmt/1cbn -> ../../devices/pci@8.../st@w500104f0008120fe,0:cbn
   3    /dev/rmt/2cbn -> ../../devices/pci@8.../st@w500104f000c086e1,0:cbn
   4    /dev/rmt/3cbn -> ../../devices/pci@8.../st@w500104f000b6d98d,0:cbn
```

So we create a line in **/etc/devlink.tab** that remaps **rmt/60** to the number **1** drive in the library, **w500104f0008120fe**:

```
root@mds1~# vi /etc/devlink.tab
# Copyright (c) 1993, 2011, Oracle and/or its affiliates. All rights reserved.
...
type=ddi_byte:tape;addr=w500104f0008120fe,0;    rmt/60\M0
:w
```

5. Continue to add lines to the **/etc/devlink.tab** file for each tape device that is assigned for Oracle HSM archiving, so that the drive order in the device tree on the metadata server matches the installation order on the library. Save the file.

   In the example, we note the order and addresses of the three remaining devices—library drive **2** at **w500104f00093c438**, library drive **3** at **w500104f000c086e1**, and library drive **4** at **w500104f000c086e1**:

   ```
   root@mds1~# cat /root/device-mappings.txt
   ...
      2    /dev/rmt/0cbn -> ../../devices/pci@8.../st@w500104f00093c438,0:cbn
      1    /dev/rmt/1cbn -> ../../devices/pci@8.../st@w500104f0008120fe,0:cbn
      3    /dev/rmt/2cbn -> ../../devices/pci@8.../st@w500104f000c086e1,0:cbn
      4    /dev/rmt/3cbn -> ../../devices/pci@8.../st@w500104f000b6d98d,0:cbn
   ```

   Then we map the device addresses to the next three Solaris device nodes, maintaining the same order as in the library:

   ```
   root@mds1~# vi /etc/devlink.tab
   ...
   type=ddi_byte:tape;addr=w500104f0008120fe,0;    rmt/60\M0
   type=ddi_byte:tape;addr=w500104f00093c438,0;    rmt/61\M0
   type=ddi_byte:tape;addr=w500104f000c086e1,0;    rmt/62\M0
   type=ddi_byte:tape;addr=w500104f000b6d98d,0;    rmt/63\M0
   :wq
   root@mds1~#
   ```

6. Delete all existing links to the tape devices in **/dev/rmt**.

   ```
   root@mds1~# rm /dev/rmt/*
   ```

7. Create new, persistent tape-device links from the entries in the **/etc/devlink.tab** file. Use the command **devfsadm -c tape**.

   Each time that the **devfsadm** command runs, it creates new tape device links for devices specified in the **/etc/devlink.tab** file using the configuration specified by the file. The **-c tape** option restricts the command to creating new links for tape-class devices only:

```
root@mds1~# devfsadm -c tape
```

8. On each potential metadata server and datamover in the shared file system configuration, add the same lines to the **/etc/devlink.tab** file, delete the links in **/dev/rmt**, and run **devfsadm -c tape**.

   In the example, we use **ssh** to log in to the potential metadata server host **mds2** and the client host **clnt1**. We then configure the same four logical devices, **rmt/60\M0**, **rmt/61\M0**, **rmt/62\M0**, and **rmt/63\M0**, on each:

```
root@mds1~# ssh root@mds2
Password:
root@mds2:~# vi /etc/devlink.tab
...
type=ddi_byte:tape;addr=w500104f0008120fe,0;    rmt/60\M0
type=ddi_byte:tape;addr=w500104f00093c438,0;    rmt/61\M0
type=ddi_byte:tape;addr=w500104f000c086e1,0;    rmt/62\M0
type=ddi_byte:tape;addr=w500104f000b6d98d,0;    rmt/63\M0
:wq
root@mds2:~# rm /dev/rmt/*
root@mds2:~# devfsadm -c tape
root@mds2:~# exit
root@mds1~# ssh root@clnt1
Password:
root@clnt1:~# vi /etc/devlink.tab
...
type=ddi_byte:tape;addr=w500104f0008120fe,0;    rmt/60\M0
type=ddi_byte:tape;addr=w500104f00093c438,0;    rmt/61\M0
type=ddi_byte:tape;addr=w500104f000c086e1,0;    rmt/62\M0
type=ddi_byte:tape;addr=w500104f000b6d98d,0;    rmt/63\M0
:wq
root@clnt1:~# rm /dev/rmt/*
root@clnt1:~# devfsadm -c tape
root@clnt1:~# exit
root@mds1~#
```

9. Return to the task that you were performing: "Configuring Datamover Clients for Distributed Tape I/O" on page 3-55 or "Configuring Additional File System Clients" on page 3-44.

Switching from the Active Metadata Server to a Potential Metadata Server

The procedures in this section move the metadata service for the file system from the current host (the active metadata server) to a standby host (the potential metadata server). How you proceed depends on the health of the current host:

- If the current, active host has failed, you move the metadata service from the faulty host to the healthy standby host, even if file systems are still mounted.
- If the current, active host is healthy, you unmount the file systems before you move the metadata service from the current to the standby host.

Activate a Potential Metadata Server to Replace a Faulty Active Metadata Server

This procedure lets you move the metadata service off of an active metadata server host that has stopped functioning. It activates a potential metadata server, even if a file system is still mounted. Proceed as follows:

Caution: Never activate a potential metadata server until you have stopped, disabled, or disconnected the faulty metadata server!

To activate a potential server when a file system is mounted and the active metadata server is down, you have to invoke the **samsharefs** command with the **-R** option, which acts on raw devices rather than on file-system interfaces. So, if you activate a potential metadata server while the faulty server is still connected to the devices, the faulty server can corrupt the file system.

1. If the active metadata server is faulty, make sure that it cannot access the metadata devices before you do anything else. Power the affected host off, halt the host, or disconnect the failed host from the metadata devices.
2. Wait at least until the maximum lease time has run out, so that all client read, write, and append leases can expire.
3. Log in to a potential metadata server as **root**.

   In the example, we log in to the potential metadata server **mds2**:

   ```
   root@mds2:~#
   ```

4. Activate the potential metadata server. From the potential metadata server, issue the command **samsharefs -R -s** *server file-system*, where *server* is the host name of the potential metadata server and *file-system* is the name of the Oracle HSM shared file system.

   In the example, the file system name is **shrfs1**:

   ```
   root@mds2:~# samsharefs -R -s mds2 shrfs1
   ```

5. If you need to check the integrity of a file system and repair possible problems, unmount the file system now using the procedure "Unmount a Shared File System" on page 3-43.
6. If you have unmounted the file system, perform the file system check. Use the command **samfsck -F** *file-system*, where **-F** specifies repair of errors and where *file-system* is the name of the file system.

   In the example, we check and repair the file system name is **shrfs1**:

   ```
   root@mds2:~# samfsck -F shrfs1
   samfsck: Configuring file system
   samfsck: Enabling the sam-fsd service.
   name:     shrfs1       version:     2A
   ...
   root@mds2:~#
   ```

7. Stop here.

Activate a Potential Metadata Server to Replace a Healthy Active Metadata Server

You can move the metadata service off of a healthy, active metadata server host and on to a newly activated potential metadata server when required. For example, you might transfer metadata services to an alternate host to keep file systems available while you upgrade or replace the original server host or some of its components. Proceed as follows:

1. Log in to both the active and potential metadata servers as **root**.

In the example, we log in to the active metadata server, **mds1**. Then, in a second terminal window, we use secure shell (**ssh**) to log in to the potential metadata server **mds2**:

```
root@mds1~# ssh root@mds2
Password:
root@mds2:~#
```

2. If the active metadata server mounts an Oracle HSM archiving file system, finish active archiving and staging jobs and stop any new activity before proceeding further.
3. If you use a **crontab** entry to run the recycler process, remove the entry and make sure that the recycler is not currently running.
4. Activate the potential metadata server. From the potential metadata server, issue the command **samsharefs -s** *server file-system*, where *server* is the host name of the potential metadata server and *file-system* is the name of the Oracle HSM shared file system.

   In the example, the potential metadata server is **mds2** and the file system name is **shrfs1**:

   ```
   root@mds2:~# samsharefs -s mds2 shrfs1
   ```

5. Load the configuration files and start Oracle HSM processes on the potential metadata server. Use the command **samd config**.

   For archiving shared file systems, the **samd config** command restarts archiving processes and the library control daemon. But shared file system clients that are waiting for files to be staged from tape to the primary disk cache must reissue the stage requests.
6. If you still need to use a **crontab** entry to run the recycler process, restore the entry.
7. Stop here.

Converting an Unshared File System to a Shared File System

To convert an unshared file system to a shared file system, carry out the following tasks:

- Create a hosts file on the active and potential metadata servers.
- Share the unshared file system and configure the clients.

Create a Hosts File on the Active and Potential Metadata Servers

On each metadata server, you must create a hosts file that lists network address information for the servers and clients of a shared file system. The hosts file is stored alongside the **mcf** file in the **/etc/opt/SUNWsamfs/** directory. During the initial creation of a shared file system, the **sammkfs -S** command configures sharing using the settings stored in this file. So create it now, using the procedure below.

1. Gather the network host names and IP addresses for the hosts that will share the file system as clients.

   In the examples below, we will share the **shrfs1** file system with the clients **mds2** (a potential metadata server), **clnt1**, and **clnt2**.
2. Log in to the metadata server as **root**.

In the example, we log in to the host **mds**:

```
root@mds1~#
```

3. Using a text editor, create the file **/etc/opt/SUNWsamfs/hosts.***family-set-name* on the metadata server, replacing *family-set-name* with the name of the family-set name of the file-system that you intend to share.

   In the example, we create the file **hosts.shrfs1** using the **vi** text editor. We add some optional headings, starting each line with a hash sign (**#**), indicating a comment:

```
root@mds1~# vi /etc/opt/SUNWsamfs/hosts.shrfs1
# /etc/opt/SUNWsamfs/hosts.shrfs1
#                                          Server   On/  Additional
#Host Name            Network Interface    Ordinal  Off  Parameters
#------------------   ------------------   -------  ---  ----------
```

4. Enter the host name of the metadata server in the first column and the corresponding IP address or domain name the second. Separate the columns with whitespace characters.

   In the example, we enter the host name and IP address of the metadata server, **mds1** and **10.79.213.117**, respectively:

```
root@mds1~# vi /etc/opt/SUNWsamfs/hosts.shrfs1
# /etc/opt/SUNWsamfs/hosts.shrfs1
#                                          Server   On/  Additional
#Host Name            Network Interface    Ordinal  Off  Parameters
#------------------   ------------------   -------  ---  ----------
mds1                 10.79.213.117
```

5. Add a third column, separated from the network address by whitespace characters. In this column, enter the ordinal number of the server (**1** for the active metadata server, **2** for the first potential metadata server, and so on).

   In this example, there is only one metadata server, so we enter **1**:

```
root@mds1~# vi /etc/opt/SUNWsamfs/hosts.shrfs1
# /etc/opt/SUNWsamfs/hosts.shrfs1
#                                          Server   On/  Additional
#Host Name            Network Interface    Ordinal  Off  Parameters
#------------------   ------------------   -------  ---  ----------
mds1                 10.79.213.117        1
```

6. Add a fourth column, separated from the server ordinal number by whitespace characters. In this column, enter **0** (zero).

   A **0**, **-** (hyphen), or blank value in the fourth column indicates that the host is **on**—configured with access to the shared file system. A **1** (numeral one) indicates that the host is **off**—configured but without access to the file system (for information on using these values when administering shared file systems, see the **samshrfs1** man page).

```
root@mds1~# vi /etc/opt/SUNWsamfs/hosts.shrfs1
# /etc/opt/SUNWsamfs/hosts.shrfs1
#                                          Server   On/  Additional
#Host Name            Network Interface    Ordinal  Off  Parameters
#------------------   ------------------   -------  ---  ----------
mds1                 10.79.213.117        1        0
```

7. Add a fifth column, separated from the on/off status column by whitespace characters. In this column, enter the keyword **server** to indicate the currently active metadata server:

```
root@mds1~# vi /etc/opt/SUNWsamfs/hosts.shrfs1
# /etc/opt/SUNWsamfs/hosts.shrfs1
#                                        Server   On/  Additional
#Host Name            Network Interface  Ordinal  Off  Parameters
#------------------   -----------------  -------  ---  ----------
mds1                  10.79.213.117      1        0    server
```

8. If you plan to include one or more hosts as a potential metadata servers, create an entry for each. Increment the server ordinal each time. But do not include the **server** keyword (there can be only one active metadata server per file system).

   In the example, the host **mds2** is a potential metadata server with the server ordinal **2**. Until and unless we activate it as a metadata server, it will be a client:

```
root@mds1~# vi /etc/opt/SUNWsamfs/hosts.shrfs1
# /etc/opt/SUNWsamfs/hosts.shrfs1
#                                        Server   On/  Additional
#Host Name            Network Interface  Ordinal  Off  Parameters
#------------------   -----------------  -------  ---  ----------
mds1                  10.79.213.117      1        0    server
mds2                  10.79.213.217      2        0
```

9. Add a line for each client host, each with a server ordinal value of **0**.

   In the example, we add two clients, **clnt1** and **clnt2**.

```
root@mds1~# vi /etc/opt/SUNWsamfs/hosts.shrfs1
# /etc/opt/SUNWsamfs/hosts.shrfs1
#                                        Server   On/  Additional
#Host Name            Network Interface  Ordinal  Off  Parameters
#------------------   -----------------  -------  ---  ----------
mds1                  10.79.213.117      1        0    server
mds2                  10.79.213.217      2        0
clnt1                 10.79.213.33       0        0
clnt2                 10.79.213.47       0        0
```

10. Save the **/etc/opt/SUNWsamfs/hosts.***family-set-name* file, and quit the editor.

    In the example, we save the changes to **/etc/opt/SUNWsamfs/hosts.shrfs1** and exit the **vi** editor:

```
root@mds1~# vi /etc/opt/SUNWsamfs/hosts.shrfs1
# /etc/opt/SUNWsamfs/hosts.shrfs1
#                                        Server   On/  Additional
#Host Name            Network Interface  Ordinal  Off  Parameters
#------------------   -----------------  -------  ---  ----------
mds1                  10.79.213.117      1        0    server
mds2                  10.79.213.217      2        0
clnt1                 10.79.213.33       0        0
clnt2                 10.79.213.47       0        0
:wq
root@mds1~#
```

11. Place a copy of the new **/etc/opt/SUNWsamfs/hosts.***family-set-name* file on any potential metadata servers that are included in the shared file-system configuration.

    In the examples, we use Secure File Transfer Protocol to place a copy on host **mds2**:

```
root@mds1~# sftp root@mds2
Password:
sftp> cd /etc/opt/SUNWsamfs/
sftp> put /etc/opt/SUNWsamfs/hosts.shrfs1
sftp> bye
root@mds1~#
```

12. Now share the unshared file system and configure the clients.

Share the Unshared File System and Configure the Clients

1. Log in to the metadata server as **root**.

   In the example, we log in to the host **mds**:

   ```
   root@mds1~#
   ```

2. If you do not have current backup copies of the system files and configuration files, create backups now. Use the procedures in "Backing Up the Oracle HSM Configuration" on page 7-4.

3. If you do not have a current file-system recovery point file and a recent copy of the archive log, create them now. Use the procedures in "Backing Up File Systems" on page 7-1.

   If you set up an automated backup process for the file system during initial configuration, you may not need additional backups.

4. If you are converting an archiving file system, finish active archiving and staging jobs and stop any new activity before proceeding further.

5. Unmount the file system. Use the command **umount** *family-set-name*, where *family-set-name* is the family-set name of the file-system that you intend to share.

   For more information on mounting and unmounting Oracle HSM file systems, see the **mount_samfs** man page. In the example, we unmount the **hqfs1** file system:

   ```
   root@mds1~# umount /hsm/hqfs1
   root@mds1~#
   ```

6. Convert the file system to an Oracle HSM shared file system. Use the command **samfsck -S -F** *file-system-name*, where *file-system-name* is the family-set name of the file system.

   In the example, we convert the file system named **hqfs1**:

   ```
   root@mds1~# samfsck -S -F hqfs1
   ...
   root@mds1~#
   ```

7. Open the **/etc/opt/SUNWsamfs/mcf** file in a text editor, and locate the line for the file system.

   In the example, we use the **vi** editor:

   ```
   root@mds1~# vi /etc/opt/SUNWsamfs/mcf
   # Equipment           Equipment  Equipment  Family   Device   Additional
   # Identifier          Ordinal    Type       Set      State    Parameters
   #------------------  ---------  ---------  -------  ------   -----------------
   hqfs1                 200        ma         hqfs1    on
   /dev/dsk/c0t0d0s0     201        mm         hqfs1    on
   /dev/dsk/c0t3d0s0     202        md         hqfs1    on
   /dev/dsk/c0t3d0s1     203        md         hqfs1    on
   ```

8. In the **mcf** file, add the **shared** parameter to the additional parameters field in the last column of the file system entry. Then save the file and close the editor.

```
root@mds1~# vi /etc/opt/SUNWsamfs/mcf
# Equipment          Equipment  Equipment  Family   Device   Additional
# Identifier         Ordinal    Type       Set      State    Parameters
#------------------  ---------  ---------  -------  ------   -----------------
hqfs1                200        ma         hqfs1    on       shared
/dev/dsk/c0t0d0s0    201        mm         hqfs1    on
/dev/dsk/c0t3d0s0    202        md         hqfs1    on
/dev/dsk/c0t3d0s1    203        md         hqfs1    on
:wq
root@mds1~#
```

9. Open the **/etc/vfstab** file in a text editor, and locate the line for the file system.

   In the example, we use the **vi** editor:

```
root@mds1~# vi /etc/vfstab
#File
#Device    Device   Mount       System  fsck  Mount    Mount
#to Mount  to fsck  Point       Type    Pass  at Boot  Options
#--------  -------  --------    ------  ----  -------  -----------------------
/devices   -        /devices    devfs   -     no       -
/proc      -        /proc       proc    -     no       -
...
hqfs1      -        /hsm/hqfs1  samfs   -     yes
```

10. In the **/etc/vfstab** file, and add the **shared** mount option to mount options field in the last column of the file system entry. Then save the file and close the editor.

```
root@mds1~# vi /etc/vfstab
#File
#Device    Device   Mount       System  fsck  Mount    Mount
#to Mount  to fsck  Point       Type    Pass  at Boot  Options
#--------  -------  --------    ------  ----  -------  -----------------------
/devices   -        /devices    devfs   -     no       -
/proc      -        /proc       proc    -     no       -
...
hqfs1      -        /hsm/hqfs1  samfs   -     yes      shared
:wq
root@mds1~#
```

11. Initialize the shared file system and host configuration. Use the command **samsharefs -u -R** *family-set-name*, where *family-set-name* is the family-set name of the file system.

```
root@mds1~# samsharefs -u -R hqfs1
```

12. Tell the Oracle HSM software to re-read the **mcf** file and reconfigure itself accordingly:

```
root@mds1~# samd config
Configuring SAM-FS ...
root@mds:~#
```

13. Mount the shared file system on the metadata server.

```
root@mds1~# mount /hsm/hqfs1
```

14. If your hosts are configured with multiple network interfaces, use local host files to route network communications.

15. Add any required clients to the newly shared file system, using the procedures outlined in "Configuring Additional File System Clients" on page 3-44.

Use Local Hosts Files to Route Network Communications

Individual hosts do not require local hosts files. The file system's global file on the metadata server identifies the active metadata server and the network interfaces of active and potential metadata servers for all file system hosts (see "Create a Hosts File on the Active and Potential Metadata Servers" on page 3-66). But local hosts files can be useful when you need to selectively route network traffic between file-system hosts that have multiple network interfaces.

Each file-system host identifies the network interfaces for the other hosts by first checking the **/etc/opt/SUNWsamfs/hosts.***family-set-name* file on the metadata server, where *family-set-name* is the name of the file system family specified in the **/etc/opt/SUNWsamfs/mcf** file. Then it checks for its own, specific **/etc/opt/SUNWsamfs/hosts.***family-set-name***.local** file. If there is no local hosts file, the host uses the interface addresses specified in the global hosts file in the order specified in the global file. But if there is a local hosts file, the host compares it with the global file and uses only those interfaces that are listed in both files in the order specified in the local file. By using different addresses in each file, you can thus control the interfaces used by different hosts.

To configure local hosts files, use the procedure outlined below:

1. On the metadata server host and on each potential metadata server host, create a copy of the global hosts file, **/etc/opt/SUNWsamfs/hosts.***family-set-name*.

   For the examples in this section, the shared file system, **shrfs1**, includes an active metadata server, **mds1**, and a potential metadata server, **mds2**, each with two network interfaces. There are also two clients, **clnt1** and **clnt2**.

   We want the active and potential metadata servers to communicate with each other via private network addresses and with the clients via host names that Domain Name Service (DNS) can resolve to addresses on the public, local area network (LAN). So **/etc/opt/SUNWsamfs/hosts.shrfs1**, the file system's global host file, specifies a private network address in the **Network Interface** field of the entries for the active and potential servers and a host name for the interface address of each client. The file looks like this:

```
# /etc/opt/SUNWsamfs/hosts.shrfs1
#                                          Server   On/  Additional
#Host Name           Network Interface     Ordinal  Off  Parameters
#-----------------   --------------------  -------  ---  ----------
mds1                 172.16.0.129          1        0    server
mds2                 172.16.0.130          2        0
clnt1                clnt1                 0        0
clnt2                clnt2                 0        0
```

2. Create a local hosts file on each of the active and potential metadata servers, using the path and file name **/etc/opt/SUNWsamfs/hosts.***family-set-name***.local**, where *family-set-name* is the name specified for the shared file system in the **/etc/opt/SUNWsamfs/mcf** file. *Only include interfaces for the networks that you want the active and potential servers to use.*

   In the example, we want the active and potential metadata servers to communicate with each other over the private network, so the local hosts file on each server, **hosts.shrfs1.local**, lists only private addresses for active and potential servers:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/hosts.shrfs1.local
# /etc/opt/SUNWsamfs/hosts.shrfs1.local
#                                          Server   On/  Additional
#Host Name            Network Interface    Ordinal  Off  Parameters
#------------------   ------------------   -------  ---  ----------
mds1                  172.16.0.129         1        0    server
mds2                  172.16.0.130         2        0
:wq
root@mds1:~# ssh root@mds2
Password:
root@mds2:~# vi /etc/opt/SUNWsamfs/hosts.shrfs1.local
# /etc/opt/SUNWsamfs/hosts.shrfs1.local
#                                          Server   On/  Additional
#Host Name            Network Interface    Ordinal  Off  Parameters
#------------------   ------------------   -------  ---  ----------
mds1                  172.16.0.129         1        0    server
mds2                  172.16.0.130         2        0
:wq
root@mds2:~# exit
root@mds1:~#
```

3. Create a local hosts file on each of the clients, using the path and file name **/etc/opt/SUNWsamfs/hosts.***family-set-name***.local**, where *family-set-name* is the name specified for the shared file system in the **/etc/opt/SUNWsamfs/mcf** file. *Only include interfaces for the networks that you want the clients to use.*

   In our example, we want the clients to communicate with the server only via the public network. So the file includes only the host names of the active and potential metadata servers:

```
root@mds1:~# ssh root@clnt1
Password:
root@clnt1:~# vi /etc/opt/SUNWsamfs/hosts.shrfs1.local
# /etc/opt/SUNWsamfs/hosts.shrfs1
#                                          Server   On/  Additional
#Host Name            Network Interface    Ordinal  Off  Parameters
#------------------   ------------------   -------  ---  ----------
mds1                  mds1                 1        0    server
mds2                  mds2                 2        0
:wq
root@clnt1:~# exit
root@mds1:~# ssh root@clnt2
Password:
root@clnt2:~# vi /etc/opt/SUNWsamfs/hosts.shrfs1.local
# /etc/opt/SUNWsamfs/hosts.shrfs1.local
#                                          Server   On/  Additional
#Host Name            Network Interface    Ordinal  Off  Parameters
#------------------   ------------------   -------  ---  ----------
mds1                  mds1                 1        0    server
mds2                  mds2                 2        0
:wq
root@clnt2:~# exit
root@mds1:~#
```

4. If you started this procedure while finishing the configuration of the server, add clients.

Converting a Shared File System to an Unshared File System

When you need to unshare a file system, proceed as follows:

Convert a Shared Metadata Server to an Unshared System

1. Log in to the metadata server as **root**.

   In the example, we log in to the host **mds**:

   ```
   root@mds:~#
   ```

2. Remove the clients from the metadata server configuration.

3. If you do not have current backup copies of the system files and configuration files, create backups now. See "Backing Up the Oracle HSM Configuration" on page 7-4.

4. If you do not have a current file-system recovery point file and a recent copy of the archive log, create them now. See "Backing Up File Systems" on page 7-1.

   If you set up an automated backup process for the file system during initial configuration, you may not need additional backups.

5. If you are converting an archiving file system, finish active archiving and staging jobs and stop any new activity before proceeding further.

6. Unmount the file system. Use the command **umount** *family-set-name*, where *family-set-name* is the name specified for the shared file system in the **/etc/opt/SUNWsamfs/mcf** file.

   For more information on mounting and unmounting Oracle HSM file systems, see the **mount_samfs** man page. In the example, we unmount the **hqfs1** file system:

   ```
   root@mds:~# umount /hsm/hqfs1
   ```

7. Convert the Oracle HSM shared file system to an unshared file system. Use the command **samfsck -F -U** *file-system-name*, where *file-system-name* is the name specified for the shared file system in the **/etc/opt/SUNWsamfs/mcf** file.

   In the example, we convert the file system named **hqfs1**:

   ```
   root@mds:~# samfsck -F -U hqfs1
   samfsck: Configuring file system
   samfsck: Enabling the sam-fsd service.
   name:     hqfs1       version:     2A
   First pass
   Second pass
   Third pass
   ...
   root@mds:~#
   ```

8. Open the **/etc/opt/SUNWsamfs/mcf** file in a text editor, and locate the line for the file system.

   In the example, we use the **vi** editor:

   ```
   root@mds:~# vi /etc/opt/SUNWsamfs/mcf
   # Equipment          Equipment  Equipment  Family   Device   Additional
   # Identifier         Ordinal    Type       Set      State    Parameters
   #-----------------  ---------  ---------  -------  ------   -----------------
   hqfs1                200        ma         hqfs1    on       shared
   /dev/dsk/c0t0d0s0    201        mm         hqfs1    on
   /dev/dsk/c0t3d0s0    202        md         hqfs1    on
   /dev/dsk/c0t3d0s1    203        md         hqfs1    on
   ```

9. In the **mcf** file, delete the **shared** parameter from the additional parameters field in the last column of the file system entry. Then save the file and close the editor.

```
root@mds:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment          Equipment  Equipment  Family   Device   Additional
# Identifier         Ordinal    Type       Set      State    Parameters
#------------------  ---------  ---------  -------  ------   ----------------
hqfs1                200        ma         hqfs1    on
/dev/dsk/c0t0d0s0    201        mm         hqfs1    on
/dev/dsk/c0t3d0s0    202        md         hqfs1    on
/dev/dsk/c0t3d0s1    203        md         hqfs1    on
:wq
root@mds:~#
```

10. Open the **/etc/vfstab** file in a text editor, and locate the line for the file system.

    In the example, we use the **vi** editor:

```
root@mds:~# vi /etc/vfstab
#File
#Device    Device   Mount       System  fsck  Mount    Mount
#to Mount  to fsck  Point       Type    Pass  at Boot  Options
#--------  -------  --------    ------  ----  -------  -----------------------
/devices   -        /devices    devfs   -     no       -
/proc      -        /proc       proc    -     no       -
...
hqfs1      -        /hsm/hqfs1  samfs   -     yes      shared
```

11. In the **/etc/vfstab** file, delete the **shared** mount option from the mount options field in the last column of the file system entry. Then save the file and close the editor.

    In the example, we use the **vi** editor:

```
root@mds:~# vi /etc/vfstab
#File
#Device    Device   Mount       System  fsck  Mount    Mount
#to Mount  to fsck  Point       Type    Pass  at Boot  Options
#--------  -------  --------    ------  ----  -------  -----------------------
/devices   -        /devices    devfs   -     no       -
/proc      -        /proc       proc    -     no       -
...
hqfs1      -        /hsm/hqfs1  samfs   -     yes
:wq
root@mds:~#
```

12. Delete the file **/etc/opt/SUNWsamfs/hosts.***file-system-name*.

13. Tell the Oracle HSM software to re-read the **mcf** file and reconfigure itself accordingly:

```
root@mds:~# samd config
Configuring SAM-FS ...
root@mds:~#
```

14. Mount the file system.

```
root@mds:~# mount /hsm/hqfs1
root@mds:~#
```

15. Stop here.

4

Managing Files and Directories

This chapter addresses the following file and directory management topics:

- setting Oracle HSM file attributes
- using extended file attributes
- accommodating large files
- working with Linear Tape File System (LTFS) volumes
- managing directories and files in SMB/CIFS shares
- administering access control lists (ACLs).

Setting Oracle HSM File Attributes

The ability to interact with users via a familiar interface—the standard UNIX file system—is a key advantage of Oracle Hierarchical Storage Manager and StorageTek QFS Software. Most users do not even need to be aware of the differences. However, Oracle HSM file systems can provide advanced users with significantly greater capabilities when necessary. Oracle HSM file attributes let users optimize the behavior of the file system for working with individual files and directories. Users who understand their workloads and the characteristics of their data can significantly improve performance file-by-file. Users can, for example, specify direct or buffered I/O based on the characteristics of the data in a given file or directory. They can preallocate file-system space so that large files can be written more sequentially and can specify the stripe width used when writing particular files or directories.

The **`setfa`** command sets these file attributes on both new and existing files and directories. The command creates specified files or directories that do not exist. When applied to a directory, it sets the specified attributes on all files and subdirectories in the directory. Subsequently created files and directories inherit these attributes.

The basic tasks are outlined below (for additional information, see the **`setfa`** man page).

- restoring default file attribute values
- preallocating space in the file system
- specifying round-robin or striped allocation for a file
- allocating file storage on a specified stripe-group device.

Restore Default File Attribute Values

1. Log in to the file system host.

In the example, we log in to the metadata server **mds1**:

```
user@mds1:~#
```

2. To reset the default attribute values on a file, use the command **setfa -d** *file*, where *file* is the path and name of the file.

   In the example, we reset defaults on the file **/samfs1/data/2014/03/series3.15**:

   ```
   user@mds1:~# setfa -d /samfs1/data/2014/03/series3.15
   ```

3. To recursively reset the default attribute values on a directory and all of its contents, use the command **setfa -r** *directory*, where *directory* is the path and name of the directory.

   In the example, we reset the defaults on the subdirectory **/samfs1/data/2014/02**:

   ```
   user@mds1:~# setfa -r /samfs1/data/2014/02/
   ```

4. Stop here.

Preallocate File-System Space

Preallocating space for a file insures that there is enough room to write out the entire file sequentially when the file is written. Writing and reading large files in sequential blocks improves efficiency and overall performance by reducing the overhead associated with seeking and with buffering smaller, more scattered blocks of data. Preallocation is thus best for writing a predictable number of large blocks of data. Preallocated but unused space remains part of the file when the file closes and cannot be freed for other use until the entire file is deleted.

1. Log in to the file system host.

   In the example, we log in to the metadata server **mds1**:

   ```
   user@mds1:~#
   ```

2. If you need to preallocate space for writing to an existing file that already contains data, use the command **setfa -L** *number-bytes file*, where *number-bytes* is an integer or an integer plus **k** for kilobytes, **m** for megabytes, or **g** for gigabytes, and where *file* is the name of the file.

   The command **setfa -L** uses standard allocation.It supports striping. Pre-allocated files can grow beyond their pre-allocated size. In the example, we preallocate 121 megabytes for the existing file **tests/series119b**:

   ```
   user@mds1:~# setfa -L 121m tests/series119b
   ```

3. If you need to preallocate space for writing a new file that has no storage blocks assigned, use the command **setfa -l** *number-bytes file*, where:

   - **l** is the lower-case letter "L".
   - *number-bytes* is an integer or an integer plus **k** for kilobytes, **m** for megabytes, or **g** for gigabytes.
   - *file* is the name of the file.

   The command **setfa -l** preallocates the specified number of bytes. The resulting files are fixed at their preallocated size and can neither grow beyond nor shrink below their pre-allocated size. In the example, we create the file **data/2014/a3168445** and preallocate two gigabytes of space for its content:

   ```
   user@mds1:~# setfa -l 2g data/2014/a3168445
   ```

4. Stop here.

Specify Round-Robin or Striped Allocation for a File

By default, Oracle HSM file systems use the allocation method specified for the file system at mount time. But users can specify a preferred allocation method—round-robin or striping with a specified stripe-width—for specified directories or files.

1. Log in to the file system host.

   In the example, we log in to the metadata server **mds1**:

   ```
   user@mds1:~#
   ```

2. To specify round-robin allocation, specify a stripe width of **0** (zero). Use the command **setfa -s 0** *directory-or-file*, where *directory-or-file* is the name of the directory or file that will be written using the specified allocation method.

   A stripe width of **0** (zero) specifies unstriped, round-robin allocation. The file system starts writing a file on the next available device. It writes successive disk allocation units (DAUs) to the file on the same device until the file is complete or the device runs out of space. If the device runs out of space, the file system moves to the next available device and continues to write disk allocation units. The process repeats until the file is complete. In the example, we specify round-robin allocation for all files written to the **data/field-reports** directory:

   ```
   user@mds1:~# setfa -s 0 data/field-reports
   ```

3. To specify striped allocation, specify a stripe width. Use the command **setfa -s** *stripe-width directory-or-file*, where *stripe-width* is an integer in the range [**1-255**] and *directory-or-file* is the name of the directory or file that will be written using the specified allocation method.

   A stripe width in the range [**1-255**] specifies striped allocation. The file system writes the number of disk allocation units (DAUs) specified in the stripe width to multiple devices in parallel until the file is complete. In the example, we specify striped allocation with a stripe width of **1** for all files written to the directory, so the file allocation for all files written to the **data/field-reports** directory **data/2014/**, so the file system will write one disk allocation unit to each available device until the file is complete:

   ```
   user@mds1:~# setfa -s 1 data/2014/
   ```

4. Stop here.

Allocate File Storage on a Specified Stripe-Group Device

A user can specify the *stripe group* device where round-robin or striped allocation should begin. An Oracle HSM stripe group is a logical volume that stripes data across multiple physical volumes. When round-robin file allocation is in effect, the entire file is written on the designated stripe group. When striped allocation is in effect, the first allocation is made on the designated stripe group.

1. Log in to the file system host.

   In the example, we log in to the metadata server **mds1**:

   ```
   user@mds1:~#
   ```

2. To write an entire file to a specific stripe group, use round-robin allocation. Use the command **setfa -s 0 -g***stripe-group-number*, where *stripe-group-number* is an integer in the range [**0-127**] that identifies the specified stripe group.

   In the example, we specify round-robin allocation starting on stripe group **0** when writing the file **reports/site51**:

   ```
   user@mds1:~# setfa -s 0 -g0 reports/site51
   ```

3. To stripe a file across a number of stripe groups starting from a specified stripe group, use striped allocation. Use the command **setfa -s** *stripe-width* **-g** *stripe-group-number*, where *stripe-width* is an integer in the range [**1-255**] that specifies a number of disk allocation units and *stripe-group-number* is an integer in the range [**0-127**] that identifies the specified stripe group.

   In the example, we specify striped allocation for the file **assessments/site52**. We specify three disk allocation units per group, starting from stripe group **21**:

   ```
   user@mds1:~# setfa -s 3 -g21 assessments/site52
   ```

4. Stop here.

Using Extended File Attributes

Like other Solaris and Linux file systems, Oracle HSM file systems support extended file attributes. Extended attributes hold arbitrary metadata that is associated with a file by a user or application, rather than by the file system itself. Extended attributes have been used to hold file digests, the names of authors and source applications, and the character encoding used by text files.

Starting in Release 6.1, Oracle HSM stores small extended attribute files that contain 464 or fewer characters in extension inodes within the metadata partition, instead of using a block in the data partition. The new approach significantly improves file-system performance when extended attributes are in use and file-system metadata is being stored on faster devices, such as flash storage.

Extended file attributes are automatically enabled whenever you create a new file system or restore an old file system from a recovery point (**samfsdump**) file. For more information on using extended attributes, see the Solaris **fsattr(5)** and Linux **xattr(7)** man pages.

Accommodating Large Files

Oracle HSM file systems are particularly well suited to working with unusually large files. This section covers the following topics:

- managing disk cache with very large files
- segmenting large files
- using removable media files for large data sets.

Managing Disk Cache With Very Large Files

When manipulating very large files, pay careful attention to the size of available disk cache. If you try to write a file that is larger than your disk cache, non-archiving file systems return an **ENOSPC** error, while archiving file systems simply waits for space that may never become available, causing applications to block.

Oracle HSM provides two possible alternatives to increasing the size of the disk cache:

- Segmenting files, so that users stage only part of a large file to disk at any given time
- Using removable media files for large data sets, so that users never stage data to disk.

Segmenting Files

When you set the Oracle HSM segmentation attribute on a file, the file system breaks the file down into segments of a specified size and manages access requests so that, at any given time, only the currently required segment resides on disk. The remainder of the file resides on removable media.

Segmentation of large files has a number of advantages:

- Users can create and access files that are larger than the available disk cache.

  Since only segments reside in cache at any given time, you only need to choose a segment size that fits in the disk cache. The complete file can grow to any size that the media can accommodate.
- Users can access large files that have been released from the disk cache more quickly. Staging a portion of a large file to disk is much faster than waiting for the entire file to stage.
- The speed and efficiency of archiving can improve when files are segmented, because only changed portions of each file are re-archived.
- Files can be striped across removable media volumes mounted on multiple drives. Archiving and staging operations can then proceed in parallel, further improving performance.

There are two limitations:

- You cannot segment files in a shared file system.
- You cannot segment binary executable files, because the Solaris memory-mapping function, **`mmap()`**, cannot map the bytes in a segmented file to the address space of a process.

To create segmented files, proceed as follows:

Segment a File

1. Log in to the file system host.

   In the example, we log in to the metadata server **`mds1`**:

   ```
   user@mds1:~#
   ```

2. Select or, if necessary, create the file(s) that you need to segment.
3. To segment a single file, use the command **`segment [-s stage_ahead] -l`** *`segment_size file-path-name`*, where:

   - **`stage_ahead`** (optional) is an integer specifying the number of consecutive extra segments to read when a given segment is accessed. Well-chosen values can improve utilization of the system page cache and thus improve I/O performance. The default is **`0`** (disabled).
   - **`segment_size`** is an integer and a unit that together specify the size of each segment. Supported units are **`k`** (kilobytes), **`m`** (megabytes, and **`g`** (gigabytes). The minimum size is one megabyte (**`1m`** or **`1024k`**).

- *file-path-name* is the path and file name of the file.

For full details, see the **segment** man page. In the example, we segment the file **201401.dat** using a 1.5-megabyte (**1536k**) segment size:

```
user@mds1:~# segment -l 1536k 201401.dat
```

4. To recursively segment the files in a directory and all of its subdirectories, use the command **segment [-s** *stage_ahead***] -l** *segment_size* **-r** *directory-path-name*, where *directory-path-name* is the path and name of the starting directory.

   In the example, we segment all files in the **/hsm/hsmfs1/data** directory and its subdirectories using a 1-megabyte (**1m**) segment size:

   ```
   user@mds1:~# segment -l 1m -r /hsm/hsmfs1/data
   ```

5. Stop here.

Stripe a Segmented File Across Multiple Volumes

You configure segmented files for striped I/O by assigning them to an archive set that specifies multiple drives. Proceed as follows:

1. Log in to the host as **root**.

   In the example, we log in to the metadata server **mds1**:

   ```
   root@mds1:~#
   ```

2. Open the file **/etc/opt/SUNWsamfs/archiver.cmd** in a text editor.

   In the example, we use the **vi** editor to open the file:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/archiver.cmd
   # Configuration file for Oracle HSM archiving file systems ...
   ```

3. To stripe segmented files across drives, specify the use of at least two drives for each copy of each archive set that contains segmented files. In the **archiver.cmd** file, locate the **params** section. Make sure that the parameters for each copy include the **-drives** *number* parameter, where *number* is two (**2**) or more. Make any required changes, save the file, and close the editor.

   In the example, the **archiver.cmd** file specifies two drives for all three copies of all configured archive sets:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/archiver.cmd
   # Configuration file for Oracle HSM archiving file systems ...
   ...
   #-----------------------------------------------------------------------
   # Copy Parameters
   params
   allsets -sort path -offline_copy stageahead -reserve set
   allsets.1 -startage 10m -drives 2
   allsets.2 -startage 24h -drives 2
   allsets.3 -startage 48h -drives 2
   endparams
   ...
   :wq
   root@mds1:~#
   ```

4. Check the **archiver.cmd** file for errors. Use the command **archiver -lv**.

The **archiver -lv** command prints the **archiver.cmd** file to screen and generates a configuration report if no errors are found. Otherwise, it notes any errors and stops.

```
root@mds1:~# archiver -lv
Reading '/etc/opt/SUNWsamfs/archiver.cmd'.
...
 Total space available:  300T
root@mds1:~#
```

5. Tell the Oracle HSM software to re-read the **archiver.cmd** file and reconfigure itself accordingly. Use the **/opt/SUNWsamfs/sbin/samd config** command.

```
root@mds1:~# samd config
```

6. Stop here.

Using Removable Media Files for Large Data Sets

Oracle HSM *removable media files* reside entirely on removable media and thus never occupy space in the file-system disk cache. The file system reads removable media files directly into memory. So the storage medium does not limit the size of the file at all. Removable files that exceed the capacity of a single media cartridge can become multiple-cartridge, *volume overflow files*. The file system reads and writes data to the media sequentially.

In most respects, removable media files look like typical UNIX files. They have permissions, a user name, a group name, and a file size. When a user or application requests a removable media file, the system automatically mounts the corresponding volume(s) and the user accesses the data from memory, much as if the data were on disk. But removable media files differ from other Oracle HSM files in two major ways: they are never archived by the Oracle Hierarchical Storage Manager software, and they are not supported over NFS.

The Oracle Hierarchical Storage Manager software does not manage removable media files. The files are never archived or released, and the media that contains them is never recycled. This makes removable media files useful when you need to use removable media for purposes other than archiving. These files are ideal for creating removable disaster-recovery volumes that back up your Oracle HSM configuration and metadata dump files. You can also read data from foreign volumes (volumes created by other applications) by loading the volume read-only and reading the files into memory as removable media files.

Since removable media files cannot be released and the associated volume(s) cannot be recycled, you should generally segregate removable media files on dedicated volumes, rather than mixing them in with archive copies.

Create a Removable Media or Volume Overflow File

1. Log in to the file system host.

   In the example, we log in to the metadata server **mds1**:

```
user@mds1:~#
```

2. Select the Oracle HSM file system, path, and file name for the removable media file.

   Once the removable media file is created, the file system will address requests for this path and file name using data from removable media.

3. Create the removable media file. Use the command **request -m** *media-type* **-v** *volume-specifier data-file*, where *mediatype* is one of the two-character media type codes listed in Appendix A, *data-file* is the path and name that you selected for the removable media file, and *volume-specifier* is one of the following:

   - a volume serial number or a slash-delimited list of volume serial numbers

     In the first example, we create **file1** on LTO (**li**) volume **VOL080**:

     ```
     user@mds1:~# request -m li -v VOL080 /hsm/hsmfs1/data/file1
     ```

     In the second example, we create **file2** on LTO (**li**) volumes **VOL080**, **VOL082**, and **VOL098**:

     ```
     user@mds1:~# request -m li -v VOL081/VOL082/VOL098 /hsm/hsmfs1/data/file2
     ```

   - **-l** *volume-list-file*, where *volume-list-file* is the path and name of a file that, on each line, lists a single volume serial number and, optionally, a space and a decimal or hexadecimal number specifying a starting position on the specified volume (prefix hexadecimals with **0x**).

     In the example, using the **vi** editor, we create **file3** on the LTO (**li**) volumes listed in the file **vsnsfile3**:

     ```
     user@mds1:~# vi vsnsfile3
     VOL180
     VOL181
     VOL182
     :wq
     user@mds1:~# request -m li -v -l vsnsfile3 /hsm/hsmfs1/data/file3
     ```

4. Stop here.

Read a Foreign Tape Volume as a Removable Media File

1. Log in to the file system host.

   In the example, we log in to the metadata server **mds1**:

   ```
   user@mds1:~#
   ```

2. Make sure that the foreign tape is barcoded, write protected, opened read-only, and positioned to **0**.

3. Select the Oracle HSM file system, path, and file name for the removable media file.

   Once the removable media file is created, the file system will address requests for this path and file name using data from the foreign tape.

4. Create the removable media file using the **-N** (foreign media) option. Use the command **request -m** *media-type* **-N -v** *volume-serial-number data-file*, where:

   - *mediatype* is one of the two-character media type codes listed in Appendix A.
   - *volume-serial-number* is the volume serial number of the foreign tape.
   - *data-file* is the path and name for the removable media file.

   In the example, we create a removable media file for the foreign LTO (**li**) volume **FOR991**:

   ```
   user@mds1:~# request -m li -N -v FOR991 /hsm/hsmfs1/foreignfile
   ```

5. Stop here.

Working with Linear Tape File System (LTFS) Volumes

Linear Tape File System is a self-describing tape format that organizes the data on sequential-access tape media into a file system, so that files can be accessed much as if they resided on random-access disk. Oracle HSM provides extensive support for LTFS. The software lets you use LTFS files in Oracle HSM file systems and supplies tools for creating, accessing, and managing LTFS media.

This section addresses the following topics:

- importing LTFS media into the library
- attaching LTFS directories and files to an Oracle HSM file system
- accessing LTFS media using Oracle HSM software
- managing LTFS media using Oracle HSM software.

Importing LTFS Media Into the Library

The Oracle HSM software automatically recognizes LTFS media. So you can import LTFS volumes with the **samimport** command, just as you would any other media. See "Importing and Exporting Removable Media" on page 5-2 and the **import** (1m) man page for additional information.

Attaching LTFS Directories and Files to an Oracle HSM File System

The Oracle HSM software can attach Linear Tape File System (LTFS) directories and files to an Oracle HSM file system, so that they can be accessed and managed as if they were themselves Oracle HSM files. The software copies the LTFS meta-data from the LTFS volume to an empty directory in an Oracle HSM file system. Using this metadata, Oracle HSM manages the LTFS media and files as it would an archived Oracle HSM file. LTFS files are staged from the LTFS media to the Oracle HSM disk cache for use, either when users access them or all at once, as soon as the LTFS metadata is in place. The Oracle HSM file system's archiving and space-management policies apply as they would for any Oracle HSM file.

This section describes the following tasks:

- making LTFS files accessible on demand
- making LTFS files immediately accessible in the disk cache.

Making LTFS Files Accessible On Demand

When you *attach* LTFS files to an Oracle HSM file system, the Oracle HSM software copies file-system metadata from the LTFS volume to a specified directory in the Oracle HSM file system. Files will then be staged to the disk cache when users access them. To attach LTFS files, proceed as follows:

1. Log in to the file system host.

   In the example, we log in to the metadata server **mds1**:

   ```
   user@mds1:~#
   ```

2. In the Oracle HSM file system that will host the LTFS files, create the directory that will hold the LTFS metadata.

In the example, we create the directory **ltfs1/** under the file-system mount point **/hsm/hsmfs1**:

```
user@mds1:~# mkdir /hsm/hsmfs1/ltfs1
user@mds1:~#
```

3. Attach the LTFS files to the Oracle HSM file system. Use the command **samltfs attach** *LTFS-media-type*.*LTFS-volume-serial-number SAMQFS-directory*, where:

   - *LTFS-media-type* is the two-character media type code for the type of media that holds the LTFS data (see Appendix A).
   - *LTFS-volume-serial-number* is the six-character, alphanumeric volume serial number of the LTFS volume.
   - The specified media type and volume serial number identify a volume that the catalog lists as an LTFS volume.

     In the Oracle HSM catalog, LTFS media are unlabeled and marked **non-SAM** and **tfs**.
   - *SAMQFS-directory* is the path and name of the directory that will hold LTFS metadata.

   In the example, we attach LTO (**li**) volume **TFS233**:

   ```
   user@mds1:~# samltfs attach li.TFS233 /hsm/hsmfs1/ltfs1
   user@mds1:~#
   ```

4. Stop here.

Making LTFS Files Immediately Accessible in the Disk Cache

When you *ingest* LTFS files into an Oracle HSM file system, the Oracle HSM software copies file-system metadata from the LTFS volume to a specified directory in the Oracle HSM file system and immediately stages all files to the disk cache. To ingest LTFS files, proceed as follows:

1. Log in to the file system host.

   In the example, we log in to the metadata server **mds1**:

   ```
   user@mds1:~#
   ```

2. In the Oracle HSM file system that will host the LTFS files, create the directory that will hold the LTFS metadata.

   In the example, we create the directory **ltfs2/** under the file-system mount point **/hsm/hsmfs1**:

   ```
   user@mds1:~# mkdir /hsm/hsmfs1/ltfs2
   user@mds1:~#
   ```

3. Ingest the LTFS files into the Oracle HSM file system. Use the command **samltfs ingest** *LTFS-media-type*.*LTFS-volume-serial-number SAMQFS-directory*, where:

   - *LTFS-media-type* is the two-character media type code for the type of media that holds the LTFS data (see Appendix A).
   - *LTFS-volume-serial-number* is the six-character, alphanumeric volume serial number of the LTFS volume.
   - The specified media type and volume serial number identify a volume that the catalog lists as an LTFS volume.

In the Oracle HSM catalog, LTFS media are unlabeled and marked **non-SAM** and **tfs**.

- *SAMQFS-directory* is the path and name of the directory that hold LTFS metadata.

In the example, we ingest LTO (**li**) volume **TFS234**:

```
user@mds1:~# samltfs ingest li.TFS234 /hsm/hsmfs1/ltfs2
user@mds1:~#
```

4. Stop here.

Accessing LTFS Media Using Oracle HSM Software

Oracle HSM software can also carry out the following tasks using the LTFS mount point specified in the Oracle HSM **defaults.conf** file:

- loading LTFS volumes into tape drives and mounting LTFS file systems
- unmounting LTFS file systems and unloading LTFS volumes.

Load an LTFS Volume Into a Tape Drive and Mount the LTFS File System

1. Log in to the file system host.

   In the example, we log in to the metadata server **mds1**:

   ```
   user@mds1:~#
   ```

2. Load the LTFS volume into a tape drive and mount the file system on the mount point specified in the **defaults.conf** file. Use the command **samltfs load** *LTFS-media-type*.*LTFS-volume-serial-number*, where:

   - *LTFS-media-type* is the two-character media type code for the type of media that holds the LTFS data (see Appendix A).
   - *LTFS-volume-serial-number* is the six-character, alphanumeric volume serial number of the LTFS volume.
   - The specified media type and volume serial number identify a volume that the catalog lists as an LTFS volume.

     In the Oracle HSM catalog, LTFS media are unlabeled and marked **non-SAM** and **tfs**.

   In the example, we load LTO (**li**) volume **TFS434** and mount it on the directory specified in the **defaults.conf** file, **/mnt/ltfs**:

   ```
   user@mds1:~# samltfs load li.TFS234
   ```

3. Stop here.

Unmount an LTFS File System and Unload the Volume from the Tape Drive

1. Log in to the file system host.

   In the example, we log in to the metadata server **mds1**:

   ```
   user@mds1:~#
   ```

2. Unmount the LTFS file system and unload the corresponding volume from the tape drive. Use the command **samltfs unload** *LTFS-media-type*.*LTFS-volume-serial-number*, where:

- *LTFS-media-type* is the two-character media type code for the type of media that holds the LTFS data (see Appendix A).
- *LTFS-volume-serial-number* is the six-character, alphanumeric volume serial number of the LTFS volume.
- The specified media type and volume serial number identify an LTFS volume that the catalog lists as an LTFS volume.

  In the Oracle HSM catalog, LTFS media are unlabeled and marked **non-SAM** and **tfs**.

In the example, we unmount the LTFS file system and unload LTO (**li**) volume **TFS435**:

```
user@mds1:~# samltfs unload li.TFS435
```

3. Stop here.

Managing LTFS Media Using Oracle HSM Software

The Oracle HSM software provides the basic tools needed for carrying out the following LTFS administrative tasks:

- formatting a volume as an LTFS file system
- erasing LTFS data and removing LTFS formatting and partitions from a volume
- checking the integrity of an LTFS file system
- displaying LTFS configuration and status information.

Format a Volume as an LTFS File System

1. Log in to the file system host.

   In the example, we log in to the metadata server **mds1**:

   ```
   user@mds1:~#
   ```

2. Partition and format a removable media volume for the LTFS file system. Use the command **samltfs mkltfs** *media-type*.*volume-serial-number*, where:

   - *media-type* is the two-character media type code for an LTFS-compatible type of media (see Appendix A).
   - *volume-serial-number* is the six-character alphanumeric volume serial number of the volume.

   In the example, we partition LTO (**li**) volume **VOL234** and format it as an LTFS volume:

   ```
   user@mds1:~# samltfs mkltfs li.VOL234
   ```

3. Stop here.

Erase LTFS Data and Remove LTFS Formatting and Partitions from a Volume

1. Log in to the file system host.

   In the example, we log in to the metadata server **mds1**:

   ```
   user@mds1:~#
   ```

2. Erase the LTFS volume and restore it to general use. Use the command **samltfs unltfs** *media-type*.*volume-serial-number*, where:

   - *media-type* is the two-character media type code for an LTFS-compatible type of media (see Appendix A).
   - *volume-serial-number* is the six-character alphanumeric volume serial number of the volume.

   In the example, we erase the LTFS file system data and metadata and remove the partitions on LTO (**li**) volume **VOL234**:

   ```
   user@mds1:~# samltfs unltfs li.VOL234
   ```

3. Stop here.

Check the Integrity of an LTFS File System

1. Log in to the file system host.

   In the example, we log in to the metadata server **mds1**:

   ```
   user@mds1:~#
   ```

2. Check the integrity of the LTFS file system. Use the command **samltfs ltfsck** *LTFS-media-type*.*LTFS-volume-serial-number*, where:

   - *LTFS-media-type* is the two-character media type code for the type of media that holds the LTFS data (see Appendix A).
   - *LTFS-volume-serial-number* is the six-character, alphanumeric volume serial number of the LTFS volume.
   - The specified media type and volume serial number identify an LTFS volume that the catalog lists as an LTFS volume.

     In the Oracle HSM catalog, LTFS media are unlabeled and marked **non-SAM** and **tfs**.

   In the example, we check the LTFS file system on LTO (**li**) volume **VOL234**:

   ```
   user@mds1:~# samltfs ltfsck li.VOL234
   ```

3. Stop here.

Display LTFS Configuration and Status Information

To display the configuration and status of LTFS, use the command **samltfs status**.

```
user@mds1:~# samltfs status
```

Managing Directories and Files in SMB/CIFS Shares

Using SMB/CIFS shares, administrators can make the directories and files in Oracle HSM UNIX file systems available to Microsoft Windows clients. This section addresses the following topics:

- managing system attributes in SMB/CIFS shares
- administering access control lists (ACLs).

Managing System Attributes in SMB/CIFS Shares

System attributes support SMB/CIFS file sharing by associating Oracle HSM files with non-UNIX metadata that can be interpreted by Microsoft Windows file systems. This section starts with a brief overview of the system attributes supported by Oracle HSM. It then provides basic instructions for the following tasks:

- displaying system attributes
- modifying system attributes.

Oracle HSM Supported System Attributes

System attributes are Boolean (true or false) values expressed by an attribute *name* with the value **true** or the negation of the name, **no***name*, with the value **false**. Oracle HSM provides the following system attributes in support of SMB/CIFS file sharing:

- **appendonly** means that users can only append data to the file. **noappendonly** means that this restriction is not in effect.
- **archive** means that the file has changed since it was last copied or backed up. **noarchive** means that the file has not changed since it was last copied or backed up. Oracle HSM does not currently use this attribute.
- **hidden** means that the file is not displayed in file listings by default. **nohidden** means that the file is displayed by default.
- **immutable** means that the directory or file and its contents cannot be changed or deleted. **noimmutable** means that the directory or file can be changed or deleted.
- **nodump** means that the file cannot be backed up. **nonodump** means that the file can be backed up. Oracle Solaris does not use this attribute.
- **nounlink** means that the file or the directory and its contents cannot be deleted or renamed. **nonounlink** means that the file or the directory and its contents can be deleted or renamed.
- **offline** means that the file has been released from an Oracle HSM file system. Microsoft Windows systems will not preview the file. **nooffline** means that the file is online and has not been released from an Oracle HSM file system.
- **readonly** means that the file cannot be deleted or modified. **noreadonly** means that the file can be deleted or modified. The attribute is ignored when applied to directories.
- **sparse** means that the stored file contains only non-zero data, with zeroes reduced to ranges that are restored by the file system when the file is accessed or copied to a file system that does not support sparse files. **nosparse** means that the file is not sparse.
- **system** means that the file is critical to the Microsoft Windows operating system, must not be altered or deleted, and should not be displayed in listings by default. **nosystem** means that the file is not a system file.

Display System Attributes

To view the system attributes of an Oracle HSM file, use the Solaris command **ls -/v** *file*, where *file* is the path and name of the file.

In the example, we list system attributes for the file **/hsm/hsmfs1/docs/plan.odt**:

```
user@mds1:~# ls -/v /hsm/hsmfs1/documents/master-plan.odt
-rw-r--r--   1 root root  40560 Mar 4 15:52 /hsm/hsmfs1/docs/plan.odt
{archive,nohidden,noreadonly,nosystem,noappendonly,nonodump,noimmutable,nonounlink
```

```
, nooffline,nosparse}
user@mds1:~#
```

Modify System Attributes

To change a system attribute value for a file to a specified value, use the Solaris command **chmod S+v{***attributes***)**, where *attributes* is a comma-delimited list of Oracle HSM supported system attributes.

See the **chmod** man page for a comprehensive explanation of syntax and available options. In the example, we change the archive attribute from **noarchive** (false) to **archive** (true):

```
root@mds1:~# ls -/v /hsm/hsmfs1/documents/master-plan.odt
-r-xr-xr-x 1 root root 40561 Mar 4 15:52 /hsm/hsmfs1/documents/master-plan.odt
{noarchive,nohidden,readonly,nosystem,noappendonly,nonodump,noimmutable,
nonounlink,offline,nosparse}
root@mds1:~# chmod S+v{archive} /hsm/hsmfs1/documents/master-plan.odt
root@mds1:~# ls -/v /hsm/hsmfs1/documents/master-plan.odt
-r-xr-xr-x 1 root root 40561 Mar 4 15:52 /hsm/hsmfs1/documents/master-plan.odt
{archive,nohidden,readonly,nosystem,noappendonly,nonodump,noimmutable,
nonounlink,offline,nosparse}
root@mds1:~#
```

Administering Access Control Lists

An Access Control List (ACL) is a table that defines access permissions for a file or directory. Each record or Access Control Entry (ACE) in the table defines the access rights of a particular user, group, or class of users or groups. By default, new file systems that you create with Oracle HSM Release 6.1.4 use the Access Control List (ACL) implementation introduced in Network File System (NFS) version 4 and Solaris 11.

A comprehensive account of Solaris ACL administration, syntax, and usage is outside the scope of this document. For full information, see the following references:

- the chapter "Using ACLs and Attributes to Protect Oracle Solaris ZFS Files" in the volume *Oracle Solaris 11.3 Administration: ZFS File Systems*, available in the *Oracle Solaris 11.3 Information Library* at **docs.oracle.com**
- the chapter "Access Control Lists on SMB Shares" in the volume *Managing SMB File Sharing and Windows Interoperability in Oracle Solaris 11.3*, also available in the *Oracle Solaris 11.3 Information Library* at **docs.oracle.com**
- the Solaris **ls** (1) and **chmod** (1) man pages.

5

Managing Libraries, Media, and Drives

This chapter covers the following media and drive management topics:

- managing automated media libraries
- managing drives
- managing removable media
- managing cloud storage.

Managing Automated Media Libraries

This section covers basic tasks associated with library maintenance and management:

- taking the library on and off line
- importing and exporting removable media
- maintaining library catalogs
- determining the order in which drives are installed in the library.

Taking the Library On and Off Line

You typically need to carry out the following tasks before and after routine library maintenance:

- taking the library offline
- bringing the library back online.

Take the Library Offline

If you need to stop Oracle HSM operations on only one library or if you need to power down a library, start by taking the library offline as described below:

1. Log in to the file system host as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

2. Finish up active archiving and staging jobs and keep any new jobs from starting. See "Idle Archiving and Staging Processes" on page 3-23 and "Stop Archiving and Staging Processes" on page 3-24.

3. Take the library offline. Use the command **samcmd off** *library-equipment-number*, where *library-equipment-number* is the equipment ordinal number assigned to the library in the **/etc/opt/SUNWsamfs/mcf** file.

Placing a library in the **off** state stops I/O operations and removes the library from the control of the Oracle HSM software. Any drives that have not been powered off remain in the **on** state. In the example, we take library **800** offline and check the result using **samcmd c**:

```
root@mds1:~# samcmd off 800
root@mds1:~# samcmd c
Device configuration samcmd     5.4 14:34:04 Mar  7 2014
samcmd on hsmfs1host
Device configuration:
ty   eq state   device_name                    fs   family_set
sn  800 off     /dev/scsi/changer/c1t2d0       800  lib800
li  801 on      /dev/rmt/0cbn                  800  lib800
li  802 on      /dev/rmt/1cbn                  800  lib800
li  803 on      /dev/rmt/2cbn                  800  lib800
li  804 on      /dev/rmt/3cbn                  800  lib800
hy  900 on      historian                      900
root@mds1:~#
```

4. When you are ready, bring the library online.

Bring the Library Online

1. Log in to the file system host as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

2. Bring the library online. Use the command **samcmd on** *library-equipment-number*, where *library-equipment-number* is the equipment ordinal number assigned to the library in the **/etc/opt/SUNWsamfs/mcf** file.

   The library comes online. Oracle HSM software queries the device state and updates the catalog as needed. In the example, we bring library **800** online and check the result using **samcmd c**:

   ```
   root@mds1:~# samcmd on 800
   root@mds1:~# samcmd c
   Device configuration samcmd     5.4 15:04:14 Mar  7 2014
   samcmd on hsmfs1host
   Device configuration:
   ty   eq state   device_name                    fs   family_set
   sn  800 on      /dev/scsi/changer/c1t2d0       800  lib800
   li  801 on      /dev/rmt/0cbn                  800  lib800
   li  802 on      /dev/rmt/1cbn                  800  lib800
   li  803 on      /dev/rmt/2cbn                  800  lib800
   li  804 on      /dev/rmt/3cbn                  800  lib800
   hy  900 on      historian                      900
   root@mds1:~#
   ```

3. Stop here.

Importing and Exporting Removable Media

Many automated libraries include a loading bay that lets you add or remove media cartridges without physically entering the library. Depending on the vendor, it may be called the mailbox, mailslot, media access port (MAP), or cartridge access port (CAP). With this type of library, you can use Oracle HSM commands to carry out the following tasks:

- importing removable media cartridges
- export removable media cartridges.

If your library does *not* include a mailbox, consult the library vendor's documentation and your local site policies for instructions on adding and removing library media. When the library reinitializes following the change and audits its contents, the Oracle HSM software will update its library and historian catalogs automatically.

Import Removable Media Cartridges

If the library mailbox contains media cartridges when the Oracle HSM software starts, the software automatically loads them into the library. Once the software is running, you can import media from the mailbox at any time using the following procedure:

1. Place media cartridge(s) in the media access port according to the library vendor's instructions.
2. Log in to the file system host as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

3. Import the cartridge(s) into the automated library. Use the command **samimport** *library-equipment-number*, where *library-equipment-number* is the equipment ordinal number specified for the library in the **/etc/opt/SUNWsamfs/mcf** file.

   The Oracle HSM software assigns the media to storage slots and catalogs their locations. In the example, we import media into library **800**.

   ```
   root@mds1:~# samimport 800
   ```

4. Stop here.

Export Removable Media Cartridges

1. Log in to the file system host as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

2. If required, add an informational note to the catalog record for a cartridge before exporting it. Use the command **chmed -I "***note***"** *identifier*, where *note* is a string of up to 128 characters and *identifier* is either of the following:

   - *mediatype*.*volume-serial-number*, where *mediatype* is one of the two-character media type codes listed in Appendix A and *volume-serial-number* is the six-character, alphanumeric string that uniquely identifies the volume within the library.
   - *library-equipment-number*:*slot*, where *library-equipment-number* is the equipment ordinal number specified for the automated tape library in the **/etc/opt/SUNWsamfs/mcf** file and *slot* is the slot address where the cartridge resides within the library.

   The note will be retained in the historian catalog after the volume has been exported. In the example, we add a note to the catalog entry for LTO (**li**) cartridge **VOL054**:

   ```
   root@mds1:~# chmed -I "To vault 20150411" li.VOL054
   ```

3. To move a cartridge from a specified storage slot to the mailbox, use the command **samexport** *library-equipment-number*:*slot*, where *library-equipment-number* is the equipment ordinal number specified for the automated tape library in the **/etc/opt/SUNWsamfs/mcf** file and *slot* is the slot address where the cartridge resides within the library.

   In the example, we export the magnetic tape cartridge located in slot **11** of library **800**:

   ```
   root@mds1:~# samexport 800:11
   ```

4. To move a specified cartridge to the mailbox, use the command **samexport** *mediatype*.*volume-serial-number*, where *mediatype* is one of the two-character media type codes listed Appendix A and *volume-serial-number* is the alphanumeric string that uniquely identifies the volume within the library.

   The Oracle HSM software adds the cartridge to the catalog maintained by the Oracle HSM **historian**. In the example, we export the LTO (**li**) tape cartridge **VOL109**:

   ```
   root@mds1:~# samexport li.VOL109
   ```

5. Remove media cartridge(s) from the mailbox according to the library vendor's instructions.
6. Stop here.

Maintaining Library Catalogs

Oracle Hierarchical Storage Manager library catalogs are the software's internal representation of the automated library and its contents. If the automated library is direct-attached, the Oracle HSM software has full control over the library and its contents. The library catalog entries are, accordingly, a one-to-one representation of the slots in the physical library. If the automated library is network-attached, Oracle HSM accesses only the parts of the library that the library software makes available in the form of a virtual library or library partition. So the Oracle HSM library catalog entries reflect only the contents of a portion of the library.

This section explains the following tasks:

- viewing the library catalog
- auditing the contents of a library slot
- auditing an entire direct-attached automated library
- clearing media errors from the catalog.

View the Library Catalog

1. Log in to the file system host as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

2. To view the most commonly used library catalog information, use the command **samcmd v** *library-equipment-number*, where *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the automated tape library.

   In the example, we display the catalog for library **800**

```
root@mds1:~# samcmd v 800
Robot catalog samcmd     5.4    16:45:25 Mar 14 2014
samcmd on samqfshost                          count 32
Robot VSN catalog by slot       : eq 800
slot          access time count use  flags         ty vsn
   0     2016/12/14 11:23  875   0%  -il-o-b-----  li VOL001
   1     2016/12/13 17:54  866   0%  -il-o-b-----  li VOL002
   2     2016/12/14 11:26    3   0%  -il-o-b-----  li VOL003
   3     2016/12/14 10:33    3   0%  -il-o-b-----  li VOL004
   4     2016/12/14 11:34    5   0%  -il-o-b-----  li VOL005
   5     2016/12/14 11:32    2   0%  -ilEo-b----f  li VOL006 MEDIA ERROR
   6     2016/12/13 18:07    2   0%  -il-o-b-----  li VOL007
   7     2016/12/13 18:07    1   0%  -il-o-b-----  li VOL008
   8     2016/12/13 18:07    1   0%  -il-o-b-----  li VOL009
...
  18     2016/12/13 18:16    1   0%  -il-o-b-----  li VOL019
  19     none               50   0%  -il-oCb-----  li CLN020
```

3. To determine the status of a volume using the **samcmd v** display, examine the entry in the **flags** column and consult the list below:

   - **A** means that the slot needs an audit.
   - **C** means that the slot contains a cleaning cartridge.
   - **E** means that the volume is bad or the cleaning media has expired.
   - **L** means that the volume is a Linear Tape File System (LTFS) volume.
   - **N** means that the volume is foreign media (not in Oracle HSM format).
   - **R** means that the volume is read-only (a software flag).
   - **U** means that the volume is unavailable.
   - **W** means that the volume is physically write-protected.
   - **X** means that the slot is an export slot.
   - **b** means that the volume has a bar code.
   - **c** means that the volume is scheduled for recycling.
   - **f** means that the archiver found the volume full or corrupted.
   - **d** means that the volume has a duplicate volume serial number (VSN).
   - **l** means that the volume is labeled.
   - **o** means that the slot is occupied.
   - **p** means that the volume is a high priority volume.
   - **-** means that the corresponding flag is not set.

4. To identify the type of media used for a volume using the **samcmd v** display, consult the **ty** column and look up the code displayed in Appendix A or in the **mcf** man page.

5. To list all information in the catalog, use the command **dump_cat** *catalog-path-name*, where *catalog-path-name* is the path and file name of the catalog file, as specified in the **/etc/opt/SUNWsamfs/mcf** file.

   In the example, we dump the catalog file **catalog/800_cat**.

   ```
   root@mds1:~# dump_cat catalog/800_cat
   # audit_time Wed Dec 31 17:00:00 1969
   ```

```
# version 530  count 32 mediatype
#Index VSN    Barcode  Type PTOC  Access Capacity ...  LVTime LVPos
#
 0     S00001 S00001L4 li   0x747    875   512000 ...       0   0x3
 1     S00002 S00002L4 li   0x5db    866   512000 ...       0   0x3
13     S00014 S00014L4 li       0      4   512000 ...       0   0
17     S00018 S00018L4 li       0      1   512000 ...       0   0
18     S00003 S00003L4 li       0      3   512000 ...       0   0
```

6. Stop here.

Audit the Contents of a Library Slot

To update the library catalog with the reported space remaining on a removable media volume, audit the library slot. Use the command **auditslot**.

1. Log in to the file system host as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

2. To audit a specified tape volume, skip to EOD (*end of data*), and update the space available, use the command **auditslot -e** *library-equipment-number*:*slot*, where *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the library and *slot* is the location of the cartridge within the library.

   The **auditslot** command loads the cartridge that contains the volume, reads the label, and updates the library catalog entry for the slot. Note that you cannot interrupt skipping to EOD once you start it, and, under certain conditions, it can take hours to complete. In the example we audit slot **11** in tape library **800**:

   ```
   root@mds1:~# auditslot -e 800:11
   root@mds1:~#
   ```

3. To audit a specified optical volume, use the command **auditslot** *library-equipment-number*:*slot*[:*side*], where *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the library, *slot* is the location of the cartridge within the library, and *side* (optional) is the specified side of a two-sided optical disk.

   In the example we audit side **1** of the volume in slot **21** of optical library **700**:

   ```
   root@mds1:~# auditslot 800:21:1
   root@mds1:~#
   ```

4. Stop here.

Audit the Entire Direct-Attached Automated Library

A full audit loads each cartridge into a drive, reads the label, and updates the library catalog. Audit a library in the following situations:

- after moving cartridges in the automated library without using Oracle HSM commands
- when the library catalog may be unreliable (following a power outage, for example)
- when you have added, removed, or moved cartridges in an automated library that has no mailbox.

To perform a full audit, use the command **samcmd audit** *library-equipment-number*, where *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the library.

Note that a full audit can take a long time, depending on the number of slots that contain media.

In the example we audit tape library **800**:

```
root@mds1:~# audit 800
root@mds1:~#
```

Clear a Media Error from the Catalog

When Oracle HSM has problems using a removable media cartridge, it sets an error flag on the corresponding catalog entry. The media may be worn, damaged, or, the case of cleaning media, expired. In such cases, the media should not be reused. But problems accessing media can also result from faults in the drive, in which case the media can be reused without difficulty. In the latter case, you need to clear the error flag for the cartridge.

Be sure that you know the nature of the problem before clearing error flags. Error flags are critical to Oracle HSM operations and to the security of your data. You do not want to clear this flag if a cartridge is actually faulty.

Once you are sure, you can clear the error and try to use the cartridge. Proceed as follows:

1. Log in to the file system host as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

2. Check the status of removable media volumes. Use the command **samcmd r**.

   In the example, the **samcmd r** command shows that drive **801** has set the error flag on LTO (**li**) volume **VOL004**.

   ```
   root@mds1:~# samcmd r
   Removable media status: all          samcmd 5.4        17:40:11 Mar 13 2014
   ty   eq  status      act  use  state   vsn
   li  801 -E-------r     0   0%  notrdy  VOL004  MEDIA ERROR
           MEDIA ERROR
   li  802  ---------p    0   0%  notrdy
           empty
   li  803  ---------p    0   0%  notrdy
           empty
   li  804  ---------p    0   0%  notrdy
           empty
   root@mds1:~#
   ```

3. If the drive that set the error flag is suspect, unload the cartridge and clear the error flag. Use the command **samcmd unload** *drive-number*, where *drive-number* is the equipment-ordinal number specified for the drive in the **/etc/opt/SUNWsamfs/mcf** file.

   In the example, we unload drive **801**:

   ```
   root@mds1:~# samcmd unload 801
   ```

4. To clear the media error flag for a specified volume serial number and media type, use the command **chmed -E** *media-type*.*volume-serial-number*, where *mediatype* is one of the two-character media type codes listed in Appendix A and *volume-serial-number* is the alphanumeric string that uniquely identifies the volume within the library.

   In the example, we clear the error flag on LTO (**li**) volume **VOL004**:

```
root@mds1:~# chmed -E li.VOL004
   3:0 li VOL004     Ail---b-----    2.3T    2.3T    0          0 800 4  0 //
root@mds1:~#
```

5. To clear the media error flag for a cartridge that resides in a specified library slot, use the command **chmed -E** *library-equipment-number*:*slot*[:*disk-side*], where *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the automated tape library, *slot* is the slot address where the target volume resides within the library, and the optional *disk-side* value, either **1** or **2**, specifies one of the sides of a two-sided magneto-optical disk.

   In the example, we clear the error flag on the cartridge in slot **4** of library **800**:

```
root@mds1:~# chmed -E 800:4
   3:0 li VOL004     Ail---b-----    2.3T    2.3T    0          0 800 4  0 //
root@mds1:~#
```

6. Update the library catalog to reflect the change. Use the command **auditslot -e** *library-equipment-number*:*slot*[:*disk-side*].

   In the example, we update the catalog by auditing slot **4** of library **800**:

```
root@mds1:~# auditslot -e 800:4
root@mds1:~#
```

7. Mount the cartridge in a different drive, and see if the error recurs. Use the command **samcmd load** *media-type*.*volume-serial-number*, where *mediatype* is one of the two-character media type codes listed in Appendix A and *volume-serial-number* is the alphanumeric string that uniquely identifies the volume within the library.

```
root@mds1:~# samcmd load li.VOL004
root@mds1:~#
```

8. Re-check the status of removable media volumes. Use the command **samcmd r**.

```
root@mds1:~# samcmd r
Removable media status: all          samcmd 5.4         17:42:10 Mar 13 2014
ty   eq  status      act  use  state  vsn
li  801  ---------p    0   0%  notrdy
        empty
li  802  --l------r    0   0%  ready   VOL004
        idle
li  803  ---------p    0   0%  notrdy
        empty
li  804  ---------p    0   0%  notrdy
        empty
root@mds1:~#
```

9. If the error does not recur on the new drive, the cartridge is probably OK.

10. If the error recurs, consider retiring the removable media volume.

11. Stop here.

Managing the Historian Catalog

The Oracle Hierarchical Storage Manager *historian* is a pseudo-library that has a catalog but no equipment. The historian catalogs volumes that are no longer under direct Oracle HSM control. It thus maintains a record of any volumes that have been exported from a library and sent for offsite storage and volumes that are hand-loaded into standalone drives. Oracle HSM automatically updates the historian catalog when you export volumes from the library. But you can also use the historian for manual record keeping by adding and/or removing records and attaching notes. In general, you interact with the historian much as you would with a physical media library:

This section outlines the following tasks:

- viewing the historian catalog
- adding entries to the historian catalog
- removing entries from the historian catalog
- updating the information field of a catalog record.

View the Historian Catalog

You view the historian catalog exactly as you would that of a physical library. Use the command **samcmd v** *historian-equipment-number*, where *historian-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the historian.

In the example, we display the catalog for a historian that has the equipment ordinal number **900**:

```
root@mds1:~# samcmd v 900
Robot catalog samcmd     5.4    16:45:25 Mar 14 2014
samcmd on samqfshost                          count 32
Robot VSN catalog by slot       : eq 900
slot          access time count use  flags         ty vsn
   0     2016/12/14 11:23  875   0%  -il-o-b-----  li EXT001
   1     2016/12/13 17:54  866   0%  -il-o-b-----  li EXT002
...
root@mds1:~#
```

Add an Entry to the Historian Catalog

To add an entry to the historian catalog, proceed as follows:

1. To add an entry to the historian catalog for a specified volume serial number, use the command **samimport -v** *volume-serial-number* **-m** *mediatype historian-equipment-number*, where:

   - *volume-serial-number* is the alphanumeric string that uniquely identifies the volume within the catalog.
   - *mediatype* is one of the two-character media type codes listed in Appendix A.
   - *historian-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the historian.

   In the example, we add a record for the LTO (**li**) volume **EXT003** to the catalog for historian **900**:

   ```
   root@mds1:~# samimport -v EXT003 -m li 900
   ...
   ```

```
root@mds1:~#
```

2. To add an entry to the historian catalog for a specified barcode, use the command **samimport -b** *barcode* **-m** *mediatype historian-equipment-number*, where *barcode* is the barcode affixed to the corresponding physical cartridge.

   In the example, we add a record for the LTO (**li**) volume with barcode **EXT003L4** to the catalog for historian **900**:

```
root@mds1:~# samimport -b EXT003L4 -m li 900
...
root@mds1:~#
```

3. Stop here.

Remove an Entry from the Historian Catalog

To remove an entry from the historian catalog, use the command **samexport** *historian-equipment-number*:*slot*, where *historian-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the historian and *slot* is the historian slot address for the record.

In the example, we remove the record for volume **EXT002** in slot **1** of the catalog for historian **900**:

```
root@mds1:~# samcmd v 900
Robot catalog samcmd     5.4    16:45:25 Mar 14 2014
samcmd on samqfshost                          count 32
Robot VSN catalog by slot       : eq 900
slot          access time count use  flags         ty vsn
   0     2016/12/14 11:23  875   0%  -il-o-b-----  li EXT001
   1     2016/12/13 17:54  866   0%  -il-o-b-----  li EXT002
   2     2016/12/13 17:57  866   0%  -il-o-b-----  li EXT003
...
root@mds1:~# samexport 900:1
...
root@mds1:~#
```

Update Historian Information

You can update the information field in a historian catalog entry to note changes to the disposition or status of an exported volume. Use the command **chmed -I "***note***"** *identifier*, where *note* is a string of up to 128 characters and *identifier* is either of the following:

- *mediatype*.*volume-serial-number*, where *mediatype* is one of the two-character media type codes listed in Appendix A and *volume-serial-number* is the alphanumeric string that uniquely identifies the volume within the library. Or use the command
- *library-equipment-number*:*slot*, where *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the automated tape library and *slot* is the slot address where the cartridge resides within the library.

In the example, we note that LTO (**li**) cartridge **VOL06E** has been recalled from the vault, successfully validated, and returned to the vault:

```
root@mds1:~# chmed -I "validated and revaulted 20150310" li.VOL06A
```

Determining the Order in Which Drives are Installed in the Library

If your automated library contains more than one drive, the order of the drives in the **mcf** file must be the same as the order in which the drives are seen by the library controller. This order can be different from the order in which devices are seen on the host and reported in the host's **/var/adm/messages** file. So whenever you configure an Oracle Hierarchical Storage Manager metadata server and datamover host, change libraries, or change the configuration of a library, you should check the drive order by carrying out the tasks listed below:

- gathering drive information for the library and the Solaris host
- mapping the drives in a direct-attached or ACSLS-attached library to Solaris device names (depending on the equipment that you are using).

Gather Drive Information for the Library and the Solaris Host

1. Consult the library documentation. Note how drives and targets are identified. If there is a local operator panel, see how it can be used to determine drive order.
2. If the library has a local operator panel mounted on the library, use it to determine the order in which drives attach to the controller. Determine the SCSI target identifier or World Wide Name of each drive.
3. Log in to the Solaris host as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

4. List the Solaris logical device names in **/dev/rmt/**, redirecting the output to a text file.

   In the example, we redirect the listings for **/dev/rmt/** to the file **device-mappings.txt** in the **root** user's home directory:

   ```
   root@mds1:~# ls -l /dev/rmt/ > /root/device-mappings.txt
   ```

5. Now, map the drives to Solaris device names, using the procedure specific to your equipment: direct-attached tape library or ACSLS-attached library.

Map the Drives in a Direct-Attached Library to Solaris Device Names

For each Solaris logical drive name listed in **/dev/rmt/** and each drive that the library assigns to the Oracle HSM server host, carry out the following procedure:

1. If you are not already logged in to the Oracle HSM Solaris host, log in as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

2. Stop all running archiving processes so that drives are not in use. See "Idle Archiving and Staging Processes" and "Stop Archiving and Staging Processes" on page 3-24.
3. In a text editor, open the device mappings file that you created. Organize the file into a simple table, and save the changes.

   You will need to refer to this information in subsequent steps. In the example, we have used the **vi** editor to delete the permissions, ownership, and date attributes from the **/dev/rmt/** list, while adding headings and space for library device information:

```
root@mds1:~# vi /root/device-mappings.txt
LIBRARY SOLARIS          SOLARIS
DEVICE  LOGICAL          PHYSICAL
NUMBER  DEVICE           DEVICE
------- -------------    --------------------------------------------------
        /dev/rmt/0cbn -> ../../devices/pci@8.../st@w500104f00093c438,0:cbn
        /dev/rmt/1cbn -> ../../devices/pci@8.../st@w500104f0008120fe,0:cbn
        /dev/rmt/2cbn -> ../../devices/pci@8.../st@w500104f000c086e1,0:cbn
        /dev/rmt/3cbn -> ../../devices/pci@8.../st@w500104f000b6d98d,0:cbn
:w
```

4. On the library, make sure that all drives are empty.
5. Load a tape into the first drive in the library that you have not yet mapped to a Solaris logical device name.

   For the purposes of the examples below, we load an LTO4 tape into an HP Ultrium LTO4 tape drive.
6. If you are mapping the drives in a tape library, identify the Solaris **/dev/rmt/** entry that corresponds to the drive that mounts the tape. Until you identify the drive, run the command **mt -f /dev/rmt/***number* **status** where *number* identifies the drive in **/dev/rmt/**.

   In the example, the drive at **/dev/rmt/0** is empty, but the drive at **/dev/rmt/1** holds the tape. So the drive that the library identifies as drive 1 corresponds to Solaris **/dev/rmt/1**:

   ```
   root@mds1:~# mt -f /dev/rmt/0 status
   /dev/rmt/0: no tape loaded or drive offline
   root@mds1:~# mt -f /dev/rmt/1 status
   HP Ultrium LTO 4 tape drive:
      sense key(0x0)= No Additional Sense   residual= 0   retries= 0
      file no= 0   block no= 3
   ```
7. In the device-mappings file that you created in the previous procedure, locate the entry for the Solaris device that holds the tape, and enter the library's device identifier in the space provided. Then save the file.

   In the example, we enter **1** in the **LIBRARY DEVICE NUMBER** field of the row for **/dev/rmt/1**:

   ```
   root@mds1:~# vi /root/device-mappings.txt
   LIBRARY SOLARIS          SOLARIS
   DEVICE  LOGICAL          PHYSICAL
   NUMBER  DEVICE           DEVICE
   ------- -------------    --------------------------------------------------
           /dev/rmt/0cbn -> ../../devices/pci@8.../st@w500104f00093c438,0:cbn
      1    /dev/rmt/1cbn -> ../../devices/pci@8.../st@w500104f0008120fe,0:cbn
           /dev/rmt/2cbn -> ../../devices/pci@8.../st@w500104f000c086e1,0:cbn
           /dev/rmt/3cbn -> ../../devices/pci@8.../st@w500104f000b6d98d,0:cbn
   :w
   ```
8. Unload the tape.
9. Repeat this procedure until the device-mappings file holds entries that map all devices to Solaris logical device names. Then save the file and close the editor.

   ```
   root@mds1:~# vi /root/device-mappings.txt
   LIBRARY SOLARIS          SOLARIS
   DEVICE  LOGICAL          PHYSICAL
   NUMBER  DEVICE           DEVICE
   ------- -------------    --------------------------------------------------
   ```

```
   2    /dev/rmt/0cbn -> ../../devices/pci@8.../st@w500104f00093c438,0:cbn
   1    /dev/rmt/1cbn -> ../../devices/pci@8.../st@w500104f0008120fe,0:cbn
   3    /dev/rmt/2cbn -> ../../devices/pci@8.../st@w500104f000c086e1,0:cbn
   4    /dev/rmt/3cbn -> ../../devices/pci@8.../st@w500104f000b6d98d,0:cbn
:wq
root@mds1:~#
```

10. Stop here. Keep the mappings file for later use.

Map the Drives in an ACSLS-Attached Library to Solaris Device Names

1. If you are not already logged in to the Oracle HSM Solaris host, log in as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

2. Stop all running archiving processes, so that drives are not in use. See "Idle Archiving and Staging Processes" and "Stop Archiving and Staging Processes".

3. In a text editor, open the device mappings file that you created. Organize the file into a simple table.

   You will need to refer to this information in subsequent steps. In the example, we are using the **vi** editor to delete the permissions, ownership, and date attributes from the **/dev/rmt/** list, while adding headings and space for library device information:

   ```
   root@mds1:~# vi /root/device-mappings.txt
   SOLARIS LOGICAL DEVICE  DEVICE SERIAL NUMBER  ACSLS DEVICE ADDRESS
   ----------------------  --------------------  -------------------------------
   /dev/rmt/0
   /dev/rmt/1
   /dev/rmt/2
   /dev/rmt/3
   ```

4. For each logical device name listed in **/dev/rmt/**, display the serial number using the command **luxadm display /dev/rmt/***number*, where *number* identifies the drive in **/dev/rmt/**.

   In the example, we obtain **HU92K00200**, the serial number of device **/dev/rmt/0**:

   ```
   root@mds1:~# luxadm display /dev/rmt/0
   DEVICE PROPERTIES for tape: /dev/rmt/0
   Vendor: HP
   Product ID: Ultrium 4-SCSI
   Revision: G25W
   Serial Num: HU92K00200
   ...
   Path status: Ready
   root@mds1:~#
   ```

5. Then, using a text editor, enter the serial number of each device in the corresponding row of your **device-mappings.txt** file.

   In the example, we record the serial number for device **/dev/rmt/0** in the **device-mappings.txt** file using the **vi** editor:

   ```
   root@mds1:~# vi /root/device-mappings.txt
   SOLARIS LOGICAL DEVICE  DEVICE SERIAL NUMBER  ACSLS DEVICE ADDRESS
   ----------------------  --------------------  -------------------------------
   /dev/rmt/0              HU92K00200
   /dev/rmt/1
   ```

```
/dev/rmt/2
/dev/rmt/3
```

6. For each device serial number mapped to **/dev/rmt/**, obtain the corresponding ACSLS drive address. Use the ACSLS command **display drive * -f serial_num**.

   In the example, we obtain the ACSLS addresses of devices **HU92K00200** (**/dev/rmt/0**), **HU92K00208** (**/dev/rmt/1**), **HU92K00339** (**/dev/rmt/2**), **HU92K00289** (**/dev/rmt/3**):

```
ACSSA> display drive * -f serial_num
2014-03-29 10:49:12 Display Drive
Acs Lsm Panel Drive Serial_num
0   2   10    16    331002031352
0   2   10    17    HU92K00200
0   2   10    18    HU92K00208
0   3   10    10    HU92K00339
0   3   10    11    HU92K00189
0   3   10    12    HU92K00289
root@mds1:~#
```

7. Using a text editor, enter the ACSLS address for each serial number in the corresponding row of the **device-mappings.txt** file. Save the file, and close the editor.

   In the example, we record the information in the **device-mappings.txt** file using the **vi** editor:

```
root@mds1:~# vi /root/device-mappings.txt
SOLARIS LOGICAL DEVICE DEVICE SERIAL NUMBER ACSLS DEVICE ADDRESS
---------------------- -------------------- --------------------------------
/dev/rmt/0             HU92K00200           (acs=0, lsm=2, panel=10, drive=17)
/dev/rmt/1             HU92K00208           (acs=0, lsm=2, panel=10, drive=18)
/dev/rmt/2             HU92K00339           (acs=0, lsm=2, panel=10, drive=10)
/dev/rmt/3             HU92K00289           (acs=0, lsm=2, panel=10, drive=12)
:wq
root@mds1:~#
```

8. Stop here. Keep the mappings file for later use.

Managing Drives

You can handle a variety of drive management tasks from the Oracle HSM interfaces, including the following:

- loading and unloading drives
- cleaning tape drives
- managing drives that have encryption capability
- handling drive problems.

Loading and Unloading Drives

When removable media are stored in automated libraries, file-system archiving and staging processes automatically load cartridges into drives as required. But you can also load cartridges on demand when managing removable media files, backing up the Oracle HSM configuration, or recovering a file system. This section covers the following tasks:

- loading and unloading drives installed in an automated library
- manually loading and unloading standalone drives
- notifying operators when volumes must be loaded manually.

Loading and Unloading Drives Installed in an Automated Library

This section covers the following tasks:

- loading a drive from a specified library location
- loading a drive with a specified media type and volume serial number (VSN)
- unloading a specified drive.

Load a Drive from a Specified Library Location

Use the command **samcmd load** *library-equipment-number*:*slot*[:*disk-side*], where *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the automated tape library, *slot* is the slot address where the target volume resides within the library, and the optional *disk-side* value, either **1** or **2**, specifies one of the sides of a two-sided magneto-optical disk.

The cartridge is loaded in the next available drive in the library. In the example, we load the magnetic tape cartridge located in slot **11** of library **800**:

```
root@mds1:~# samcmd load 800:11
```

Load a Drive with a Specified Media Type and Volume Serial Number

Use the command **samcmd load** *mediatype*.*volume-serial-number*, where *mediatype* is one of the two-character media type codes listed in Appendix A and *volume-serial-number* is the alphanumeric string that uniquely identifies the volume within the library.

The cartridge is loaded in the next available drive in the library. In the example, we load the LTO (**li**) tape cartridge **VOL109**:

```
root@mds1:~# samcmd load li.VOL109
```

Unload a Specified Drive in the Library

Use the command **samcmd unload** *drive-equipment-number*, where *drive-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the drive.

The cartridge is unloaded, even if the drive is **unavail**. In the example, we unload drive **801**:

```
root@mds1:~# samcmd unload 801]
```

Manually Loading and Unloading Standalone Drives

The Oracle HSM software treats standalone, removable-media drives as if they were small, single-slot libraries with their own catalogs.

Load a Cartridge Into a Standalone Drive

To load a standalone drive, place the cartridge in the drive according to the manufacturer's instructions. The Oracle HSM system recognizes that the cartridge is loaded, reads the label, and updates the catalog for the drive.

Unload a Cartridge from a Standalone Drive

To unload a standalone drive, proceed as follows:

1. Idle the drive. Use the command **samcmd idle** *drive-equipment-number*, where *drive-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the drive.

   When a drive is idled, the Oracle HSM software finishes any current archiving processes that use the drive but does not start any new ones.

   ```
   root@mds1:~# samcmd idle 801
   ```

2. Wait until Oracle HSM finishes and turns the drive **off**.

   You can check on the status of the drive using the command **samcmd r**.

3. Remove the cartridge according to the vendor's instructions.

4. Stop here.

Notifying Operators When Volumes Must Be Loaded Manually

If you are using a standalone drive or if you store required cartridges in a vault or some other location outside the library, the Oracle HSM software can send email to a specified address when it needs an operator to load a non-resident cartridge. To enable this feature, follow the procedure below:

Enable Load Notification

1. Log in to the file system host as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

2. Copy the file **load_notify.sh** from the directory **/opt/SUNWsamfs/examples/** to the directory **/etc/opt/SUNWsamfs/scripts/**.

   Note that the command below is entered as a single line—the line break is escaped by the backslash character:

   ```
   root@mds1:~# cp /opt/SUNWsamfs/examples/load_notify.sh \
   /etc/opt/SUNWsamfs/scripts/
   root@mds1:~#
   ```

3. Open the file **/etc/opt/SUNWsamfs/defaults.conf** in a text editor. Search for the directive **exported_media**. Uncomment the line or add it, if necessary, and set its value to **exported_media=available**.

   In the example, we use the **vi** editor:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
   # These are the defaults.
   ...
   exported_media=available
   ```

4. In the file **/etc/opt/SUNWsamfs/defaults.conf**, search for the directive **attended**. Uncomment the line or add it, if necessary. Set its value to **attended=yes**. Save the file, and close the editor.

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
   # These are the defaults. ...
   # These are the defaults. ...
   exported_media=available
   attended=yes
   ```

```
:wq
root@mds1:~#
```

5. Open the file **/etc/opt/SUNWsamfs/scripts/load_notify.sh** in a text editor. Locate the default recipient of the notification email, **root**.

```
root@mds1:~# vi /etc/opt/SUNWsamfs/scripts/load_notify.sh
#       Notify operator to load volume.
...
# Change the email address on the following line to send email to
# the appropriate recipient.
/bin/ppriv -s I=basic -e /usr/bin/mailx -s "SAM-FS needs VSN $5" root <<EOF
...
```

6. In the file **/etc/opt/SUNWsamfs/scripts/load_notify.sh**, change the recipient of the notification email from the default, **root**, to the email address of the operator responsible for the non-resident volumes. Save the file, and close the editor.

   In the example, we change the recipient to **tapetech**:

```
#       Notify operator to load volume.
...
/bin/ppriv -s I=basic -e /usr/bin/mailx -s "SAM-FS needs VSN $5" tapetech <<EOF
...
:wq
root@mds1:~#
```

7. Reinitialize the Oracle HSM software. Use the **sam-fsd** command.

   The **sam-fsd** is an initialization command that reads Oracle HSM configuration files. It will stop if it encounters an error. In the example, no errors are found:

```
root@mds1:~# sam-fsd
...
Would start sam-archiverd()
Would start sam-stagealld()
Would start sam-stagerd()
Would start sam-amld()
root@mds:~#
```

8. Tell the Oracle HSM software to re-read the **mcf** file and reconfigure file systems and hardware accordingly. Use the command **samd config**:

```
root@mds1:~# samd config
Configuring SAM-FS ...
root@mds1:~#
```

9. Stop here.

Cleaning Tape Drives

Modern, Oracle StorageTek T10000D and Linear Tape Open (LTO) tape drives self-monitor and request cleaning when and as needed. The Oracle Hierarchical Storage Manager software honors these requests and automatically loads a cleaning cartridge when required. So in most cases, you need only insure that your library contains adequate cleaning cartridges and that Oracle HSM is able to locate them.

When drive-requested cleaning is not feasible, you can initiate cleaning manually. But be advised: most manufacturers emphatically discourage routine cleaning in the absence of a request from the drive. Cleaning cartridges are abrasive. Overuse can

damage drives and media. So exercise caution, and follow the manufacturer's recommendations.

The remainder of this section provides instructions for the following tasks:

- providing sufficient cleaning cartridges
- enabling automatic tape-drive cleaning (recommended)
- cleaning a tape drive manually.

Provide Sufficient Cleaning Cartridges

1. Log in to the file system host as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

2. If you plan to configure automatic cleaning (recommended) and if your library has more than two drives, make sure that you provide at least two cleaning cartridges for each file-system catalog that lists tapes in the library.

   If a cleaning cartridge is unavailable when a drive requires cleaning, the Oracle HSM software sets the drive state to **down** until cleaning can be completed.

3. Place the cleaning cartridge(s) in the library mail slot (also known as the cartridge access port).

4. Import the cleaning cartridge(s) into the automated library. Use the command **samimport** *library-equipment-number*, where *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the library.

   In the example, we place cleaning cartridges in the mail slot of library **800** and import them into the library:

   ```
   root@mds1:~# samimport 800
   ...
   root@mds1:~#
   ```

5. If the cleaning cartridge label reads **CLEAN** or starts with the letters **CLN**, stop here.

   The Oracle HSM software recognizes the cleaning cartridge and moves it from the mailbox to a storage slot. Oracle HSM updates the library catalog, sets the cleaning media flag, and sets the access count to the maximum number of cleanings recommended for the media type (each time the cartridge is used to clean a drive, this count decrements).

6. If the cartridge is not labeled, flag it as cleaning media. Use the command **chmed +C** *library-equipment-number*:*slot*, where *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the library and *slot* is the location of the cleaning cartridge within the library.

   In the example, we set the **C** (cleaning-media) flag on the cartridge in slot **31** of library **800**.

   ```
   root@mds1:~# chmed +C 800:31
   ...
   root@mds1:~#
   ```

7. Set the access count to the maximum number of cleanings recommended for the media type. Use the command **chmed -count** *cleanings* *library-equipment-number*:*slot*, where:

- *cleanings* is the maximum number of cleanings that the manufacturer recommends per cartridge.
- *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the library.
- *slot* is the location of the cleaning cartridge within the library.

Each time the cartridge is used to clean a drive, the cleaning count decrements. In the example, we set the count to a maximum of **50** cleanings, the maximum recommended for the LTO (type **li**) cleaning cartridges used in library **800**:

```
root@mds1:~# chmed -count 50 800:31
...
root@mds1:~#
```

8. Next, enable automatic tape-drive cleaning (recommended) or stop here.

Enable Automatic Tape-Drive Cleaning (Recommended)

1. Log in to the file system host as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

2. If your library includes an Auto Clean feature that you wish to use, configure the feature according to the library manufacturer's recommendations. Stop here.

   Now, when drives request cleaning, the library will automatically supply the required cleaning media.

3. If your library includes an Auto Clean feature that you *do not* wish to use, disable the feature according to the manufacturer's recommendations.

4. Open the file **/etc/opt/SUNWsamfs/defaults.conf** in a text editor, and enable Oracle HSM automatic cleaning. Add the line **tapeclean = all autoclean on logsense on**. Then save the file and close the editor.

   In the example, we use the **vi** editor:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
   # These are the defaults. ...
   #sef = all on once
   ...
   tapeclean = all autoclean on logsense on
   :wq
   root@mds1:~#
   ```

5. Reinitialize the Oracle HSM software. Use the **sam-fsd** command.

   The **sam-fsd** is an initialization command that reads Oracle HSM configuration files. It will stop if it encounters an error. In the example, no errors are found:

   ```
   root@mds1:~# sam-fsd
   ...
   Would start sam-archiverd()
   Would start sam-stagealld()
   Would start sam-stagerd()
   Would start sam-amld()
   root@mds:~#
   ```

6. Tell the Oracle HSM software to re-read the **mcf** file and reconfigure file systems and hardware accordingly. Use the command **samd config**:

```
root@mds1:~# samd config
...
root@mds1:~#
```

7. Stop here.

Clean a Tape Drive Manually

1. Check the drive manufacturer's guidelines for manual cleaning before proceeding.

   Exercise caution. Over-frequent cleaning is a common cause of drive damage. Many manufacturers now strongly discourage routine or scheduled cleanings. So make sure that you understand when your drives need to be cleaned.

2. Monitor the device logs for indications that drives need cleaning. There is one log in the directory **/var/opt/SUNWsamfs/devlog/** for each *drive-equipment-number*, where *drive-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the drive.
3. Monitor the system log file **/var/adm/messages** for device errors.
4. Clean the tape drive. Use the command **cleandrive** *drive-equipment-number*.

   In the example, we clean drive **802**:

   ```
   root@mds1:~# cleandrive 802
   ...
   root@mds1:~#
   ```

5. Stop here.

Managing Drives that Have Encryption Capability

If you are archiving files to drives that have encryption capability, consider the following points when planning archiving operations:

- Do not mix encryption-capable and non-encryption-capable drives in a library.
- After a drive has encryption enabled, encryption cannot be disabled.
- Do not mix encrypted and non-encrypted files on a tape volume.
- An encryption-enabled drive cannot append encrypted files to a tape volume that contains non-encrypted data.
- An encryption-enabled drive can read non-encrypted data.

For further information, consult the documentation for your drives and your encryption key-management system.

For information on encryption-capable cloud drives, see "Managing Cloud Storage" on page 5-32.

Handling Drive Problems

In general, you handle drive problems according to the vendor's recommendations. But before you can start drive maintenance, troubleshooting, or repair, you may need to perform one or both of the following tasks:

- taking a drive offline for maintenance or repair
- return media to the library following a drive problem.

Take a Drive Offline for Maintenance or Repair

1. Log in to the file system host as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

2. Finish up active archiving and staging jobs and keep any new jobs from starting. See "Idle Archiving and Staging Processes" on page 3-23 and "Stop Archiving and Staging Processes" on page 3-24.
3. Carry out the vendor-specified maintenance, diagnostic. or repair procedures.

   For example, before attempting to remove a stuck cartridge, be sure to check the vendor's recommendations. Improperly removing a stuck cartridge can damage the cartridge and the drive.

4. When the drive is again operational, bring the library and drives online and restart archiving and staging processes. Use the procedure "Restart Archiving and Staging Processes" on page 3-25.
5. Stop here.

Return Media to the Library Following a Drive Problem

If drive problems occur with media mounted in the drive, you may need to remove the media manually as part of the repair process. This can leave the catalog inconsistent. So follow the appropriate procedure below:

Return Media to a Library that Has Not Performed an Automatic Audit

To return media to a Library does not perform an automatic audit when the library and drive are brought back online following repairs, proceed as follows:

1. Return the cartridge to its storage slot by hand.

   In this case, the catalog has not been updated and continues to list the cartridge among the library contents. So you correct the discrepancy by putting the cartridge back in the same slot that it previously occupied.

2. Log in to the file system host as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

3. Update the Oracle HSM catalog to show that the slot is again occupied. Use the command **chmed** *library-equipment-number*:*slot*, where *slot* is the address of the slot within the library.

   In the example, we update the status of slot **42** in library **800**:

   ```
   root@mds1:~# chmed +o 800:42
   root@mds1:~#
   ```

4. Stop here.

Returning Media to a Library After an Automatic Audit

If the library performs an automatic audit when the library and drive are brought back online following repairs, proceed as follows:

1. Place the cartridge in the library mail slot.

2. Log in to the file system host as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

3. Import the cartridge into the library. Use the command **samimport** *library-equipment-number*.

   In this case, the audit has reconciled the catalog, which no longer lists the cartridge in the library. So importing the cartridge adds it to both the library and the Oracle HSM catalog. In the example, we have placed the cartridge in the mailslot of library **800** and imported it into the library.

   ```
   root@mds1:~# samimport 800
   ```

4. Stop here.

Managing Removable Media

This section covers the following topics:

- labeling removable media
- maintaining data integrity.

Labeling Removable Media

> **Caution:** Labeling or relabeling a cartridge renders any data on the cartridge permanently inaccessible. Relabel a cartridge only if you are certain that you do not need the data that is stored on it.

The labeling process writes identifying information on the recording media and initializes it for use (see ANSI X3.27-1987, *File Structure and Labeling of Magnetic Tapes for Information Interchange*, for full information).

When you need to label media, select the appropriate procedure below:

- generating labels from barcodes
- labeling a new tape or relabeling an existing tape
- labeling a new optical disk or relabel an existing optical disk.

Generate Labels from Barcodes

To automatically label write-enabled, unlabeled cartridges with a volume serial number (VSN) derived from the barcodes on the cartridges, proceed as follows.

1. Make sure that all barcodes are readable.
2. Log in to the file system host as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

3. Open the **/etc/opt/SUNWsamfs/defaults.conf** file in a text editor.

   In the example, we use the **vi** editor to view the file:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
   ```

```
# These are the defaults.
...
```

4. If you need to generate volume serial numbers (VSNs) from the first six characters in the corresponding barcodes, first see if Oracle HSM is set to the default value, **barcodes**. In the **defaults.conf** file, locate the line for the **labels** directive, if present. If the **labels** directive is set to **barcodes**, is commented out, or is not present in the file, then Oracle HSM is set to the default value, **barcodes**.

   In the example, the **defaults.conf** file contains the line **#labels = barcodes**:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
# These are the defaults.
...
#labels = barcodes
root@mds1:~#
```

5. If you need to generate volume serial numbers (VSNs) from the first six characters in the corresponding barcodes and if Oracle HSM is set to the default value, close the **defaults.conf** files without making any changes. Stop here.

   When the **labels** directive is set to **barcodes**, the software automatically generates the required volume serial numbers (VSNs) from the first six characters in the corresponding barcodes. In the example, Oracle HSM is using the default setting. So we close the **vi** editor without saving the file:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
# These are the defaults.
...
#labels = barcodes
:q
root@mds1:~#
```

6. Otherwise, if you need to generate volume serial numbers (VSNs) from the first six characters in the corresponding barcodes, enter **labels = barcodes**, comment out the non-default directive, or simply delete the non-default directive. Then save the file, and close the editor.

   In the example, the directive has been set to the non-default value **barcodes_low**. So we comment out the non-default line. We insert the line **labels = barcodes**. We save the modified file, and close the editor:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
# These are the defaults.
...
#labels = barcodes_low
labels = barcodes
:q
root@mds1:~#
```

7. If you need to generate the volume serial number (VSN) from the last six characters in the cartridge's barcode, set the value of the **labels** parameter to **barcodes_low**. Save the file, and close the editor.

   In the example, we insert the line **labels = barcodes_low**, save the file, and close the editor:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
# These are the defaults.
...
labels = barcodes_low
:wq
```

```
root@mds1:~#
```

8. If you edited the **defaults.conf** file, run the **sam-fsd** command.

   The **sam-fsd** is an initialization command that reads Oracle HSM configuration files. It will stop if it encounters an error. In the example, no errors are found:

```
root@mds1:~# sam-fsd
...
Would start sam-archiverd()
Would start sam-stagealld()
Would start sam-stagerd()
Would start sam-amld()
root@mds:~#
```

9. If you edited the **defaults.conf** file, tell the Oracle HSM software to re-read the **mcf** file and reconfigure itself accordingly. Use the command **samd config**.

```
[metadata-server]root@mds1:~# samd config
```

10. Stop here.

Label a New Tape or Relabel an Existing Tape

> **Caution:** Labeling or relabeling a cartridge renders any data on the cartridge permanently inaccessible. Relabel a cartridge only if you are certain that you do not need the data that is stored on it.

1. Log in to the file system host as **root**.

   In the example, the host is named **mds1**:

```
root@mds1:~#
```

2. To label a new tape that is already loaded into a drive, use the command **tplabel -new** *volume-serial-number drive-equipment-number*, where:

   - *volume-serial-number* is the required volume serial number.
   - *drive-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the drive.

   In the example, we assign the volume serial number **VOL600** to the new tape cartridge in drive **803**:

```
root@mds1:~# tplabel -new -vsn VOL600 803
root@mds1:~#
```

3. To label a new tape that resides in an automated media library, use the command **tplabel -new** *volume-serial-number library-equipment-number:slot*, where:

   - *volume-serial-number* is the required volume serial number.
   - *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the drive.
   - *slot* is the location of the cartridge within the library.

   In the example, we assign the volume serial number **VOL601** to the new tape cartridge in slot **19** of library **800**:

```
root@mds1:~# tplabel -new -vsn VOL601 800:19
```

```
root@mds1:~#
```

4. To relabel a tape that is loaded into a drive, use the command **tplabel -old** *old-volume-serial-number* **-new** *new-volume-serial-number drive-equipment-number*, where:

   - *volume-serial-number* is the required volume serial number.
   - *drive-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the drive.

   In the example, we reinitialize the tape cartridge in drive **804**, replacing the old volume serial number **AZ0001** with the new volume serial number **VOL120**:

   ```
   root@mds1:~# tplabel -old AZ0001 -vsn VOL120 804
   root@mds1:~#
   ```

5. To relabel a tape that resides in a tape library, use the command **tplabel -old** *old-volume-serial-number* **-new** *new-volume-serial-number library-equipment-number:slot*, where:

   - *volume-serial-number* is the required volume serial number.
   - *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the drive.
   - *slot* is the location of the cartridge within the library.

   You can reuse the existing volume serial number, if required. In the example, we reinitialize the tape cartridge in slot **23** of library **800** by relabeling it with its existing volume serial number **VOL121**:

   ```
   root@mds1:~# tplabel -old VOL601 -vsn VOL601 800:23
   root@mds1:~#
   ```

6. Stop here.

Label a New Optical Disk or Relabel an Existing Optical Disk

1. Log in to the file system host as **root**.

   In the example, the host is named **mds1**:

   ```
   root@mds1:~#
   ```

2. To label a new optical cartridge that is loaded into a drive, use the command **odlabel -new** *volume-serial-number drive-equipment-number***[:***side***]**, where:

   - *volume-serial-number* is the required volume serial number.
   - *drive-equipment-number* is the equipment ordinal number that the */etc/opt/SUNWsamfs/mcf* file assigns to the drive.
   - *side* (optional) is the specified side of a two-sided disk.

   In the example, we assign the volume serial number **OD1700** to the new, single-sided optical cartridge in drive **701**:

   ```
   root@mds1:~# odlabel -new -vsn OD1700 701
   root@mds1:~#
   ```

3. To label a new optical cartridge that resides in an automated media library, use the command **odlabel -new** *volume-serial-number library-equipment-number***:***slot***[:***side***]**, where:

- *volume-serial-number* is the required volume serial number.
- *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the drive.
- *slot* is the location of the cartridge within the library, and *side* (optional) is the specified side of a two-sided disk.

In the example, we assign the volume serial number **OD1701** to side **2** of the new, two-sided optical cartridge in slot **42** of library **700**:

```
root@mds1:~# odlabel -new -vsn OD1701 700:42:2
root@mds1:~#
```

4. To relabel an optical cartridge that is loaded into a drive, use the command **odlabel -old** *old-volume-serial-number* **-new** *new-volume-serial-number drive-equipment-number***[:***side***]**, where:

   - *volume-serial-number* is the required volume serial number.
   - *drive-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the drive.
   - *side* (optional) is the specified side of a two-sided disk.

   In the example, we reinitialize the optical cartridge in drive **702**, replacing the old volume serial number **OD1120** with the new volume serial number **OD1120** to:

   ```
   root@mds1:~# odlabel -old OD0001 -vsn OD1120 702
   root@mds1:~#
   ```

5. To relabel an existing optical cartridge that resides in an automated media library, use the command **odlabel -old** *volume-serial-number library-equipment-number***:***slot***[:***side***]**, where:

   - *volume-serial-number* is the required volume serial number.
   - *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the library.
   - *side* (optional) is the specified side of a two-sided disk.

   You can reuse the existing volume serial number if required. In the example, we reinitialize the optical cartridge in slot **23** of library **700** by relabeling it with its existing volume serial number, **OD1121**:

   ```
   root@mds1:~# odlabel -old OD1121 -vsn OD1121 800:23
   root@mds1:~#
   ```

6. Stop here.

Maintaining Data Integrity

The Oracle Hierarchical Storage Manager software provides both on-demand and automated tools for maintaining the integrity of data files stored on removable, tape media. This section addresses the following topics:

- displaying Data Integrity Validation (DIV) settings and status
- checking the integrity of a given tape volume
- monitoring automated integrity verification.

Displaying Data Integrity Validation (DIV) Settings and Status

This section covers the following tasks:

- displaying the DIV setting
- monitoring the Verify After Write status of archive files
- monitoring the Verify After Write status of devices.

Display the DIV Setting

To display the Data Integrity Validation (DIV) setting, use the command **samcmd L** and pipe the output to the Solaris **grep** command and the regular expression **div**.

In the example, DIV is **OFF**:

```
root@mds1:~# samcmd L | grep div
div             OFF
root@mds1:~#
```

Monitor the Verify After Write Status of Archive Files

To monitor the verification status of archive files during archiving, use the **samu** interface. Use the command **samu -d a**.

```
root@mds1:~# samu -d a
Archiver status                          samu 5.4         22:22:31 Mar 4 2014
sam-archiverd: Archiving files
sam-arfind:  hsmfs1 mounted at /hsm/hsmfs1
Files waiting to start     12,576  schedule     26,695  archiving     13,120
...
sam-arcopy: qfs.arset1.2.344 ti.TKC960
Verifying archive file at position 1175
```

Monitor the Verify After Write Status of Devices

To monitor the verification status of devices during archiving, use the **samu** interface. Use the command **samu -d s**:

```
root@mds1:~# samu -d s
Device status                          samu 5.4           22:27:53 Mar 4 2014
ty     eq state  device_name                      fs  status
sn    800 on     /dev/scsi/changer/c1t2d0        800 n--------r
ti    801 on     /dev/rmt/0cbn                   800 ---------p
...
hy    805 on     historian                       805 ----------
ti     91 on     /dev/rmt/4cbn                    90 -l----oVr
Verify averaging 240.9 MB/s
```

Checking the Integrity of a Given Tape Volume

When you need to verify the data integrity of particular tape volumes, use the Oracle HSM **tpverify** command. The **tpverify** command supports Oracle T10000C/D, LTO, and other commonly used media. T10000C/D media are verified using Oracle Data Integrity Validation. Other formats are checked using the widely supported SCSI **verify(6)** command.

The following sections outline some of the ways in which **tpverify** can be used. See the **tpverify** man page for full details:

- verifying the data on a tape that you have specified by library location
- verifying the data on a tape that you have specified by media type and VSN

- verifying the data on a tape using a specified drive
- restarting data verification from the start of the tape
- verifying ECC for all blocks on a T10000C/D tape
- verifying ECC and DIV checksums for all blocks on a T10000C/D tape
- rebuilding the Media Information Region (MIR) of a T10000C/D tape
- cancelling data verification for a specified tape
- displaying the DIV status and verification progress for a tape.

Verify the Data on a Tape Specified by Library Location

Use the command **tpverify** *library-equipment-number*:*slot*, where *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the automated tape library and *slot* is the slot address where the target volume resides within the library.

The **tpverify** command locates the last tape position that was verified by checking the library media catalog. It then loads the tape into the first available drive and starts validating from the point where it last stopped, using the default method—the **tpverify** Standard method for T10000C/D media or SCSI **verify(6)** for other media. The Standard method is optimized for speed and verifies the edges, beginning, end, and first 1,000 blocks of Oracle HSM media.

In the example, we validate the T10000D tape stored in slot **9** on library **800** using the Standard method:

```
root@mds1:~# tpverify 800:9
```

Verify the Data on a Tape Specified by Media Type and Volume Serial Number

Use the command **tpverify** *mediatype*.*volume-serial-number*, where *mediatype* is one of the two-character media type codes listed in Appendix A and *volume-serial-number* is the alphanumeric string that uniquely identifies the volume within the library.

The **tpverify** command locates the last tape position that was verified by checking the library media catalog. It then loads the tape into the first available drive and starts validating from the point where it last stopped, using the default method—the **tpverify** Standard method for T10000C/D media or SCSI **verify(6)** for other media.

In the example, we validate LTO (**li**) volume **VOL006** using the SCSI **verify(6)** command:

```
root@mds1:~# tpverify li.VOL006
```

Verify the Data on a Tape Using a Specified Drive

Use the command **tpverify** *library-equipment-number*:*slot* *device-equipment-number*, where:

- *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the automated tape library.
- *slot* is the slot address where the target volume resides within the library.
- *device-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the drive.

In the example, we validate the T10000D tape stored in slot **17** on library **800** using drive **803**:

```
root@mds1:~# tpverify 800:17 803
```

Restart Data Verification from the Start of the Tape

Use the command **tpverify -a** *library-equipment-number*:*slot* or **tpverify -a** *mediatype*.*volume-serial-number*, where:

- *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the automated tape library.
- *slot* is the slot address where the target volume resides within the library.
- *mediatype* is one of the two-character media type codes listed in Appendix A.
- *volume-serial-number* is the alphanumeric string that uniquely identifies the volume within the library.

The **-a** option overrides the default behavior and starts verifying from the beginning of the media, ignoring the position recorded in the media catalog.

In the example, we validate LTO (**li**) volume **VOL016** from the beginning of the tape:

```
root@mds1:~# tpverify -a li.VOL016
```

Verify ECC for All Blocks on a T10000C/D Tape

Use the command **tpverify -C** *library-equipment-number*:*slot* or **tpverify -C** *mediatype*.*volume-serial-number* where:

- *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the automated tape library.
- *slot* is the slot address where the target volume resides within the library.
- *mediatype* is one of the two-character media type codes listed in Appendix A.
- *volume-serial-number* is the alphanumeric string that uniquely identifies the volume within the library.

The **tpverify** command locates the last tape position that was verified by checking the library media catalog. It then starts validating from the point where it last stopped, using the Complete method specified by the **-C** option. The Complete method is more thorough than the standard method but can also be significantly slower. It checks Error Correction Codes (ECC) on all blocks on the media.

In the example, we validate T10000D (**ti**) volume **VOL516** using the Complete method:

```
root@mds1:~# tpverify -C ti.VOL516
```

Verify ECC and DIV Checksums for All Blocks on a T10000C/D Tape

Use the command **tpverify -P** *library-equipment-number*:*slot* or **tpverify -P** *mediatype*.*volume-serial-number*, where:

- *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the automated tape library.
- *slot* is the slot address where the target volume resides within the library.
- *mediatype* is one of the two-character media type codes listed in Appendix A.
- *volume-serial-number* is the alphanumeric string that uniquely identifies the volume within the library.

The **tpverify** command locates the last tape position that was verified by checking the library media catalog. It then starts validating from the point where it last stopped,

using the Complete Plus method specified by the **-P** option. The Complete Plus method is very thorough but also even slower than the other methods. It checks Error Correction Codes (ECC) and Data Integrity Validation checksums on all blocks on the media.

In the example, we validate T10000D (**ti**) volume **VOL521** using the Complete Plus method:

```
root@mds1:~# tpverify -P ti.VOL521
```

Rebuild the Media Information Region (MIR) of a T10000C/D Tape

Use the command **tpverify -M** *library-equipment-number*:*slot* or **tpverify -M** *mediatype*.*volume-serial-number*, where:

- *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the automated tape library.
- *slot* is the slot address where the target volume resides within the library.
- *mediatype* is one of the two-character media type codes listed in Appendix A.
- *volume-serial-number* is the alphanumeric string that uniquely identifies the volume within the library.

The **tpverify** command rebuilds a missing or damaged media information region (MIR) on an Oracle StorageTek tape cartridge, even if the tape is marked bad in the media catalog. Rebuilding is automatically specified when MIR damage is detected.

In the example, we validate T10000D (**ti**) volume **VOL523** using the MIR Rebuild method:

```
root@mds1:~# tpverify -M ti.VOL523
```

Cancel Data Verification for a Specified Tape

Use the command **tpverify -c** *library-equipment-number*:*slot* or **tpverify -c** *mediatype*.*volume-serial-number*, where:

- *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the automated tape library.
- *slot* is the slot address where the target volume resides within the library.
- *mediatype* is one of the two-character media type codes listed in Appendix A.
- *volume-serial-number* is the alphanumeric string that uniquely identifies the volume within the library.

The **tpverify -c** command cancels the current verification operation and records the last verified position on the tape in the media catalog. So you can stop a verification job to free a drive or volume for archiving or staging and then resume verification at the same point later.

In the example, we cancel verification of T10000D (**ti**) volume **VOL533**:

```
root@mds1:~# tpverify -c ti.VOL523
```

Display the DIV Status and Verification Progress for a Tape

Use the command **itemize -2** *library-equipment-number*, where *library-equipment-number* is the equipment ordinal number that the **/etc/opt/SUNWsamfs/mcf** file assigns to the automated tape library.

The **itemize -2** command catalogs the media in the specified library and lists the DIV status and verification progress for each volume.

In the example, we display verification status for volumes in the library with equipment ordinal number **800**. The **lvtime** (time last verified) fields display the time when **tpverify** last completed a full verification of the tape. A **status** field value of **div** indicates that the tape is DIV-capable while a value of **none** indicates that it is not. The **lvpos** (last verified position) fields show where **tpverify** was last canceled and where it will start when run again.

```
root@mds1:~# itemize -2 800
Robot VSN catalog: eq: 800       count: 60
slot    access_time  count  use  ty vsn
        lvtime      status          lvpos
   0    Apr  2 16:34     6   0%  ti VOL519
        Apr  2 09:23   div           0
   1    Apr  2 16:17    28  29%  ti VOL510
        Apr  2 16:17   div           0x9bb9
   2    none             0   0%  ti VOL511
        none          none           0
  ...
root@mds1:~#
```

Monitoring Automated Integrity Verification

Periodic Media Validation is the automated form of the **tpverify** command. This section provides instructions for maintenance tasks that may occasionally be necessary. These tasks include:

- viewing and validating the **verifyd.cmd** configuration file
- reloading the **verifyd.cmd** configuration file
- displaying all tape defects
- displaying defects for a particular tape volume
- clearing entries from the Periodic Media Verification tape defects database.

For instructions on configuring Periodic Media Verification, see the *Oracle Hierarchical Storage Manager and StorageTek QFS Installation and Configuration Guide* in the *Customer Documentation Library* (**http://docs.oracle.com/en/storage/#sw**).

View and Validate the verifyd.cmd Configuration File

To view the **verifyd.cmd** file at any time or to validate the file following editing, use the command **tpverify -x**.

The **tpverify -x** command checks the **/etc/opt/SUNWsamfs/verifyd.cmd** file and either calls out errors or displays the contents of the file.

```
root@mds1:~# tpverify -x
Reading '/etc/opt/SUNWsamfs/verifyd.cmd'.
PMV: off
     Run-time:
     Start Time: 2200
End Time: 0500
PMV Scan: all
PMV Method: Standard
STA Scan: off
Action: none
PMV VSNs: all
PMV Policy:
```

```
     Last Verified Age: 6m
root@mds1:~#
```

Reload the `verifyd.cmd` Configuration File

To reload the **`verifyd.cmd`** file without stopping the verification process, use the command **`tpverify -r`**.

```
root@mds1:~# tpverify -r
root@mds1:~#
```

Display All Defects Listed in the Periodic Media Verification Tape Defects Database

To list all defects that have been identified by Periodic Media Verification and stored in the tape defects database, use the command **`tpverify -l`**.

In the example, there are no defects in the database:

```
root@mds1:~# tpverify -l
No defects found.
root@mds1:~#
```

Display Defects Listed for a Particular Volume

To list all defects that have been identified on a particular volume, use the command **`tpverify -l`** *`mediatype`***`.`***`volume-serial-number`*, where:

- *`mediatype`* (optional) is one of the two-character media type codes listed in Appendix A.
- *`volume-serial-number`* is the alphanumeric string that uniquely identifies the volume within the library.

In the example, there are no defects listed in the database for the LTO (**`ti`**) volume **`VOL514`**:

```
root@mds1:~# tpverify -l ti.VOL514
No defects found.
root@mds1:~#
```

Clear Defects Listed in the Periodic Media Verification Tape Defects Database

To delete all defects that have been identified by Periodic Media Verification from the tape defects database, use the command **`tpverify -d`**.

To delete all defects listed for a particular volume, use the command **`tpverify -d`** *`mediatype`***`.`***`volume-serial-number`* where:

- *`mediatype`* (optional) is one of the two-character media type codes listed in Appendix A.
- *`volume-serial-number`* is the alphanumeric string that uniquely identifies the volume within the library.

```
root@mds1:~# tpverify -d
root@mds1:~# tpverify -d ti.VOL514
root@mds1:~#
```

Managing Cloud Storage

To properly maintain cloud storage libraries, carry out the following tasks:

- Periodically change the account passwords for all associated cloud storage accounts.

- Maintain local encryption keystore files as necessary.

Update Storage Cloud Account Password Files

To maintain the security of cloud storage accounts and your access to data in the Oracle Storage Cloud, you must periodically change account passwords. When you do, you must also update the encrypted password files that give Oracle HSM software access to the cloud resources.

Whenever you are notified of the need to change the password for an account, proceed as follows:

1. Log in to the file system metadata server host as **root**.

   In the example, the host name is **mds1**:

   ```
   root@mds1:~#
   ```

2. Idle all archiving processes. Use the command **samcmd aridle**.

   This command will allow current archiving and staging to complete, but will not start any new jobs:

   ```
   root@mds1:~# samcmd aridle
   root@mds:~#
   ```

3. Idle all staging processes. Use the command **samcmd stidle**.

   This command will allow current archiving and staging to complete, but will not start any new jobs:

   ```
   root@mds1:~# samcmd stidle
   root@mds:~#
   ```

4. Wait for active archiving jobs to complete. Check on the status of the archiving processes using the command **samcmd a**.

   When archiving processes are **Waiting for :arrun**, the archiving process is idle:

   ```
   root@mds1:~# samcmd a
   Archiver status samcmd 5.4 10:20:34 May 20 2014
   samcmd on samfs-mds
   sam-archiverd: Waiting for :arrun
   sam-arfind: ...
   Waiting for :arrun
   ```

5. Wait for active staging jobs to complete. Check on the status of the staging processes using the command **samcmd u**.

   When staging processes are **Waiting for :strun**, the staging process is idle:

   ```
   root@mds1:~# samcmd u
   Staging queue samcmd 5.4 10:20:34 May 20 2014
   samcmd on solaris.demo.lan
   Staging queue by media type: all
   sam-stagerd: Waiting for :strun
   root@mds1:~#
   ```

6. Stop the Oracle HSM services. Use the command **samd stop**.

   ```
   root@mds1:~# samd stop
   root@mds1:~#
   ```

7. Open a browser window, and log in to the Oracle Cloud Profile page for the cloud storage account.
8. From the Oracle Cloud Profile page, change the account password. Follow the instructions under "Managing Your Password" in *Getting Started with Oracle Cloud*.
9. Close the browser.
10. On the metadata server, update the password file for the cloud storage account. Use the command **sam-cloudd -p path/filename**, where:
    - **path** is the absolute path to the directory where you store the password.
    - **filename** is the name of the file that holds the password.

    The command prompts you for a password.

    In the example, we update the file **/etc/opt/SUNWsamfs/ocld1auth**:

    ```
    root@mds1:~# sam-cloudd -p /etc/opt/SUNWsamfs/ocld1auth
    Password:
    ```

11. At the prompt, enter the new password for the cloud storage account. When you are warned that the specified file already exists, confirm that the file can be overwritten.

    The **sam-cloudd -p path/filename** command encrypts the password and stores the result in the specified file. In the example, the character string **NeWp^sSwRd** represents the new Oracle Storage Cloud account password:

    ```
    root@mds1:~# sam-cloudd -p /etc/opt/SUNWsamfs/ocld1auth
    Password: NeWp^sSwRd
    root@mds1:~#
    ```

12. Once you have updated the password file, restart the Oracle HSM services. Use the command **samd start**.

    ```
    root@mds1:~# samd start
    Configuring Oracle HSM
    Starting Oracle HSM sam-amld daemon
    root@mds1:~#
    ```

13. Stop here.

    The new password has been encrypted and stored.

Maintaining Local Encryption Keystore Files

If you manage cloud library encryption keys using local keystore files, you may need to carry out the following maintenance tasks from time to time:

- adding new keys to a keystore file
- configuring a cloud library to use a new encryption key (rotating keys)
- retiring encryption keys.

Add a Key to a Cloud Library Keystore File

If you need to add one or more keys to an existing keystore file, proceed as follows:

1. If you have not already done so, log in to the Oracle HSM metadata server as **root**.

    ```
    root@mds:~#
    ```

2. Before proceeding, back up the keystore file. Create one or more additional copies. To create a time-stamped backup file, use the command **cp keystore_file keystore_file.`date +%F.%T`**, where **keystore_file** is the fully qualified path and file name of the keystore file.

   ```
   root@mds1:~# cp /root/cl800.ksf /root/cl800.ksf.`date +%F.%T`
   root@mds1:~#
   ```

3. Decrypt the keystore file. Use the command **decrypt -a aes -i inputfile -o outputfile**, where both **inputfile** and **outputfile** are the fully qualified path and file name of the keystore file. When prompted, enter the password for the keystore file.

   In the example, the keystore file is **/root/cl800.ksf**, where **cl800** is the family set name of corresponding cloud library. The string **P^ssw0rd** represents the corresponding password:

   ```
   root@mds1:~# decrypt -a aes -i /root/cl800.ksf -o /root/cl800.ksf
   Enter passphrase: P^ssw0rd
   root@mds1:~#
   ```

4. Using a text editor, open the keystore file for the cloud library, and scroll to the bottom of the file.

   In the example, the keystore file contains two, three-line entries for two keys, labeled **Key1** and **Key2**:

   ```
   root@mds1:~# vi /root/cl800.ksf
   # keystore file for Oracle HSM cloud library cl800
   key-label = Key1
   key-value = 4e6e2666f053e84ce8f1b67308c77d2d884e2c182bdaf965040c590f41ba25e3
   key-hash = 1384cec4e2e81eb80bed983a484b57dcaeaccea0d98ef8d068f00fb29f522186
   key-label = Key2
   key-value = 240dd62a6af501fafdd693fd05b0ac5779e7f743ca09d116408c1b5ff53a1c07
   key-hash = 073ffe4c4184977939195a68c8ba1c1febb4a28abda0cce121a2b9cf50435297
   ```

5. Using the text editor, add a line to the keystore file of the form **key-label = string**, where **string** is the alias that will indirectly identify the new key.

   In the example, the new key label is **Key3**.

   ```
   root@mds1:~# vi /root/cl800.ksf
   # keystore file for Oracle HSM cloud library cl800
   key-label = Key1
   key-value = 4e6e2666f053e84ce8f1b67308c77d2d884e2c182bdaf965040c590f41ba25e3
   key-hash = 1384cec4e2e81eb80bed983a484b57dcaeaccea0d98ef8d068f00fb29f522186
   key-label = Key2
   key-value = 240dd62a6af501fafdd693fd05b0ac5779e7f743ca09d116408c1b5ff53a1c07
   key-hash = 073ffe4c4184977939195a68c8ba1c1febb4a28abda0cce121a2b9cf50435297
   key-label = Key3
   ```

6. In another terminal window, create the new Advanced Encryption Standard (AES) encryption key. Use the command **dd if=/dev/urandom bs=32 count=1 2>/dev/null | od -t x1 -An | tr -d '\n \t' ; echo**, where:

   - **dd** is the Solaris utility that copies an input source to an output destination.
   - **if=/dev/urandom** makes the Solaris pseudorandom number generator the input source for **dd**.
   - **bs=32** sets the input and output block size for **dd** to the maximum AES key size, 32 bytes.

- **count=1** tells dd to copy one 32-byte block.
- **2>/dev/null** redirects any errors that **dd** generates from **stderr** to **/dev/null**.
- | **od** pipes the output of the **dd** utility to **od**, the Solaris octal dump utility.
- **-t x1** specifies the type of output that **od** is to generate: a one byte, hexadecimal number.
- **-An** tells **od** to omit an input offset address from the output.
- | **tr** pipes the output of the **dd** utility to the Solaris character translation utility.
- **-d'\n \t'** tells **tr** to delete all tab and newline characters from the input.
- **echo** writes the result of the preceding command string to standard output.

In the example, the output is the new **AES key ee77524f...72eec75c**:

```
root@mds1:~# dd if=/dev/urandom bs=32 count=1 2>/dev/null | od -t x1 -An | tr
-d '\n \t' ; echo
ee77524fb9ace964c7a31d68bccbec2e4dbe7c63c3197af922bd4c7d72eec75c
root@mds1:~#
```

7. Using the text editor, add a line to the keystore file of the form **key-value = AES_key**, where **AES_key** is the new key that you just generated.

   In the example, the key-value is **ee77524f...72eec75c**:

```
root@mds1:~# vi /root/cl800.ksf
# keystore file for Oracle HSM cloud library cl800
key-label = Key1
key-value = 4e6e2666f053e84ce8f1b67308c77d2d884e2c182bdaf965040c590f41ba25e3
key-hash = 1384cec4e2e81eb80bed983a484b57dcaeaccea0d98ef8d068f00fb29f522186
key-label = Key2
key-value = 240dd62a6af501fafdd693fd05b0ac5779e7f743ca09d116408c1b5ff53a1c07
key-hash = 073ffe4c4184977939195a68c8ba1c1febb4a28abda0cce121a2b9cf50435297
key-label = Key3
key-value = ee77524fb9ace964c7a31d68bccbec2e4dbe7c63c3197af922bd4c7d72eec75c
```

8. In another terminal window, create a SHA-256 hash of the key label and the AES key value. Use the command **print -n "KeylabelKeyvalue" | digest -a sha256**, where **Keylabel** is the value of the key-label parameter and **Keyvalue** is the value of the key-value parameter.

   In the example, the SHA-256 hash of **Key3** and **ee77524f...72eec75c is b2c73aae...26834c20**:

```
root@mds1:~# print -n "Key3ee77524f...72eec75c" | digest -a sha256
b2c73aae8317d8b43c4e7ca3c13e6edc992555e90daa0c230444223f26834c20
root@mds1:~#
```

9. Using the text editor, add a line to the keystore file of the form **key-hash = hash**, where **hash** is the hash value that you just calculated.

   In the example, the **key-hash** is **b2c73aae...26834c20**:

```
root@mds1:~# vi /root/cl800.ksf
# keystore file for Oracle HSM cloud library cl800
...
key-hash = 073ffe4c4184977939195a68c8ba1c1febb4a28abda0cce121a2b9cf50435297
key-label = Key3
key-value = ee77524fb9ace964c7a31d68bccbec2e4dbe7c63c3197af922bd4c7d72eec75c
key-hash = b2c73aae8317d8b43c4e7ca3c13e6edc992555e90daa0c230444223f26834c20
```

10. Save the keystore file.

```
root@mds1:~# vi /root/cl800.ksf
# keystore file for Oracle HSM cloud library cl800
...
key-hash = 073ffe4c4184977939195a68c8ba1c1febb4a28abda0cce121a2b9cf50435297
key-label = Key3
key-value = ee77524fb9ace964c7a31d68bccbec2e4dbe7c63c3197af922bd4c7d72eec75c
key-hash = b2c73aae8317d8b43c4e7ca3c13e6edc992555e90daa0c230444223f26834c20
:w
```

11. For each additional key that you intend to use when encrypting volumes in this cloud library, repeat steps 5 through 10.

12. When you have created keystore records for all required keys, close the editor.

    In the example, the finished keystore file holds entries for three AES keys:

```
root@mds1:~# vi /root/cl800.ksf
# keystore file for Oracle HSM cloud library cl800
key-label = Key1
key-value = 4e6e2666f053e84ce8f1b67308c77d2d884e2c182bdaf965040c590f41ba25e3
key-hash = 1384cec4e2e81eb80bed983a484b57dcaeaccea0d98ef8d068f00fb29f522186
key-label = Key2
key-value = 240dd62a6af501fafdd693fd05b0ac5779e7f743ca09d116408c1b5ff53a1c07
key-hash = 073ffe4c4184977939195a68c8ba1c1febb4a28abda0cce121a2b9cf50435297
key-label = Key3
key-value = ee77524fb9ace964c7a31d68bccbec2e4dbe7c63c3197af922bd4c7d72eec75c
key-hash = b2c73aae8317d8b43c4e7ca3c13e6edc992555e90daa0c230444223f26834c20
:q
root@mds1:~#
```

13. Encrypt the keystore file. Use the command **`encrypt -a aes -i inputfile -o outputfile`**, and enter the password for this keystore file when prompted for a **`Passphrase`**.

    The command parameters have the following functions:

    - **`-a aes`** specifies Advanced Encryption Standard.
    - **`-i inputfile`** specifies the absolute path and file name of the keystore file.
    - **`-o outputfile`** also specifies the absolute path and file name of the keystore file.

    In the example, the string **`P^ssw0rd`** represents the password:

```
root@mds1:~# encrypt -a aes -i /root/cl800.ksf -o /root/cl800.ksf
Enter passphrase: P^ssw0rd
Re-enter passphrase: P^ssw0rd
root@mds1:~#
```

14. Make sure that the keystore file permissions allow the owner read and write access and deny others. Use the command **`chmod 0600 keystore_file`**, where **`keystore_file`** is the fully qualified path and file name of the keystore file.

```
root@mds1:~# chmod 0600 /root/cl800.ksf
root@mds1:~#
```

15. Stop here.

    If required, you can now configure the cloud library to use a newly added key for encryption and decryption.

Configure a Cloud Library to Use a Different Encryption Key

You can change the key that a cloud library is currently using when encrypting volumes, either to rotate through a set of encryption keys or for some other reason. To do so, proceed as follows:

1. If you need to rotate through a set of keys, make sure that the keystore file contains the required number of keys. Then settle on a rotation order.
2. If you need to add encryption keys, do so now.

   If you are adding keys to facilitate key rotation, you may want to choose key labels that reflect the desired rotation order. For example: **Key1**, **Key2**, **Key3**, etc.
3. If you have not already done so, log in to the Oracle HSM metadata server as **root**.

   ```
   root@mds:~#
   ```
4. Using a text editor, open the **file /etc/opt/SUNWsamfs/param-file-name**, where **param-file-name** is the family set name for the cloud library and, thus, the name of the corresponding parameters file. Scroll down to the **keylabel_name** parameter.

   In the example, the cloud library's family set name is cl800, so we open the parameters file **/etc/opt/SUNWsamfs/cl800** and scroll down to the line **keylabel_name = Key1**:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
   ...
   keystore_type = file
   keystore_name = /root/cl800.ksf
   keystore_password_file = /root/cl800.ksf.pwd
   keylabel_type = static
   keylabel_name = Key1
   ```
5. In the parameters file, edit the line **keylabel_name = label_of_current_key** so that it reads **keylabel_name = label_of_next_key**, where:

   - **label_of_current_key** is the label of the key currently in use.
   - **label_of_next_key** is the label of the next key in your chosen rotation scheme.

   The keystore file named in the **keystore_name** parameter is **/root/cl800.ksf**. This keystore file contains three keys with three key labels: **Key1** (the label of the key that is currently in use), **Key2**, and **Key3**. So, in the parameters file, we change the line **keylabel_name = Key1** to read **keylabel_name = Key2**, the next key in our chosen rotation order:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
   ...
   keylabel_type = static
   keylabel_name = Key2
   ```
6. Save the parameters file, and close the editor.

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
   ...
   keylabel_type = static
   keylabel_name = Key2
   :wq
   root@mds1:~#
   ```
7. Stop here. The cloud library will use the specified key label to encrypt or decrypt files in the future.

Retire an Encryption Key After Unarchiving Files

When all archive copies that have been encrypted with a given key have been unarchived, you may retire the key. But you must be careful! Once you retire a cloud library encryption key, you cannot decrypt any encrypted files that remain in the cloud library.

1. Carefully identify the archive copies that you need to remove from the archive.
2. If you have not already done so, log in to the Oracle HSM metadata server as **root**.

   ```
   root@mds:~#
   ```

3. Delete all archived file copies that have been encrypted with the key and key label that you intend to retire. Use the **unarchive** command with the options and parameters that will delete all required files.
4. Before proceeding further, back up the keystore file. Create one or more additional copies. To create a time-stamped backup file, use the command **cp keystore_file keystore_file.`date +%F.%T`**, where **keystore_file** is the fully qualified path and file name of the keystore file.

   ```
   root@mds1:~# cp /root/cl800.ksf /root/cl800.ksf.`date +%F.%T`
   root@mds1:~#
   ```

5. Decrypt the keystore file. Use the command **decrypt -a aes -i inputfile -o outputfile**, where both **inputfile** and **outputfile** are the fully qualified path and file name of the keystore file. When prompted, enter the password for the keystore file.

   In the example, the keystore file **is /root/cl800.ksf**, where **cl800** is the family set name of corresponding cloud library. The string **P^ssw0rd** represents the corresponding password:

   ```
   root@mds1:~# decrypt -a aes -i /root/cl800.ksf -o /root/cl800.ksf
   Enter passphrase: P^ssw0rd
   root@mds1:~#
   ```

6. Using a text editor, open the keystore file for the cloud library, and scroll to the entry for the key that you want to retire.

   In the example, the keystore file contains three, three-line entries for three keys, labeled **Key1**, **Key2**, and **Key3**. We need to retire **Key1**:

   ```
   root@mds1:~# vi /root/cl800.ksf
   # keystore file for Oracle HSM cloud library cl800
   key-label = Key1
   key-value = 4e6e2666f053e84ce8f1b67308c77d2d884e2c182bdaf965040c590f41ba25e3
   key-hash = 1384cec4e2e81eb80bed983a484b57dcaeaccea0d98ef8d068f00fb29f522186
   key-label = Key2
   key-value = 240dd62a6af501fafdd693fd05b0ac5779e7f743ca09d116408c1b5ff53a1c07
   key-hash = 073ffe4c4184977939195a68c8ba1c1febb4a28abda0cce121a2b9cf50435297
   key-label = Key3
   key-value = ee77524fb9ace964c7a31d68bccbec2e4dbe7c63c3197af922bd4c7d72eec75c
   key-hash = b2c73aae8317d8b43c4e7ca3c13e6edc992555e90daa0c230444223f26834c20
   ```

7. Comment out the entry for the retired key. Enter a number sign (#) at the beginning of the line **key-label = retired**, where **retired** is the key label that you want to retire, and at the beginnings of the immediately following **key-value** and **key-hash** lines.

You can delete the lines instead of commenting them out. But commenting the lines out lets you recover if you later need to restore archive file copies that were encrypted with the retired key.

In the example, we need to retire **Key1**. So we comment out the lines **key-label = Key1**, **key-value = 4e6e266...41ba25e3**, and **key-hash = 1384cec4...9f522186**:

```
root@mds1:~# vi /root/cl800.ksf
# keystore file for Oracle HSM cloud library cl800
#key-label = Key1
#key-value = 4e6e2666f053e84ce8f1b67308c77d2d884e2c182bdaf965040c590f41ba25e3
#key-hash = 1384cec4e2e81eb80bed983a484b57dcaeaccea0d98ef8d068f00fb29f522186
key-label = Key2
key-value = 240dd62a6af501fafdd693fd05b0ac5779e7f743ca09d116408c1b5ff53a1c07
key-hash = 073ffe4c4184977939195a68c8ba1c1febb4a28abda0cce121a2b9cf50435297
key-label = Key3
key-value = ee77524fb9ace964c7a31d68bccbec2e4dbe7c63c3197af922bd4c7d72eec75c
key-hash = b2c73aae8317d8b43c4e7ca3c13e6edc992555e90daa0c230444223f26834c20
```

8. Save the edited keystore file, and close the editor.

```
root@mds1:~# vi /root/cl800.ksf
# keystore file for Oracle HSM cloud library cl800
#key-label = Key1
#key-value = 4e6e2666f053e84ce8f1b67308c77d2d884e2c182bdaf965040c590f41ba25e3
#key-hash = 1384cec4e2e81eb80bed983a484b57dcaeaccea0d98ef8d068f00fb29f522186
key-label = Key2
key-value = 240dd62a6af501fafdd693fd05b0ac5779e7f743ca09d116408c1b5ff53a1c07
key-hash = 073ffe4c4184977939195a68c8ba1c1febb4a28abda0cce121a2b9cf50435297
key-label = Key3
key-value = ee77524fb9ace964c7a31d68bccbec2e4dbe7c63c3197af922bd4c7d72eec75c
key-hash = b2c73aae8317d8b43c4e7ca3c13e6edc992555e90daa0c230444223f26834c20
:wq
root@mds1:~#
```

9. Encrypt the keystore file. Use the command **encrypt -a aes -i inputfile -o outputfile**, and enter the password for this keystore file when prompted for a **Passphrase**.

   The command parameters have the following functions:

   - **-a aes** specifies Advanced Encryption Standard.
   - **-i inputfile** specifies the absolute path and file name of the keystore file.
   - **-o outputfile** also specifies the absolute path and file name of the keystore file.

   In the example, the string **P^ssw0rd** represents the password:

```
root@mds1:~# encrypt -a aes -i /root/cl800.ksf -o /root/cl800.ksf
Enter passphrase: P^ssw0rd
Re-enter passphrase: P^ssw0rd
root@mds1:~#
```

10. Make sure that the keystore file permissions allow the owner read and write access and deny others. Use the command **chmod 0600 keystore_file**, where **keystore_file** is the fully qualified path and file name of the keystore file.

```
root@mds1:~# chmod 0600 /root/cl800.ksf
root@mds1:~#
```

11. Stop here.

The key and its associated key label and key hash have been retired. The cloud library will not use this key to encrypt or decrypt files in the future.

6

Managing Archives for Digital Preservation

Thus far, this document has discussed managing Oracle Hierarchical Storage Manager and StorageTek QFS Software solutions as ordinary UNIX file systems, where users and applications are regularly creating, modifying, and deleting files. The focus has been on the disk cache, with the archive serving primarily as a highly integrated backup service. In this chapter, we refocus on the archive as a repository and management solution for long-term data preservation. The previously covered management principles and techniques remain relevant. But now the disk cache serves primarily as a means of ingesting files into an archive that does not allow deletion or modification following ingestion.

Exact requirements vary. A repository that retains business or medical records for a legally mandated period may need to discard records periodically. But an archive storing scientific data, historical or genealogical records, or digital music, films, or television programs may need to store content, in effect, forever. For this reason, Oracle HSM supports digital preservation is several ways:

- Message digests (checksums) let you detect damage, data corruption, and unauthorized modifications to files, so that you can correct hardware problems and replace unsound files with sound copies stored elsewhere in the archive.
- File fixity attributes work with message digests to insure that only the super user can alter files that have been fixed. Whenever Oracle HSM stages or archives a fixed file, it revalidates a checksum stored with the fixity attribute to prove that the file remains unchanged.
- Oracle HSM Write Once Read Many (WORM) file systems let you make files read-only and enforce retention for a specified period. These file systems can be configured so that the super user cannot alter files or file attributes, such as the fixity attribute discussed above.

The chapter starts with a brief review of the basic, Oracle HSM data-protection measures that form the foundation of any long-term storage solution. It then explains the tasks that specifically address data-preservation:

- using message digests (checksums)
- making files immutable
- using WORM file systems.

Configuring File Systems for Preservation

Every preservation solution starts with sound, highly redundant file systems. So review the implementation chapters of the companion *Oracle Hierarchical Storage Manager and StorageTek QFS Installation and Configuration Guide*, if you have not already

done so. Protect access to the archive by providing redundant servers, network connections, and storage devices. Protect file data by configuring at least two additional copies of each file, with each stored on independent media. Archiving one copy to disk or solid-state storage devices and two copies to tape media is ideal, in most situations. When possible, insure that tape blocks are correctly written and read by implementing the Oracle HSM Data Integrity Verification feature. Protect file-system metadata by regularly generating dump files and by regularly backing up the archiving logs.

Using Message Digests (Checksums)

Message digests (checksums) let preservationists test archived files for changes that might indicate gradual deterioration, hardware or operator error, or deliberate, unauthorized alterations to the content. A message digest is simply a mathematical summary of a file's contents that has been generated by a one-way cryptographic hash function. Cryptographic hash functions are extremely sensitive to changes in their input data. Even small changes in the input produce large changes in the output. So message digests are ideal for detecting file corruption and unauthorized alterations. Recomputing a file's digest and comparing the resulting value to a stored digest value shows whether the file has changed.

Oracle Hierarchical Storage Manager file systems can ingest, create, store, and validate message digests using any of the following cryptographic hash functions:

- SHA1, the 160-bit member of the Secure Hash Algorithm family of cryptographic functions

  The Secure Hash Algorithms are defined in *Federal Information Processing Standard (FIPS) Publication 180-4*, National Institute of Standards and Technology (2012). Oracle HSM uses SHA1 by default.

- SHA256, the 256-bit member of the Secure Hash Algorithm family
- SHA384, the 384-bit member of the Secure Hash Algorithm family
- SHA512, the 512-bit member of the Secure Hash Algorithm family.
- MD5, the 128-bit Message Digest function defined by the Internet Engineering Task Force (IETF) in Request for Comment (RFC) 1321
- A proprietary, 128-bit Oracle HSM function that is now mainly useful for backward compatibility with older, Storage Archive Manager implementations.

Users can supply an existing digest value when a file in ingested into the repository or they can have the file system compute one, either immediately or when the file is first archived. Oracle HSM file systems store digest values with the file system metadata, using a special file attribute. Once the attribute is set, the file system recomputes a digest and validates it against the stored value whenever the corresponding file is rearchived and, optionally, whenever the file is staged from archival media to the disk cache.

Note, however, that the Oracle HSM media migration feature copies files to new media without recalculating checksums (for information on media migration see Chapter 8, "Migrating to New Storage Media"). If a file is not copied correctly, there is thus a small risk that the corruption will not be detected until the file is restaged and validated. Using Data Integrity Validation (DIV) minimizes this risk (see the *Oracle Hierarchical Storage Manager and StorageTek QFS Installation and Configuration Guide* for details).

Before you start using message digests, you should first make sure that the host can handle the required calculations without undue reductions in host performance. You can then carry out the following tasks as needed:

- supplying a pre-existing digest and enabling validation when a file is ingested
- generating a digest and enabling validation when a file is ingested
- generating a digest and enabling validation for each file in a directory
- validating the digest of a file during staging
- changing message digesting and validation attributes before a file is archived.

Make Sure that File-System Host Performance Will Be Adequate

If you plan to make significant use of message digests, make sure that the file-system host has enough computing resources for adequate performance. Most modern platforms incorporate dedicated cryptographic hardware that can efficiently perform specialized calculations without consuming central processor cycles. Be sure to take advantage of these capabilities when available.

To check the capabilities of a potential file-system host, proceed as follows:

1. Log in to the file system host as **root**:

   ```
   root@mds1:~#
   ```

2. Make sure that the host operating system is Solaris 11.1 or higher. Use the command **uname -v**.

   Earlier versions of the operating system do not support hardware acceleration of hash functions. In the example, the host operating system is Solaris 11.2:

   ```
   root@mds1:~# uname -v
   11.2
   root@mds1:~#
   ```

3. Display the instruction set architecture. At the command prompt, enter the command **isainfo -v**:

   ```
   root@mds1:~# isainfo -v
   ```

4. If the Solaris 11 host is an Oracle Sun SPARC T3 or later system, the output of the **isainfo -v** command should list instruction-sets that support **sha512**, **sha256**, **sha1**, and **md5** cryptographic algorithms.

   In the example, the SPARC host provides hardware acceleration for the SHA1, SHA2, and MD5 algorithm families:

   ```
   root@mds1:~# isainfo -v
   64-bit sparcv9 applications
           crc32c cbcond pause mont mpmul sha512 sha256 sha1 md5 camellia kasumi
           des aes ima hpc vis3 fmaf asi_blk_init vis2 vis popc
   root@mds1:~#
   ```

5. If the Solaris host is an x86/64 system, it will support SHA-1 hardware acceleration if the output of the **isainfo -v** command includes the **ssse3** (Supplemental Streaming SIMD Extensions 3) instruction set.

   In the example, the host x86/64 system includes the **ssse3** instruction set:

   ```
   root@mds1:~# isainfo -v
   64-bit amd64 applications
   ```

```
        avx xsave pclmulqdq aes sse4.2 sse4.1 ssse3 popcnt tscp ahf cx16 sse3
        sse2 sse fxsr mmx cmov amd_sysc cx8 tsc fpu
root@mds1:~#
```

Supply a Message Digest and Enable Validation for a File

When you are archiving files that already have associated message digests, proceed as follows.

1. Log in to the file system host as **root**:

   ```
   root@mds1:~#
   ```

2. At the command prompt, enter the command **ssum -a** *algorithm* **-h** *digest* **-G [-u]***filename*, where:

   - **-a** *algorithm* identifies the cryptographic hashing function that the file system should use when validating the file against the supplied message digest.
   - **-h** *digest* identifies the message digest that the file system should use to validate the file.
   - **-G** specifies immediate validation. The file system sets the **hash** file attribute to the value of the supplied message digest, independently calculates a message digest for the file and compares the result to the stored value. If the supplied and calculated digests match, the file system sets the **validated** attribute for the file. It then sets the **generate** attribute so that validity is rechecked whenever the file is rearchived.
   - **-u** sets the **use** file attribute (optional). Whenever the file is staged, the file system recalculates the digest and validates the result against the value stored in the **hash** attribute.
   - *filename* is the path and name of the file.

   In the example, we supply a SHA256 digest and ask the file system to immediately recalculate and validate the digest value for the file **data10** against the supplied value. When we check the file attributes with the command **sls -D -h data10**, we see that the **generate** and **validated** file attributes have been set, the **algorithm** attribute has been set to **SHA-256** and the digest value has been calculated and stored in the **hash** attribute

   ```
   root@mds1:~# ssum -h f03ce01b3828...f7459503007e -a sha256 -g data10
   root@mds1:~# sls -D -h data10
   data10:
     mode: -rw-r--r--    links:   1  owner: root        group: root
     length:     14975  admin id:      0  inode:    90217.1
     project: user.root(1)
     access:      Jul 16 16:14  modification: Jul 16 16:14
     changed:     Jul 16 16:15  attributes:   Jul 16 16:14
     creation:    Jul 16 16:14  residence:    Jul 16 16:14
     checksum: generate validated  algorithm: SHA-256
     hash: f03ce01b3828...f7459503007e
   root@mds1:~#
   ```

3. When necessary, edit the file as you normally would.

   In the example, we have modified a file named **data10m** since it was last archived. The **sls -D -h** command shows that the **S** (stale) flag has been set on both copies, since neither reflects the most recent changes. When we check the SHA-256 digest value for the modified file using the Solaris **digest** command, we see that the file's **hash** attribute also stores an out of date digest value:

```
root@mds1:~# sls -D -h data10m
data10m:
  mode: -rw-r--r--    links:   1  owner: root        group: root
  length:     14983  admin id:      0  inode:    90307.1
  project: user.root(1)
  copy 1: S----- Jul 17 16:47        dd.1    dk diskarchive f221
  copy 2: S----- Jul 20 11:31       a8d.1    li VOL002
  access:      Jul 20 11:32  modification: Jul 20 11:31
  changed:     Jul 17 16:37  attributes:   Jul 17 16:36
  creation:    Jul 17 16:36  residence:    Jul 17 16:36
  checksum: generate  algorithm: SHA-256
  hash: f03ce01b3828...f7459503007e
root@mds1:~# digest -a sha256 data10m
56c55bb421cc...71ac2ac0b7b0
root@mds1:~#
```

4. If necessary, you can change the digest attributes of a modified file prior to rearchiving.

   In the example, we change the digest algorithm from SHA256 to SHA1, with immediate effect:

```
root@mds1:~# ssum -a sha1 -G data10m
root@mds1:~# sls -D -h data10m
data10m:
  mode: -rw-r--r--    links:   1  owner: root        group: root
  length:     14983  admin id:      0  inode:    90307.1
  project: user.root(1)
  release -a;
  copy 1: S----- Jul 20 13:00        e0.1    dk diskarchive f224
  copy 2: S----- Jul 20 13:05       a93.1    li VOL002
  access:      Jul 20 16:39  modification: Jul 20 16:39
  changed:     Jul 17 16:37  attributes:   Jul 17 16:36
  creation:    Jul 17 16:36  residence:    Jul 20 16:29
  checksum: generate validated  algorithm: SHA-1
  hash: 92003525f0f8...53e29d0718c8
root@mds1:~#
```

5. Otherwise, wait for the file system to archive the modified file and automatically update the digest-related attributes.

   When a modified file is archived, the file system recalculates the digest value, stores the new value to the **hash** attribute, and sets the **S** (stale) flag on any archived copies of older versions of the file. In the example, we have edited the file **data10m** without altering the digest attributes. The archiver has created a new **copy 1** on disk, as scheduled, and updated the **hash** attribute. A copy of the unmodified file remains on tape, flagged **S** (stale), until it is time for the archiver to create **copy 2**:

```
root@mds1:~# sls -D -h data10m
data10m:
  mode: -rw-r--r--    links:   1  owner: root        group: root
  length:     14983  admin id:      0  inode:    90307.1
  project: user.root(1)
  copy 1: ------ Jul 17 16:47        dd.1    dk diskarchive f221
  copy 2: S----- Jul 20 11:31       a8d.1    li VOL002
  access:      Jul 20 11:32  modification: Jul 20 11:31
  changed:     Jul 17 16:37  attributes:   Jul 17 16:36
  creation:    Jul 17 16:36  residence:    Jul 17 16:36
  checksum: generate  algorithm: SHA-256
  hash: 56c55bb421cc...71ac2ac0b7b0
```

Generate a Message Digest and Enable Validation for a File

To generate a digest for a file and enable file validation, proceed as follows:

1. Log in to the file system host as **root**:

   ```
   root@mds1:~#
   ```

2. At the command prompt, enter the command **ssum -a** *algorithm* **-g|G [-u]** *filename*, where:

   - **-a** *algorithm* specifies the cryptographic hashing function that the file system will use when generating a message digest for the file.
   - **-g** sets the **generate** file attribute for the file. The first time that the file is archived, the file system calculates a message digest. Whenever the file is rearchived, the file system recalculates the digest and validates the result against the stored value.
   - **-G** sets the **generate** and **validate** file attributes for the file. The file system immediately calculates a message digest and stores the result in the **hash** attribute. Whenever the file is archived, the file system recalculates the digest and validates the result against the stored value.
   - **-u** sets the **use** file attribute (optional). Whenever the file is staged, the file system recalculates the digest and validates the result against the value stored in the **hash** attribute.
   - *filename* is the path and name of the file.

   In the example, we ask the file system to use the SHA256 algorithm to calculate the digest for the file **data11** prior to archiving. When we check the file attributes with the command **sls -D -h data10**, we see that, for each file, the **generate** file attribute has been set and the **algorithm** attribute has been set to **SHA-256**. The file has not yet been archived, so the digest value has not as yet been calculated and stored in the **hash** attribute:

   ```
   root@mds1:~# ssum -a sha256 -g data11
   root@mds1:~# sls -D -h data11
   data11:
     mode: -rw-r--r--    links:   1  owner: root        group: root
     length:     14975  admin id:      0  inode:    90218.1
     project: user.root(1)
     access:      Jul 16 16:14  modification: Jul 16 16:14
     changed:     Jul 16 16:22  attributes:   Jul 16 16:14
     creation:    Jul 16 16:14  residence:    Jul 16 16:14
     checksum: generate  algorithm: SHA-256
     hash:
   root@mds1:~#
   ```

3. When necessary, edit the file as you normally would.

   In the example, we have modified a file named **data11m** since it was last archived. The **sls -D -h** command shows that the **S** (stale) flag has been set on both copies, since neither reflects the most recent changes. When we check the SHA-256 digest value for the modified file using the Solaris **digest** command, we see that the file's **hash** attribute also stores an out of date digest value:

   ```
   root@mds1:~# sls -D -h data11m
   data11m:
     mode: -rw-r--r--    links:   1  owner: root        group: root
     length:     14983  admin id:      0  inode:    90307.1
     project: user.root(1)
   ```

```
  copy 1: S----- Jul 17 16:47        dd.1    dk diskarchive f221
  copy 2: S----- Jul 20 11:31       a8d.1    li VOL002
  access:      Jul 20 11:32  modification: Jul 20 11:31
  changed:     Jul 17 16:37  attributes:   Jul 17 16:36
  creation:    Jul 17 16:36  residence:    Jul 17 16:36
  checksum: generate  algorithm: SHA-256
  hash: f03ce01b3828...f7459503007e
root@mds1:~# digest -a sha256 data11m
56c55bb421cc...71ac2ac0b7b0
root@mds1:~#
```

4. If necessary, you can change the digest attributes of a modified file prior to rearchiving.

   In the example, we change the digest algorithm from SHA256 to SHA1, with immediate effect:

```
root@mds1:~# ssum -a sha1 -G data11m
root@mds1:~# sls -D -h data11m
data11m:
  mode: -rw-r--r--    links:   1  owner: root        group: root
  length:     14983  admin id:      0  inode:    90307.1
  project: user.root(1)
  release -a;
  copy 1: S----- Jul 20 13:00        e0.1    dk diskarchive f224
  copy 2: S----- Jul 20 13:05       a93.1    li VOL002
  access:      Jul 20 16:39  modification: Jul 20 16:39
  changed:     Jul 17 16:37  attributes:   Jul 17 16:36
  creation:    Jul 17 16:36  residence:    Jul 20 16:29
  checksum: generate validated  algorithm: SHA-1
  hash: 92003525f0f8...53e29d0718c8
root@mds1:~#
```

5. Otherwise, wait for the file system to archive the modified file and automatically update the digest-related attributes.

   When a modified file is archived, the file system recalculates the digest value, stores the new value to the **hash** attribute, and sets the **S** (stale) flag on any archived copies of older versions of the file.

   In the example, we have edited the file **data11m** without altering the digest attributes. The archiver has created a new **copy 1** on disk, as scheduled, and updated the **hash** attribute. A copy of the unmodified file remains on tape, flagged **S** (stale), until it is time for the archiver to create **copy 2**:

```
root@mds1:~# sls -D -h data11m
mdata11:
  mode: -rw-r--r--    links:   1  owner: root        group: root
  length:     14983  admin id:      0  inode:    90307.1
  project: user.root(1)
  copy 1: ------ Jul 17 16:47        dd.1    dk diskarchive f221
  copy 2: S----- Jul 20 11:31       a8d.1    li VOL002
  access:      Jul 20 11:32  modification: Jul 20 11:31
  changed:     Jul 17 16:37  attributes:   Jul 17 16:36
  creation:    Jul 17 16:36  residence:    Jul 17 16:36
  checksum: generate  algorithm: SHA-256
  hash: 56c55bb421cc...71ac2ac0b7b0
```

Generate a Message Digest and Enable Validation for Each File in a Directory

To recursively generate a digest and set the validation attributes for every file in a directory, proceed as follows:

1. Log in to the file system host as **root**:

   ```
   root@mds1:~#
   ```

2. At the command prompt, enter the command **ssum -a** *algorithm* **-g|G [-u] -r** *directoryname*, where:

   - **-a** *algorithm* specifies the cryptographic hashing function that the file system will use when generating message digests.
   - **-g** sets the **generate** file attribute for each file. The first time that a file is archived, the file system calculates a message digest for the file. Whenever the file is rearchived, the file system recalculates the digest and validates the result against the stored value.
   - **-G** sets the **generate** and **validate** file attributes for each file. The file system immediately calculates a message digest and stores the result in the **hash** attribute. Whenever the file is archived, the file system recalculates the digest and validates the result against the stored value.
   - **-u** sets the **use** file attribute (optional). Whenever the file is staged, the file system recalculates the digest and validates the result against the stored value.
   - **-r** recursively applies the command to all files in the specified directory.
   - *directoryname* is the path and name of the directory.

   In the first example, we tell the file system to use the SHA256 algorithm to calculate the digest for the files in the directory **datasetA** prior to archiving. When we check the file attributes with the command **sls -D -h datasetA**, we see that, for each file, the **generate** file attribute has been set and the **algorithm** attribute has been set to **SHA-256**. The files have not yet been archived, so the digest values have not as yet been calculated and stored in the **hash** attribute:

   ```
   root@mds1:~# ssum -a sha256 -g -r datasetA
   root@mds1:~# sls -D -h datasetA
   datasetA/pdata0:
     mode: -rw-r--r--    links:   1  owner: root        group: root
     length:     14983  admin id:      0  inode:    90232.1
     project: user.root(1)
     access:      Jul 16 16:47  modification: Jul 16 16:47
     changed:     Jul 16 16:47  attributes:   Jul 16 16:47
     creation:    Jul 16 16:47  residence:    Jul 16 16:47
     checksum: generate  algorithm: SHA-256
     hash:
   ...
   datasetA/pdata20:
     mode: -rw-r--r--    links:   1  owner: root        group: root
     length:     14983  admin id:      0  inode:    90234.1
     project: user.root(1)
     access:      Jul 16 16:47  modification: Jul 16 16:47
     changed:     Jul 16 16:47  attributes:   Jul 16 16:47
     creation:    Jul 16 16:47  residence:    Jul 16 16:47
     checksum: generate  algorithm: SHA-256
     hash:
   ...
   root@mds1:~#
   ```

In the second example, we ask the file system to use the SHA256 algorithm to immediately calculate the digest for the files in the directory **datasetB** prior to archiving. When we check the file attributes with the command **sls -D -h datasetB**, we see that, for each file, the **generate** and **validated** file attributes have been set, the **algorithm** attribute has been set to **SHA-256**, and the digest value has been calculated and stored in the **hash** attribute:

```
root@mds1:~# ssum -a sha256 -G -r datasetB
root@mds1:~# sls -D -h datasetB
datasetB/qdata0:
  mode: -rw-r--r--    links:   1  owner: root        group: root
  length:     14983  admin id:      0  inode:    90232.1
  project: user.root(1)
  access:      Jul 16 16:47  modification: Jul 16 16:47
  changed:     Jul 16 16:47  attributes:   Jul 16 16:47
  creation:    Jul 16 16:47  residence:    Jul 16 16:47
  checksum: generate validated  algorithm: SHA-256
  hash: 4d2800eb82b3...520341edde95
...
datasetB/qdata12:
  mode: -rw-r--r--    links:   1  owner: root        group: root
  length:     14983  admin id:      0  inode:    90234.1
  project: user.root(1)
  access:      Jul 16 16:47  modification: Jul 16 16:47
  changed:     Jul 16 16:47  attributes:   Jul 16 16:47
  creation:    Jul 16 16:47  residence:    Jul 16 16:47
  checksum: generate validated  algorithm: SHA-256
  hash: 5b057f1b7b48...88c590d47dec
...
root@mds1:~#
```

Validate the Message Digest of a File During Staging

When required, you can validate a file before it is staged to the disk cache for use. Proceed as follows:

1. Log in to the file system host as **root**:

   ```
   root@mds1:~#
   ```

2. At the command prompt, enter the command **ssum -u [-a** *algorithm* **[-h** *digest***] -g|G]** *filename*, where:

   - **-u** specifies validation prior to staging by setting the **use** file attribute. When the **use** attribute is set for a file, the file system will not copy the file from archival media to the disk cache until it has generated a message digest and successfully validated the result against the value stored in the file's **hash** attribute.
   - **-a** *algorithm*, **-h** *digest*, and **-g|G** are optional parameters that set the required **algorithm**, **hash**, and **generate** attributes on the file if the attributes have not been set previously.
   - *filename* is the path and name of the file.

   In the example, we have already enabled validation for the file **data102**. As the command **sls -D -h data102** shows, the **generate** and **validated** file attributes have been set, the **algorithm** attribute has been set to **SHA-256**, and the digest value has been calculated and stored in the **hash** attribute:

   ```
   root@mds1:~# ssum -a sha256 -F data102
   ```

```
root@mds1:~# sls -D -h data102
data102:
  mode: -rw-r--r--    links:   1  owner: root        group: root
  length:     14979  admin id:      0  inode:    90264.1
  project: user.root(1)
  access:      Jul 16 17:34  modification: Jul 16 17:34
  changed:     Jul 16 17:34  attributes:   Jul 16 17:34
  creation:    Jul 16 17:34  residence:    Jul 16 17:34
  checksum: generate validated  algorithm: SHA-256
  hash: baae932ce1cf...93166a2e36b5
root@mds1:~#
```

So we can set the **use** attribute to insure that the file system validates the file prior to staging. The command **sls -D -h data102** shows that the **use** attribute is now set:

```
root@mds1:~# ssum -u data102
root@mds1:~# sls -D -h data102
data102:
  mode: -rw-r--r--    links:   1  owner: root        group: root
  length:     14979  admin id:      0  inode:    90264.1
  project: user.root(1)
  access:      Jul 16 17:34  modification: Jul 16 17:34
  changed:     Jul 16 17:34  attributes:   Jul 16 17:34
  creation:    Jul 16 17:34  residence:    Jul 16 17:34
  checksum: generate use validated  algorithm: SHA-256
  hash: baae932ce1cf...93166a2e36b5
root@mds1:~#
```

Change Message Digesting and Validation Attributes Before a File is Archived

If a file that has not been made immutable and has not yet been archived, you can change message digesting and validation attributes using the procedure below.

1. Log in to the file system host as **root**:

   ```
   root@mds1:~#
   ```

2. If necessary, change the digesting algorithm. At the command prompt, enter the command **ssum -a** *newalgorithm filename*, where:

   - **-a** *newalgorithm* specifies the cryptographic hash function that replaces the previously specified digesting algorithm.
   - *filename* is the path and name of the file.

   In the example, our preservation policies require the highly collision-resistent SHA256 function. But as the command **sls -D -h** shows, we have inadvertently specified the SHA1 algorithm when we set the digest attributes of the file **data319**. Since the file has not yet been archived, we can successfully change the algorithm to SHA256:

   ```
   root@mds1:~# sls -D -h data319
   data319:
     mode: -rw-r--r--    links:   1  owner: root        group: root
     length:     14983  admin id:      0  inode:    90301.1
     project: user.root(1)
     access:      Jul 17 15:27  modification: Jul 17 15:27
     changed:     Jul 17 15:28  attributes:   Jul 17 15:27
     creation:    Jul 17 15:27  residence:    Jul 17 15:27
     checksum: generate  algorithm: SHA-1
     hash:
   ```

```
root@mds1:~# ssum -a sha256 data319
root@mds1:~# sls -D -h data319
data319:
  mode: -rw-r--r--    links:   1  owner: root        group: root
  length:     14983  admin id:      0  inode:    90301.1
  project: user.root(1)
  access:      Jul 17 15:27  modification: Jul 17 15:27
  changed:     Jul 17 15:28  attributes:   Jul 17 15:27
  creation:    Jul 17 15:27  residence:    Jul 17 15:27
  checksum: generate  algorithm: SHA-256
  hash:
root@mds1:~#
```

3. If necessary, clear the digest attributes and restore the default file settings. At the command prompt, enter the command **ssum -d** *filename*, where:

   - **-d** resets the file digest attributes to their default values.
   - *filename* is the path and name of the file.

   In the example, we did not mean to configure message digesting and validation for the file **data44**. But, as the command **sls -D -h** shows, we have inadvertently done so. Since the file has not yet been archived, we can successfully clear **generate** and **use**, the attributes that control digest validation during archiving and staging. The data in **validated**, **algorithm**, and **hash** attributes remains but does not affect file system's behavior:

```
root@mds1:~# sls -D -h data44
data44:
  mode: -rw-r--r--    links:   1  owner: root        group: root
  length:     14983  admin id:      0  inode:    90292.1
  project: user.root(1)
  access:      Jul 17 14:58  modification: Jul 17 14:57
  changed:     Jul 17 14:58  attributes:   Jul 17 14:57
  creation:    Jul 17 14:57  residence:    Jul 17 14:57
  checksum: generate use validated  algorithm: SHA-256
  hash: 3b4b15f8f69c...bae62c7e7568
root@mds1:~# ssum -d data44
root@mds1:~# sls -D -h data44
data44:
  mode: -rw-r--r--    links:   1  owner: root        group: root
  length:     14983  admin id:      0  inode:    90292.1
  project: user.root(1)
  access:      Jul 17 14:58  modification: Jul 17 14:57
  changed:     Jul 17 14:58  attributes:   Jul 17 14:57
  creation:    Jul 17 14:57  residence:    Jul 17 14:57
  checksum: validated  algorithm: SHA-256
  hash:  3b4b15f8f69c...bae62c7e7568
root@mds1:~#
```

4. If necessary, reset any required message digesting and validation attributes before the file is archived. At the command prompt, enter the command **ssum** with the appropriate options and file name.

   In the example, we decide to re-enable message digesting on the file **qndat44** and validate digests prior to archiving. But we do not require validation prior to staging. So we restore the **generate** attribute but not the **use** attribute:

```
root@mds1:~# ssum -g data44
root@mds1:~# sls -D -h data44
data44:
  mode: -rw-r--r--    links:   1  owner: root        group: root
```

```
  length:     14983  admin id:      0  inode:    90292.1
  project: user.root(1)
  access:      Jul 17 14:58  modification: Jul 17 14:57
  changed:     Jul 17 14:58  attributes:   Jul 17 14:57
  creation:    Jul 17 14:57  residence:    Jul 17 14:57
  checksum: generate validated  algorithm: SHA-256
  hash:  3b4b15f8f69c...bae62c7e7568
root@mds1:~#
```

Making Files Immutable

Preservation requirements frequently require mechanisms that assure *file fixity*. The archive must both prevent changes and prove that such changes have not occurred. To provide fixity, Oracle HSM archival file systems combine the message digests and digest-related file attributes discussed above with additional attributes that render the file immutable. Once a file has been made immutable, only those with super-user authority can change its status. If you combine immutability with a strict Write Once Read Many (WORM) file system, even super users will be unable to make changes (for details, see "Understanding WORM File Systems" on page 6-15).

You can make a file immutable in either of the following situations:

- when supplying a message digest
- when generating a message digest.

Supply a Message Digest and Make a File Immutable

When you need to insure that a file remains unchanged after ingestion into the archive, proceed as follows.

1. Log in to the file system host as **root**:

   ```
   root@mds1:~#
   ```

2. At the command prompt, enter the command **ssum -a** *algorithm* **[-h** *digest***] -F** *filename*, where:

   - **-a** *algorithm* identifies the cryptographic hashing function that the file system should use when validating the file against the supplied message digest.
   - **-h** *digest* identifies the message digest that the file system should use to validate the file.
   - **-F** specifies immediate validation and immutability, and sets the **fixity**, **generate**, **validated**, and **use** file attributes. The file system immediately calculates and validates a message digest. When the file is staged or archived, the file system recalculates and revalidates a message digest.
   - *filename* is the path and name of the file.

   In the example, we supply a SHA256 digest and tell the file system to recalculate the digest, validate the value for the file **data20**, and make the file immutable. When we check the file attributes with the command **sls -D -h data10**, we see that, for each file, the fixity, **generate**, **use**, and **validated** file attributes have been set, the **algorithm** attribute has been set to **SHA-256**, and the digest value has been calculated and stored in the **hash** attribute:

   ```
   root@mds1:~# ssum -h bfaefde932cf...d450892eda63 -a sha256 -F data20
   root@mds1:~# sls -D -h data20
   data20:
     mode: -rw-r--r--    links:   1  owner: root        group: root
   ```

```
  length:     14979  admin id:      0  inode:    90264.1
  project: user.root(1)
  access:      Jul 16 17:34  modification: Jul 16 17:34
  changed:     Jul 16 17:34  attributes:   Jul 16 17:34
  creation:    Jul 16 17:34  residence:    Jul 16 17:34
  checksum: fixity generate use validated  algorithm: SHA-256
  hash: bfaefde932cf...d450892eda63
root@mds1:~#
```

Generate a Message Digest and Make a File Immutable

When you are archiving files that already have associated message digests and need to insure that the file remains unchanged after ingestion into the archive, proceed as follows.

1. Log in to the file system host as **root**:

   ```
   root@mds1:~#
   ```

2. At the command prompt, enter the command **ssum -a** *algorithm* **[-h** *digest***] -F** *filename*, where:

   - **-a** *algorithm* identifies the cryptographic hashing function that was used to generate the digest that is specified in the **-h** *digest* parameter.
   - **-F** sets the **fixity**, **generate**, **validated**, and **use** file attributes. The file system immediately calculates and validates a message digest. When the file is staged or archived, the file system recalculates and revalidates a message digest.
   - *filename* is the path and name of the file.

   In the example, we tell the file system to calculate a SHA256 digest, validate the value for the file **data200**, and make the file immutable. When we check the file attributes with the command **sls -D -h data10**, we see that, for each file, the **fixity**, **generate**, **validated**, and **use** file attributes have been set, the **algorithm** attribute has been set to **SHA-256**, and the digest value has been calculated and stored in the **hash** attribute:

   ```
   root@mds1:~# ssum -a sha256 -F data200
   root@mds1:~# sls -D -h data200
   data200:
     mode: -rw-r--r--    links:   1  owner: root        group: root
     length:     14979  admin id:      0  inode:    90264.1
     project: user.root(1)
     access:      Jul 16 17:34  modification: Jul 16 17:34
     changed:     Jul 16 17:34  attributes:   Jul 16 17:34
     creation:    Jul 16 17:34  residence:    Jul 16 17:34
     checksum: fixity generate use validated  algorithm: SHA-256
     hash: efde93cc12cf...d496602e36dd
   root@mds1:~#
   ```

Checking File Digest and Fixity Attributes

To view the message digest and fixity attributes of one or more files, use the Oracle HSM directory listing command, **sls**. Proceed as follows.

List Message Digesting and Validation Attributes

1. Log in to the file system host as **root**:

```
root@mds1:~#
```

2. At the command prompt, enter the command **sls -D -h** *filename*, where:

   - **-D** specifies a detailed display of file attributes.
   - **-h** includes the hash (digest) value in the display.
   - *filename* identifies one or more files by path and name.

   In the example, we see the file digest attributes for the file **data02** listed in the **checksum** and **hash** fields of the display:

```
root@mds1:~# sls -D -h data02
data02:
  mode: -rw-r--r--    links:   1  owner: root        group: root
  length:     14975  admin id:      0  inode:    90217.1
  project: user.root(1)
  access:      Jul 16 16:14  modification: Jul 16 16:14
  changed:     Jul 16 16:15  attributes:   Jul 16 16:14
  creation:    Jul 16 16:14  residence:    Jul 16 16:14
  checksum: generate use validated  algorithm: SHA-256
  hash: f03ce01b3828...f7459503007e
root@mds1:~#
```

   - The **hash** attribute stores the message digest for the file, **f03ce01b3828...f7459503007e**.
   - The **algorithm** attribute shows that the **SHA-256** cryptographic hashing function generated the stored message digest.
   - The **generate** attribute shows that the file system independently recalculates the message digest and validates it against the stored value whenever the file is archived.
   - The **use** attribute shows that the file system independently recalculates the message digest and validates it against the stored value whenever the file is staged.
   - The **validated** attribute shows that the independently calculated message digest matched the value stored in the **hash** attribute when last checked.
   - The **fixity** attribute appears if the file has been made immutable.

Using WORM File Systems

When legal or archival considerations so require, you can create write-once read-many (WORM) directories and files in any Oracle HSM file system that has been configured to support them. This section focuses on understanding WORM file systems and on specific tasks that you need to perform when working with WORM files and directories, including:

- WORM-enabling directories
- activating WORM protection for a file
- finding and listing WORM files.

For information on enabling WORM support for a file system, see the *Oracle Hierarchical Storage Manager and StorageTek QFS Installation and Configuration Guide*.

Understanding WORM File Systems

Write Once Read Many (WORM) file systems protect data by letting users make files read-only for the duration of a specified retention period. WORM-enabled Oracle HSM file systems support default and customizable file-retention periods, data and path immutability, and subdirectory inheritance of the WORM setting.

Depending on how your file systems are configured, you use one of two Oracle HSM WORM modes:

- standard compliance mode (the default)
- emulation mode

In a file system that is mounted under the standard WORM mode, a user WORM-enables directories and starts the read-only retention period for files by executing the command **`chmod 4000`** *`path_name`*, where *`path_name`* is the path and name of the file or directory. This sets UNIX **`setuid`** (*set user ID upon execution*) permission. Setting **`setuid`** permission on a file that also has **`execute`** permission is a security risk, so, in standard WORM mode, only non-executable files can be made read-only.

In a file system that is mounted under the WORM emulation mode, a user WORM-enables directories and starts the read-only retention period for files by executing the command **`chmod 555`** *`path_name`*, where *`path_name`* is the path and name of a writable file or directory. Since emulation mode does not require **`setuid`** permission, executable files can be made read-only and assigned retention periods.

Both standard and emulation modes have a strict WORM implementation and a less restrictive, *lite* implementation. Both strict and lite implementations do not allow changes to data or paths once retention has been triggered on a file or directory. Both set the default retention period to 43,200 minutes (30 days). But the lite implementation relaxes some restrictions for **`root`** users.

The strict implementations do not let anyone shorten the specified retention period or delete files or directories prior to the end of the retention period. They also do not let anyone use **`sammkfs`** to delete volumes that hold currently retained files and directories. The strict implementations are thus well-suited to meeting the most demanding legal, regulatory compliance, and preservation requirements.

The lite implementations let **`root`** users shorten retention periods, delete files and directories, and delete volumes using the **`sammkfs`** command. This provides a high level of protection against casual data loss and provides more flexibility when administering file systems and storage resources. But file systems that allow super users this degree of control may not meet some legal and regulatory compliance requirements.

You can create both hard and soft links to WORM files. You can only create hard links with files that reside in a WORM-capable directory. After a hard link is created, it has the same WORM characteristics as the original file. Soft links can also be established, but a soft link cannot use the WORM features. Soft links to WORM files can be created in any directory in an Oracle HSM file system.

For full information on creating and configuring WORM file systems, see the *Oracle Hierarchical Storage Manager and StorageTek QFS Installation and Configuration Guide* in the *Customer Documentation Library*.

WORM-Enable a Directory

When you WORM-enable a directory, you add support for WORM files, but do not otherwise change the characteristics of the directory. Users can continue to create and

edit files within a WORM-enabled directory, and WORM-enabled directories that do not contain WORM files can be deleted. To WORM-enable a directory, proceed as follows:

1. Log in to the file-system server.

   ```
   user@mds1:~#
   ```

2. See if the directory has already been WORM-enabled. Use the command **sls -Dd** *directory*, where *directory* is the path and name of the directory. Look for the attribute **worm-capable** in the output of the command.

   Usually, directories will be WORM-enabled, because, when one user WORM-enables a directory, all current and future child directories inherit the WORM capability (for full information on the command, see the **sls** man page). In the first example, we find that our target directory, **/hsm/hsmfs1/records**, is already worm-enabled:

   ```
   user@mds1:~# sls -Dd /hsm/hsmfs1/records/2013/
   /hsm/hsmfs1/records/2013:
     mode: drwxr-xr-x    links:   2  owner: root        group: root
     length:      4096  admin id:      0  inode:     1048.1
     project: user.root(1)
     access:      Mar  3 12:15  modification: Mar  3 12:15
     changed:     Mar  3 12:15  attributes:   Mar  3 12:15
     creation:    Mar  3 12:15  residence:    Mar  3 12:15
     worm-capable        retention-period: 0y, 30d, 0h, 0m
   ```

   But in the second example, we find that our target directory, **/hsm/hsmfs1/documents**, is *not* worm-enabled:

   ```
   user@mds1:~# sls -Dd /hsm/hsmfs1/documents
   /hsm/hsmfs1/documents
     mode: drwxr-xr-x    links:   2  owner: root        group: root
     length:      4096  admin id:      0  inode:     1049.1
     project: user.root(1)
     access:      Mar  3 12:28  modification: Mar  3 12:28
     changed:     Mar  3 12:28  attributes:   Mar  3 12:28
     creation:    Mar  3 12:28  residence:    Mar  3 12:28
   ```

3. If the directory is not WORM-enabled and if the file system was mounted with the **worm_capable** or **worm_lite** mount option, enable WORM support with the Solaris command **chmod 4000** *directory-name*, where *directory-name* is the path and name of the directory that will hold the WORM files.

   The command **chmod 4000** sets the **setuid** (*set user ID upon execution*) attribute on the directory and enables standard WORM support. In the example, we WORM-enable the directory **/hsm/hsmfs1/documents** and check the result with **sls -Dd**. The operation succeeds and the directory is WORM-enabled:

   ```
   user@mds1:~# chmod 4000 /hsm/hsmfs1/documents
   user@mds1:~# sls -Dd /hsm/hsmfs1/documents
   /hsm/hsmfs1/documents
     mode: drwxr-xr-x    links:   2  owner: root        group: root
     length:      4096  admin id:      0  inode:     1049.1
     project: user.root(1)
     access:      Mar  3 12:28  modification: Mar  3 12:28
     changed:     Mar  3 12:28  attributes:   Mar  3 12:28
     creation:    Mar  3 12:28  residence:    Mar  3 12:28
     worm-capable        retention-period: 0y, 30d, 0h, 0m
   ```

4. If the directory is not WORM-enabled and if the file system was mounted with the **worm_emul** or **emul_lite** mount option, enable WORM support with the Solaris command **chmod 555** *directory-name*, where *directory-name* is the path and name of the directory that will hold the WORM files.

   The command **chmod 555** removes write permissions for the directory and enables WORM-emulation support. In the example, we WORM-enable the directory **/hsm/hsmfs1/documents** and check the result using the command **sls -Dd**. The operation succeeds and the directory is WORM-enabled:

```
user@mds1:~# chmod 555 /hsm/hsmfs1/documents
user@mds1:~# sls -Dd /hsm/hsmfs1/documents
/hsm/hsmfs1/documents
  mode: drwxr-xr-x    links:   2  owner: root        group: root
  length:      4096  admin id:      0  inode:     1049.1
  project: user.root(1)
  access:      Mar  3 12:28  modification: Mar  3 12:28
  changed:     Mar  3 12:28  attributes:   Mar  3 12:28
  creation:    Mar  3 12:28  residence:    Mar  3 12:28
  worm-capable        retention-period: 0y, 30d, 0h, 0m
```

Activate WORM Protection for a File

When you activate WORM protection on a file in a WORM-enabled directory, the file system no longer allows modifications to the file data or the path to the data until the retention period expires. So you must use care. To activate WORM protection, proceed as follows:

1. Log in to the file-system server.

```
user@mds1:~#
```

2. If you need to retain the file for some period other than the default for the file system, specify the required retention time by changing the access time for the file. Use the Solaris command **touch -a -t***expiration-date path-name*, where:

   - *expiration-date* is a string of numerals consisting of a four-digit year, a two-digit month, a two-digit day of the month, a two-digit hour of the day, a two digit minute within the hour, and, optionally, a two-digit second within the minute.
   - *path-name* is the path and name of the file.

   Note that Oracle Solaris UNIX utilities such as **touch** cannot extend a retention period beyond 10:14 PM on 01/18/2038. These utilities use signed 32–bit numbers to represent time in seconds starting from 01/01/1970. So use a default retention period if you need to retain files beyond this cut-off date.

   In the example, we set the retention period for the file to expire in four years, on October 4, 2019 at 11:59 AM:

```
user@mds1:~# touch -a -t201910141159  /hsm/hsmfs1/plans/master.odt
```

3. If the file system was mounted with the **worm_capable** or **worm_lite** mount option, activate WORM protection with the Solaris command **chmod 4000** *path-name*, where *path-name* is the path and name of the file.

   The **chmod 4000** command sets the **setuid** (*set user ID upon execution*) attribute on the specified file. Setting this attribute on an executable file is insecure. So, if the file system was mounted with the **worm_capable** or **worm_lite** mount option, you cannot set WORM protections on files that have UNIX **execute** permission.

In the example, we activate WORM protection for the file **master.odt**. We check the result with **sls -D**, and note that the **retention** attribute is now set to **active**, and the **retention-period** is set to four years:

```
user@mds1:~# chmod 4000 /hsm/hsmfs1/plans/master.odt
user@mds1:~# sls -Dd /hsm/hsmfs1/plans/master.odt
/hsm/hsmfs1/plans/master.odt:
  mode: -r-xr-xr-x    links:   1  owner: root        group: root
  length:       104  admin id:      0  inode:     1051.1
  project: user.root(1)
  access:      Mar  4 2018  modification: Mar  3 13:14
  changed:     Mar  3 13:16  retention-end: Apr  2 14:16 2014
  creation:    Mar  3 13:16  residence:    Mar  3 13:16
  retention:   active        retention-period: 4y, 0d, 0h, 0m
```

4. If the file system was mounted with the **worm_emul** or **emul_lite** mount option, activate WORM protection with the Solaris command **chmod 555** *path-name*, where *path-name* is the path and name of the file.

   The command **chmod 555** removes write permissions for the directory. So you can WORM protect executable files, if required. In the example, we activate WORM retention for the file **master-plan.odt**. We check the result with **sls -D**, and note that the **retention** attribute is now set to **active**, and the **retention-period** is set to four years:

```
user@mds1:~# chmod 555 /hsm/hsmfs1/plans/master.odt
user@mds1:~# sls -Dd /hsm/hsmfs1/plans/master.odt
/hsm/hsmfs1/plans/master.odt:
  mode: -r-xr-xr-x    links:   1  owner: root        group: root
  length:       104  admin id:      0  inode:     1051.1
  project: user.root(1)
  access:      Mar  4 2018  modification: Mar  3 13:14
  changed:     Mar  3 13:16  retention-end: Apr  2 14:16 2014
  creation:    Mar  3 13:16  residence:    Mar  3 13:16
  retention:   active        retention-period: 4y, 0d, 0h, 0m
```

Find and List WORM Files

To find and list WORM files that meet specified search criteria, use the **sfind** command. Proceed as follows:

1. Log in to the file-system server.

```
user@mds1:~#
```

2. To list files that are WORM-protected and being actively retained, use the command **sfind** *starting-directory* **-ractive**, where *starting-directory* is the path and name for the directory where you want the listing process to start.

```
user@mds1:~# sfind /hsm/hsmfs1/ -ractive
/hsm/hsmfs1/documents/2013/master-plan.odt
/hsm/hsmfs1/documents/2013/schedule.ods
/samma1/records/2013/progress/report01.odt
/samma1/records/2013/progress/report02.odt
/samma1/records/2013/progress/report03.odt ...
user@mds1:~#
```

3. To list WORM-protected files for which the retention period has expired, use the command **sfind** *starting-directory* **-rover**, where *starting-directory* is the path and name for the directory where you want the listing process to start.

```
user@mds1:~# sfind /hsm/hsmfs1/ -rover
/samma1/documents/2007/master-plan.odt
/samma1/documents/2007/schedule.ods
user@mds1:~#
```

4. To list WORM-protected files for which the retention period will expire after a specified date and time, use the command **sfind** *starting-directory* **-rafter** *expiration-date*, where:

   - *starting-directory* is the path and name for the directory where you want the listing process to start
   - *expiration-date* is a string of numerals consisting of a four-digit year, a two-digit month, a two-digit day of the month, a two-digit hour of the day, a two digit minute within the hour, and, optionally, a two-digit second within the minute.

   In the example, we list any files for which the retention period expires after January 1, 2015 at one minute after midnight:

   ```
   user@mds1:~# sfind /hsm/hsmfs1/ -rafter 201501010001
   /hsm/hsmfs1/documents/2013/master-plan.odt
   user@mds1:~#
   ```

5. To list WORM-protected files that must remain in the file system for at least a specified amount of time, use the command **sfind** *starting-directory* **-rremain** *time-remaining*, where:

   - *starting-directory* is the location in the directory tree where the search starts.
   - *time-remaining* is a string of non-negative integers paired with the following units of time: **y** for years, **d** for days, **h** for hours, **m** for minutes.

   In the example, we find all files under the directory **/hsm/hsmfs1/** that will be retained for at least three more years:

   ```
   user@mds1:~# sfind /hsm/hsmfs1/ -rremain 3y
   /hsm/hsmfs1/documents/2013/master-plan.odt
   user@mds1:~#
   ```

6. To list WORM-protected files that must remain in the file system for more than a specified amount of time, use the command **sfind** *starting-directory* **-rlonger** *time*, where:

   - *starting-directory* is the location in the directory tree where the search starts.
   - *time-remaining* is a string of non-negative integers paired with the following units of time: **y** for years, **d** for days, **h** for hours, **m** for minutes.

   In the example, we find all files under the directory **/hsm/hsmfs1/** that will be retained for more than three years and ninety days:

   ```
   user@mds1:~# sfind /hsm/hsmfs1/ -rremain 3y90d
   /hsm/hsmfs1/documents/2013/master-plan.odt
   user@mds1:~#
   ```

7. To list WORM-protected files that must remain in the file system permanently, use the command **sfind** *starting-directory* **-rpermanent**.

   In the example, we find that no files under the directory **/hsm/hsmfs1/** are being retained permanently:

```
user@mds1:~# sfind /hsm/hsmfs1/ -rpermanent
user@mds1:~#
```

7

Backing Up the Configuration and File Systems

When you installed and configured Oracle Hierarchical Storage Manager and StorageTek QFS Software, you created secure locations for storing recovery point files and copies of the archiver log. You also configured automated processes for creating recovery points, backing up the logs, and protecting the system configuration. These steps provide the core protection for your file systems. But you may also need to take unscheduled protective measures from time to time as well:

- Back up the Oracle HSM configuration and file systems prior to anticipated, potentially disruptive events, such as major changes to physical infrastructure or data center facilities.
- Back up the Oracle HSM configuration and file systems after upgrading or reconfiguring software, operating systems, or host platforms, so that the current configuration is protected.
- Gather required configuration and status information prior to engaging Oracle HSM support services.

This chapter outlines the procedures for collecting, creating, and storing configuration and file-system recovery files on an as-needed basis:

- backing up file systems
- backing up the Oracle HSM configuration
- gathering configuration and diagnostic information with **`samexplorer`**.

Note that this chapter uses the command line interface for all tasks. If you wish to use the Oracle HSM Manager graphical user interface, refer to the online help for detailed procedures.

Backing Up File Systems

This section starts with a brief review of Oracle HSM file system protection. Then it provides procedures for carrying out the following tasks:

- creating a recovery point
- backing up the archiver log
- gathering configuration and diagnostic information with **`samexplorer`**
- manually back up the Oracle HSM configuration.

Understanding Recovery Points and Archive Logs

To protect a file system, you need to do two things:

- You must protect the files that hold your data.
- You must protect the file system itself, so that you can use, organize, locate, access, and manage your data.

In an Oracle HSM archiving file system, file data is automatically protected by the archiver: modified files are automatically copied to archival storage media, such as tape. But if you backed up only your files and then suffered an unrecoverable failure in a disk device or RAID group, you would have the data but no easy way to use it. You would have to create a substitute file system, identify each file, determine its proper location within the new file system, ingest it, and recreate lost relationships between it and users, applications, and other files. This kind of recovery is, at best, a daunting and long drawn-out process.

So, for fast, efficient recovery, you have to actively protect the file-system metadata that make files and archive copies usable. You must back up directory paths, inodes, access controls, symbolic links, and pointers to copies archived on removable media.

You protect Oracle HSM file-system metadata by scheduling *recovery points* and saving archive logs. A recovery point is a compressed file that stores a point-in-time backup copy of the metadata for an Oracle HSM file system. In the event of a data loss—anything from accidental deletion of a user file to catastrophic loss of a whole file system—you can recover to the last known-good state of the file or file system almost immediately by locating the last recovery point at which the file or file system remained intact. You then restore the metadata recorded at that time and either stage the files indicated in the metadata to the disk cache from archival media or, preferably, let the file system stage files on demand, as users and applications access them.

Like any point-in-time backup copy, a recovery point is seldom a complete record of the state of the file system at the time when a failure occurs. Inevitably, at least a few files are created and changed after one recovery point is completed and before the next one is created. You can—and should—minimize this problem by scheduling creation of recovery points frequently and at times when the file system is not in use. But, in practice, scheduling has to be a compromise, because the file system exists to be used.

For this reason, you must also save point-in-time copies of the archiver log file. As each data file is archived, the log file records the volume serial number of the archival media, the archive set and copy number, the position of the archive (**`tar`**) file on the media, and the path to and name of the data file within the **`tar`** file. With this information, you can recover any files that are missing from the recovery point using Solaris or Oracle HSM **`tar`** utilities. However, this information is volatile. Like most system logs, the archiver log grows rapidly and must thus be overwritten frequently. If you do not make regular copies to compliment your recovery points, you will not have log information when you need it.

Create a Recovery Point on Demand

Sometimes you may need to capture the metadata from an archiving file system at a point in time outside your normal schedule. Whenever you anticipate potentially disruptive system or facilities maintenance, for example, you can create before and after recovery points to make sure that file systems are protected.

To initiate creation of an unscheduled, on-demand recovery point on demand, proceed as follows:

1. Log in to the Oracle HSM server host as **`root`**.

   ```
   root@mds1:~#
   ```

2. Select an independent location where the recovery point will be stored.

In the example, we create a subdirectory, **unscheduled/**, under the directory that we created for recovery points when we initially configured file systems. The **/zfs1** file system is remotely located and has no components in common with the Oracle HSM file system:

```
root@mds1:~# mkdir /zfs1/samqfs_recovery/unscheduled
root@mds1:~#
```

3. Change to the file system's root directory.

   In the example, we change to the mount-point directory **/samqfs**:

   ```
   root@mds1:~# cd /samqfs
   root@mds1:~#
   ```

4. If you are backing up an archiving file system where the data is copied to removable media, back up the metadata only. Use the command **samfsdump -f** *recovery-point*, where *recovery-point* is the path and file name of the finished recovery point file.

   See the **samfsdump** man page for additional details. In the example, we create an unscheduled recovery point for the **samqfs** file system prior to a scheduled, maintenance-related power outage. We create the recovery-point file **20161215pre-outage** in the directory **/zfs1/samqfs_recovery/unscheduled/**:

   ```
   root@mds1:~# cd /samqfs
   root@mds1:~# samfsdump -f /zfs1/samqfs_recovery/unscheduled/20161215pre-outage
   root@mds1:~#
   ```

5. If you are backing up a standalone file system where data is not copied to removable media, back up both the metadata and the data. Use the command **samfsdump -U -f** *recovery-point*, where *recovery-point* is the path and file name of the finished recovery point file.

   Note that recovery-point files that include data as well as metadata can be extremely large. See the **samfsdump** man page for additional details. In the example, we create an unscheduled recovery point for the **samqfs** file system. We create the recovery-point file **20161215pre-outage** in the remote directory **/zfs1/samqfs_recovery/unscheduled/**:

   ```
   root@mds1:~# cd /samqfs
   root@mds1:~# samfsdump -f -U /zfs1/samqfs_
   recovery/unscheduled/20161215pre-outage
   root@mds1:~#
   ```

6. If you are backing up an archiving file system, back up the archiver log.
7. Otherwise, depending on the situation, you may also want to run **samexplorer** and manually back up the Oracle HSM configuration.

Back Up the Archiver Log

While recovery point files contain almost all of the information needed when restoring a file system, they do not hold the metadata for files created or modified after the recovery point was created. Because archiver logs list all of the files that have been archived and their locations on cartridges, archiver logs can be used to recover any files that were archived after the creation of the recovery point. So, if possible, create an unscheduled copy of the archiver log file whenever you create an unscheduled recovery point. Proceed as follows.

1. Log in to the Oracle HSM server host as **root**.

```
root@mds1:~#
```

2. Select an independent location where the backed up archiver log will be stored.

   In the example, we decide to store the log in the same directory as the corresponding unscheduled recovery point that we created above. The **/zfs1** file system is remotely located and has no components in common with the Oracle HSM file system:

```
root@mds1:~# ls /zfs1/samqfs_recovery/unscheduled
20161215pre-outage
root@mds1:~#
```

3. Copy the current archiver log to the chosen location and give it a unique name. Use the command **cp /var/adm/samqfs.archive.log** *path*/**"date +%y%m%d";**, where *path* is the path to the chosen location.

```
root@mds1:~# cp /var/adm/samqfs.archive.log /zfs1/samqfs_
recovery/unscheduled/20161215pre-outage/"date +%y%m%d".archive.log
root@mds1:~#
```

4. Depending on the situation, you may also want to run **samexplorer** and manually back up the Oracle HSM configuration.

Backing Up the Oracle HSM Configuration

Whenever you change the Oracle HSM configuration, protect your investment by backing up all modified configuration files and related information. Carry out the following tasks:

- back up the Oracle HSM configuration files
- run **samexplorer**.

Manually Back Up the Oracle HSM Configuration

For full redundancy, create a local copy of the configuration files whenever you make significant changes to software, operating systems, or hosts. Proceed as follows:

1. Log in to the file-system host as **root**.

```
root@mds1:~#
```

2. In the subdirectory that holds your backup configuration information, create a subdirectory for manual backup copies of your Oracle HSM configuration. Use the command **mkdir** *mount-point*/*path*, where *mount-point* is the mount point directory for the selected independent file system and *path* is the path and name of the chosen directory.

   In the example, we are configuring recovery points for the archiving file system **/samqfs**. So we have created the directory **/zfs1/sam_config/samconfig**:

```
root@mds1:~# mkdir /zfs1/sam_config/samconfig
```

3. In the subdirectory that holds manual backup copies of your Oracle HSM configuration, create a subdirectory for the current Oracle HSM configuration. Use the command **mkdir** *mount-point*/*path*/*subdirectory*, where *mount-point* is the mount point for the selected independent file system and *path*/*subdirectory* is the path and name of the chosen subdirectory.

In the example, we create a subdirectory in the directory that we created for this purpose during initial configuration, **/zfs1/sam_config/samconfig**. We use the date to name the subdirectory:

```
root@mds1:~# mkdir /zfs1/sam_config/samconfig/20161215
```

4. Copy the configuration files to another file system.

```
/etc/opt/SUNWsamfs/
     mcf
     archiver.cmd
     defaults.conf
     diskvols.conf
     hosts.family-set-name
     hosts.family-set-name.local
     preview.cmd
     recycler.cmd
     releaser.cmd
     rft.cmd
     samfs.cmd
     stager.cmd
     inquiry.conf
     samremote                  # SAM-Remote server configuration file
     family-set-name            # SAM-Remote client configuration file
     network-attached-library   # Parameters file
     scripts/*                  # Back up all locally modified files
/var/opt/SUNWsamfs/
```

5. Back up all library catalog data, including that maintained by the historian. For each catalog, use the command **/opt/SUNWsamfs/sbin/dump_cat -V** *catalog-file*, where *catalog-file* is the path and name of the catalog file. Redirect the output to *dump-file*, in a new location.

   In the example, we dump the catalog data for **library1** to the file **library1cat.dump** in a directory on the independent NFS-mounted file system **zfs1**:

```
root@mds1:~# dump_cat -V /var/opt/SUNWsamfs/catalog/library1cat > /zfs1/sam_
config/20161215/catalogs/library1cat.dump
```

6. Copy system configuration files that were modified during Oracle HSM installation and configuration. These may include:

```
/etc/
     syslog.conf
     system
     vfstab
/kernel/drv/
     sgen.conf
     samst.conf
     samrd.conf
     sd.conf
     ssd.conf
     st.conf
/usr/kernel/drv/dst.conf
```

7. Copy any custom shell scripts and **crontab** entries that you created as part of the Oracle HSM configuration to the selected subdirectory.

   For example, if you created a **crontab** entry to manage creation of recovery points, you would save a copy now.

8. Copy any custom shell scripts and **crontab** entries that you created as part of the Oracle HSM configuration to the selected subdirectory.

   For example, if you created a **crontab** entry to manage creation of recovery points, you would save a copy now.

9. Record the revision level of the currently installed software, including Oracle HSM, Solaris, and Solaris Cluster (if applicable), and save a copy of the information in a **readme** file in the chosen subdirectory.

10. In the chosen subdirectory, save copies of downloaded Oracle HSM Oracle HSM, Solaris, and Solaris Cluster packages so that you can restore the software quickly, should it become necessary.

11. Next, run **samexplorer**.

Gathering Configuration and Diagnostic Information with samexplorer

The **samexplorer** is a diagnostic tool that captures and reports comprehensive configuration and status information for the Oracle HSM software and file systems. Whenever you make changes to your Oracle HSM configuration, you should run **samexplorer** and store the resulting report with your backup copies of the configuration files. You should also run **samexplorer** when you are troubleshooting a problem and whenever you are requested to do so by Oracle HSM support services personnel. Proceed as follows:

Run samexplorer

1. Log in to the file-system host as **root**.

2. In the directory that holds your backup configuration information, create a subdirectory for **samexplorer** reports. Use the command **mkdir** *mount-point*/*path*, where *mount-point* is the mount point directory for the selected independent file system and *path* is the path and name of the chosen directory.

   In the example, we create the new subdirectory in the directory that we created for this purpose during initial configuration, **/zfs1/sam_config/**. We name the new subdirectory **explorer/**:

   ```
   root@mds1:~# mkdir /zfs1/sam_config/explorer
   ```

3. Create the **samexplorer** report in the selected directory. Use the command **samexplorer** *path*/*hostname*.*YYYYMMDD*.*hhmmz*.**tar.gz**, where *path* is the path to the chosen directory, *hostname* is the name of the Oracle HSM file system host, and *YYYYMMDD*.*hhmmz* is a date and time stamp.

   The default file name is **/tmp/SAMreport.***hostname*.*YYYYMMDD*.*hhmmz*.**tar.gz**. In the example, we create a report for the host **samhost1** dated December 15, 2016, at 4:59 PM Mountain time:

   ```
   root@mds1:~# samexplorer /zfs1/sam_
   config/explorer/samhost1.20161215.1659MST.tar.gz

        Report name:     /zfs1/sam_
   config/explorer/samhost1.20161215.1659MST.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.

/zfs1/sam_config/explorer/samhost1.20161215.1659MST.tar.gz
```

4. Repeat this procedure whenever you significantly reconfigure your file systems.
5. Stop here. The Oracle HSM configuration is backed up.

8

Migrating to New Storage Media

Types of Migration

Automated Media Migration

Automated Media Migration is ideal when you need to transfer complete volumes of archival data, from tape to tape or from tape to a cloud volume. It is also ideal for adding a cloud-based archive copy to the archive set.

The Automated Media Migration feature introduced in Oracle HSM 6.1 copies full volumes from media mounted on one library drive to media mounted on another, updating the file-system metadata in the process (manually loaded drives are not supported). This minimizes system overhead and administrator workload. Volumes are copied in the background, when drives are not required for archiving or staging. You can specify the number of drives used and the times of day when migration may occur. Or you can let Oracle HSM migrate volumes whenever drives are idle. If a drive or volume is needed for an archiving or staging job, the media migration process yields to the higher priority operation. If you have correctly configured, StorageTek T10000D tape drives, you can migrate to T10000D using the StorageTek Direct Copy (**xcopy**) method. Once a request is made, the drives handle the copying themselves, using no server resources. Otherwise, you can still minimize server load by using the StorageTek Memory Assisted Copy method. The file-system server then copies volumes from drive-to-drive via a configurable I/O buffer.

Staging and Rearchiving Process

The staging and rearchiving processes that handle ordinary archive management tasks are ideal when you need to transfer archival data selectively, directory by directory and file by file.

The older staging and rearchiving approach stages archive files from the old media to the disk cache and then rearchives them to the new media, one file at a time. This file-by-file approach can give you more control over how files are grouped and distributed. But it requires more administration. You allocate disk cache and drive resources yourself, so you must plan carefully if you need to minimize interference with normal file-system operations.

Preparing for Migration

Make Sure that File Systems Stay Backed Up

Before you start a media migration, make sure that the recovery mechanisms that normally protect Oracle HSM archive data remain effective during and following the changeover. Catastrophic hardware failures and user errors are neither more or less likely during a migration operation. So, as always, you need to be sure that you can recover files and/or complete file systems from your extant, **`samfsdump`** recovery-point files.

During the migration and for some time after, recovery will depend on recovery point files that reference the source tape volumes, rather than the new, destination volumes. If a major hardware failure disables the file system and these old tapes are not available, you will not be able to recover.

So, at a minimum, plan on preserving old tapes until you have created enough new recovery points to restore the current file system from new media. If you need to be able to restore files to particular points in time, you may need to retain the old media longer, if not indefinitely. Ideally, the old volumes should be maintained in a library, where they are readily accessible.

Provide the Required Media

Make sure that you have prepared the required destination media before you begin migration. If you are migrating volumes or volume copies to the cloud, configure and provision the cloud library. If you are migrating volumes to new tape volumes, make sure that the destination library contains enough media to hold the migrated files. Make sure that all volumes are correctly labeled, as described in "Label a New Tape or Relabel an Existing Tape" on page 5-24. Migration will fail if volumes are unlabeled.

Select the Migration Approach that Best Meets Your Needs

1. Migrate data by staging and re-archiving files if the following considerations apply:

   - You need to selectively migrate groups of archive files, rather than whole volumes, so that relationships between groups of archive files are maintained.
   - File system performance is not an issue.

     You have enough disk space and enough removable media drives to simultaneously handle normal file system activity and the additional migration-related staging and archiving requests (see "Estimate The Resources Available for Staging and Re-Archiving" on page 8-15).

     Or users and host applications can accept reduced file system performance during the migration.

2. Otherwise, select a volume migration method.

   Volume migration is especially advantageous when any of the following considerations apply:

- You need to migrate data while maintaining normal file system operations.
- You need to copy old volumes to new tape or cloud media.
- You need to economize on disk space and/or removable media drives during migration.
- You need to add a cloud-based copy to an existing archive set.

Select a Volume Migration Method

Oracle HSM can copy volumes in either of two ways, StorageTek Direct Copy or Memory Assisted Copy.

The StorageTek Direct Copy method maximizes performance and minimizes server overhead. When you specify this approach, the file-system server sends a SCSI copy request to the drive, and the T10000D drive copies the source to the destination tape, block-by-block, starting from the first valid archive file. The server plays no further role in the transfer. If the operation fails for any reason, the migration software switches automatically to the Memory Assisted Copy method.

The Memory Assisted Copy method copies valid archive files from the source drive to a configurable I/O buffer on the file-system server. If source and destination block sizes differ, the software automatically makes adjustments, as long as the destination block size is larger. The software then sends tape blocks from the buffer to the destination drive. The direct, tape-to-tape xcopy method can only copy tapes that share a common block size.

Select the method that best suits your requirements:

1. Plan to use the StorageTek Direct Copy migration method if all of the following apply:
   - You will migrate volumes to Fibre Channel StorageTek T10000D destination drives running current firmware.
   - Source and target tapes share the same block size.
   - Source and destination drives connect via the same storage area network (SAN) switch.
   - You are not using multiple paths to the tape drive through the SAN switch/fabric. Tape multi-path is not supported with StorageTek Direct Copy.

   For more information on drive and firmware requirements, see the release notes, **`README.txt`**, in the download ZIP file or on your file-system server at **`/opt/SUNWsamfs/doc/README.txt`**
2. Otherwise, plan to use the Memory Assisted Copy migration method.

   Use Memory Assisted Copy if any of the following considerations apply:
   - The destination drive is not a Fibre Channel Oracle StorageTek T10000D drive.
   - The source and destination drives are not Fibre Channel drives.
   - The source and destination drives do not run current firmware.
   - The source and destination drives do not connect via the same storage area network (SAN) switch.
   - The source and destination tapes do not share a common block size.
3. If you will use StorageTek Direct Copy, select a copy mode now.

4. If you will use Memory Assisted Copy, configure the migrationd.cmd file accordingly.

Select a StorageTek Direct Copy Mode

1. If you are planning to migrate source volumes that contain relatively few expired files, plan on using the **eod** (end-of-data) option.

   In this mode, the T10000 drive copies all archive (**tar**) files found between the first valid file and the end-of-data (EOD) mark on the tape. If some of these files are stale, they are copied to the destination volume with the valid files.

2. If you are planning to migrate source volumes that contain many expired files, use the repack option.

   In this mode, the T10000 drive copies only archive (**tar**) files that hold at least one unexpired archive copy.

3. Now configure the migrationd.cmd file for StorageTek Direct Copy.

Migrating Complete Volumes

You select the StorageTek Direct Copy or Memory Assisted Copy method and configure migration by creating the **migrationd.cmd** file. Carry out the following tasks:

- create the **migrationd.cmd** file
- check for active migration jobs
- migrate volumes.

Create the `migrationd.cmd` Configuration File

1. Log in to the Oracle HSM metadata server as **root**.

   ```
   root@solaris:~#
   ```

2. Open the **/etc/opt/SUNWsamfs/migrationd.cmd** file in a text editor.

   In the example, we open the new file in the **vi** editor:

   ```
   root@solaris:~# vi /etc/opt/SUNWsamfs/migrationd.cmd
   ```

3. Define media pools for the source and destination volumes. Define each pool by entering a line of the form **vsnpool = poolname library equipment_number media_type VSNlist**, where:

   - **poolname** uniquely identifies the pool.
   - **equipment_number** is the ordinal number that the **mcf** file assigns to the library that holds the source volumes.
   - *media_type* is the two-letter code that identifies the kind of media that holds the source (see Appendix A, "Glossary of Equipment Types" for details).
   - **VSNlist** is a space-delimited list made up of VSNs that individually identify tape volumes and/or VSN-based regular expressions that collectively identify groups or ranges of tape or cloud media volumes.

     Cloud media pools are identified by regular expressions of the form **^prefix.***, where **prefix** is the value of the **name** parameter in the cloud library's parameters file. During migration, Oracle HSM creates cloud

volumes and generates volume serial numbers as needed, using the value of the **name** parameter as the prefix that starts each VSN.

In the example, we first need to migrate recent data from old LTO6 (**li**) tape volumes to replacement LTO7 (**ti**) tape cartridges. So we add a line for a source pool, pool1, that represents the LTO6 volumes that will migrate from library **20**. These include volumes that have VSNs in the range **VOL000** to **VOL299** and two single volumes, **VOL300** and **VOL304**. Then we add a line for a destination pool, **pool2**, which represents a range of LTO7 volumes in library **30**:

```
# pool1 contains the source volumes
vsnpool = pool1 library 20 li ^VOL[0-2][0-9][0-9] VOL300 VOL304
# pool2 contains the destination volumes
vsnpool = pool2 library 30 li ^VOL50[0-9]
```

Then we need to migrate older data from the remaining LTO6 tape volumes to cloud (**cl**) media for long-term storage. We add a line for a source pool, **pool3**, that represents the LTO6 volumes in library **20**. Then we add a line for a destination pool, **pool4**, which represents the cloud (**cl**) media volumes in cloud (**cr**) library **800** (family set name **cl800**). To identify the cloud media, we supply a regular expression that specifies any VSNs starting with **cl800**, the value of the **name** parameter listed in the cloud library's parameters file, **/etc/opt/SUNWsamfs/cl800**:

```
# pool3 contains the source volumes
vsnpool = pool3 library 20 li ^VOL6[0-9][0-9]
# pool4 contains the destination volumes
vsnpool = pool4 library 800 cl ^cl800.*
```

4. If you need to move data from source to destination media, enter a line of the form **migrate = from** *sourcepool* **to** *destinationpool*, where:

   - *sourcepool* is the media pool that contains the data that will be migrated.
   - *destinationpool* is the media pool that will receive the migrated data

   In the example, we migrate data from **pool1** to **pool2** and from **pool3** to **pool4**.

   ```
   # Migrate data from tapes in pool1 to tapes in pool2.
   migrate = from pool1 to pool2
   # Migrate data from tapes in pool1 to tapes in pool2.
   migrate = from pool3 to pool4
   ```

5. If you need to add an archive copy by copying data from source to destination media, enter a line of the form **migrate = add_copy copy_number from sourcepool to destinationpool**, where:

   - **copy_number** is either a specific archive copy number (in the range [**1-4**]) or **0**, which specifies the next available copy number.

     For example, if the **archiver.cmd** file already specifies two copies for the archive set, a copy number of **0** is equivalent to copy number **3**. If the **archiver.cmd** file already specifies the maximum number of supported copies, then the migration process returns an error.

   - **sourcepool** is the media pool that holds the local, on-site copies of files that will be vaulted off-site.
   - **destinationpool** is the media pool that will receive copies of the on-site files for export from the library and vaulting off-site.

In the example, we add archive copy number **4** to the archiving configuration by copying previously archived volumes from **pool3** to the pool of cloud media, **pool4**:

```
migrate = add_copy 4 from pool3 to pool4
```

6. If you need to move an archive copy from one copy to another, enter a line of the form **migrate = move_copy copy_number from sourcepool to destinationpool**. This may be useful in circumstances where you want to take an existing copy and move it to a different media type (off-site or cloud). It may be useful also if a mis-configured archiver has made copies on the wrong media pool.

   - **copy_number** is either a specific archive copy number (in the range [**1-4**]) or **0**, which specifies the next available copy number.

     For example, if the **archiver.cmd** file already specifies two copies for the archive set, a copy number of **0** is equivalent to copy number **3**. If the **archiver.cmd** file already specifies the maximum number of supported copies, then the migration process returns an error.

   - **sourcepool** is the media pool that holds the local, on-site copies of files that will be vaulted off-site.

   - **destinationpool** is the media pool that will receive copies of the on-site files for export from the library and vaulting off-site.

   In the example, we move copy number **4** from **pool3** to **pool4**:

   ```
   migrate = move_copy 4 from pool3 to pool4
   ```

7. If you plan to use the StorageTek Memory Assisted Copy method exclusively, enter a line of the form **xcopy = off**.

   When the **xcopy** directive is set to **off**, StorageTek Memory Assisted Copy is disabled.

   ```
   # Disable StorageTek Direct Copy and use Memory Assisted Copy only
   xcopy = off
   ```

8. If you plan to use the StorageTek Direct Copy feature exclusively and do not intend to migrate data when drives that support this feature are unavailable, enter a line of the form **xcopy = only**.

   When the **xcopy** directive is set to **only**, the migration software uses the StorageTek Direct Copy method whenever the source and destination drive support the StorageTek Direct Copy feature and automatically cancels migration otherwise.

   ```
   # Enable StorageTek Direct Copy but cancel migration if
   # the source or destination drive does not support it.
   xcopy = only
   ```

9. If you plan to use StorageTek Direct Copy when possible and Memory Assisted Copy otherwise, enter a line of the form **xcopy = on**.

   When the **xcopy** directive is set to **on**, the migration software uses the StorageTek Direct Copy method whenever compatible drives are available. If either the source or destination drive does not support StorageTek Direct Copy, the migration software automatically switches to Memory Assisted Copy.

   ```
   # Enable StorageTek Direct Copy feature but
   # If the source or destination is compatible with StorageTek Direct Copy,
   # automatically switch to Memory Assisted Copy.
   ```

```
xcopy = on
```

10. If you need to migrate tape volumes that contain few expired files using StorageTek Direct Copy, enable end-of-data (**eod**) mode. Enter a line of the form **xcopy_eod = on**.

    ```
    xcopy = on
    xcopy_eod = on
    ```

11. If you need to migrate tape volumes that contain that contain significant numbers of expired files using StorageTek Direct Copy, enable repack mode. Enter a line of the form **xcopy_eod = off**.

    ```
    xcopy = on
    xcopy_eod = off
    ```

12. Specify the minimum amount of data that must be copied before **xcopy** can be interrupted by a higher priority, archiving or staging task. Enter a line of the form **xcopy_minsize =** *amountUnits*, where:

    - *amount* is an integer.
    - *Units* is **k** for kilobytes, **M** for megabytes, **G** for gigabytes, **T** for terabytes, **P** for petabytes, or **E** for exabytes.

    This value defines a compromise between efficient utilization of T10000 drives and availability of drives for more other tasks. Larger values write data to the drives more efficiently. Smaller values increase the availability of drives for archiving and staging. In the example, we set the minimum copy size to 30 gigabytes:

    ```
    root@solaris:~# vi /etc/opt/SUNWsamfs/migrationd.cmd
    ...
    xcopy_eod = on
    # xcopy can be interrupted after 30GB copied.
    xcopy_minsize = 30G
    ```

13. Define the daily period during which migration jobs are allowed to run. Enter a line of the form **runtime =** *window*, where *window* is one of the following values:

    - **always** lets the migration daemon migrate data whenever drives and media are not required for archiving or staging. If the migration daemon is using drives or media when they are needed for archiving or staging, the migration daemon yields them.
    - *start_time end_time*, where *start_time* and *end_time* are, respectively, the times when the allowed period begins and ends, expressed as hours and minutes on a 24-hour clock (*HHMM*).

    You can override this directive at any time by issuing the commands **samcmd**, **migstart**, **migidle**, or **migstop**.

    The migration service relinquishes volumes and drives when either are required by the stager or archiver. So, unless you experience problems with archiving or staging (for example, during peak hours), you should accept the default value, **always**:

    ```
    root@solaris:~# vi /etc/opt/SUNWsamfs/migrationd.cmd
    ...
    xcopy_minsize = 30G
    # Run all of the time. Migration daemon will yield VSNs and drives when
    # resources are wanted by the SAM-QFS archiver and stager.
    run_time = always
    ```

14. Enable logging by specifying a log directory. Enter a line of the form **logdir =** *path*, where *path* is the directory path and directory name.

    When the directory is defined, the migration daemon logs the destination of each archive file that migrates from each source volume. Each source volume has its own log file, named *media_type*.*VSN* where:

    - *media_type* is a two-letter code that identifies the kind of source media (see Appendix A, "Glossary of Equipment Types" for details).
    - *VSN* is the unique volume serial number that identifies the source volume.

    So for example, the log file for the source volume with the VSN **VOL300** would be named **li.VOL300**.

    Like the archiver log, these migration logs can be invaluable during disaster recovery (see "Understanding Recovery Points and Archive Logs" on page 7-1 and the *Oracle Hierarchical Storage Manager and StorageTek QFS Software File System Recovery Guide* for details). So always specify a log directory if you can. Select a location that will not be affected by the failure of Oracle HSM software or hardware, such as **/var/adm/**. In the example, we specify the directory **/var/adm/hsm_migration_logs**:

    ```
    root@solaris:~# vi /etc/opt/SUNWsamfs/migrationd.cmd
    ...
    run_time = always
    # Log directory for the migration logs.
    logdir = /var/adm/hsm_migration_logs
    ```

15. Specify a home directory for migration inode databases. Enter a line of the form **dbdir =** *path*, where *path* is an absolute directory path name.

    An inode database is created for each source volume and maintained for the duration of the migration. One 224-byte database record is created for each archive copy found on the source volume. So you must choose a location that has enough disk space to accommodate the largest number of copies that could fit on the source media. For example, each Oracle StorageTek T10000D volume might hold up to 8,200,104,892 archive copies. So you would need about 1.67 terabytes of database space for each T10000D volume that would migrate at any given time (see the **migration.cmd** (1m) man page for details).

    The default database location is **/var/opt/SUNWsamfs/sammig/db**. In the example, we specify the default directory:

    ```
    root@solaris:~# vi /etc/opt/SUNWsamfs/migrationd.cmd
    ...
    logdir = /var/adm/hsm_migration_logs
    # database home directory
    dbdir = /var/opt/SUNWsamfs/sammig/db
    ```

16. Set the migration buffer size for the destination device. Enter a line of the form **buffsize =** *media_type blocks*, where:

    - *media_type* is the two-letter code that identifies the kind of media that holds the source (see Appendix A, "Glossary of Equipment Types" for details).
    - *size* is an integer in the range **[2-8192]**, where the integer value specifies the number of the tape blocks that the buffer should be able to hold. The default is **64**.

In the example, we allocate enough space to hold the default number of Oracle StorageTek T10000 tape blocks:

```
root@solaris:~# vi /etc/opt/SUNWsamfs/migrationd.cmd
...
# database home directory
dbdir = /var/opt/SUNWsamfs/sammig/db
# allocate buffer space for 64 T10000D tape blocks
bufsize = ti 64
```

17. Specify the maximum number of drives that can be used for migration per library. Enter a line of the form **max_drives =** *library-list*, where:

    - *library-list* is a space-separated list of library entries, each of the form **library** *equipment-number device-count*.
    - *equipment-number* is the equipment ordinal number assigned to the library in the **mcf** file.
    - *device-count* is the number of drives that can be used in the specified library. By default, *device-count* is set to the number of drives in the library.

    The migration service relinquishes volumes and drives when either are required by the stager or archiver. So, unless you experience problems with archiving or staging, you should accept the default setting and allow migration to use any drive that is free. In the example, we have found that we do, in fact, need to limit drive usage. So we allocate eight drives to migration in library **20**, six in library **30**, and two in library **40**:

    ```
    root@solaris:~# vi /etc/opt/SUNWsamfs/migrationd.cmd
    ...
    dbdir = /var/opt/SUNWsamfs/sammig/db
    # allocate buffer space for 64 T10000D tape blocks
    bufsize = ti 64
    # For migration, use 8 drives in library 20, 6 in 30, and 2 in 40
    max_drives = library 20 8 library 30 6 library 40 2
    ```

18. Specify the maximum number of migration-related copy operations that be run at the same time. Enter a line of the form **max_copy =** *processes*, where *processes* is an integer.

    The default is the maximum value, which is equal to the number of configured drives in the all libraries listed in the **mcf** file divided by 2. In the example, we allow up to eight simultaneous copy processes:

    ```
    root@solaris:~# vi /etc/opt/SUNWsamfs/migrationd.cmd
    ...
    bufsize = ti 64
    # For migration, use 8 drives in library 20, 6 in 30, and 2 in 40
    max_drives = library 20 8 library 30 6 library 40 2
    # Up to 8 sam-migcopy process can be run simultaneously.
    max_copy = 8
    ```

19. Specify the maximum number of migration-related tape-scanning operations that be run at the same time. Enter a line of the form **max_scan =** *processes*, where *processes* is an integer.

    To identify archive copies on the migration source VSNs, the **sam-migrationd** daemon scans all file systems configured in the **mcf**, reads all inodes from the disk cache, and compares the **vsn** field in each inode to the Volume Serial Numbers (VSNs) of migration source volumes. The process increases metadata activity in the file system and may thus adversely affect file-system performance.

Select a value that best balances acceptable file-system performance against speed of migration, or accept the default, **4**, for most uses. If you intend to quiesce the file system in order to achieve the fastest migration, set **max_scan** to **0**, so that all source volumes are scanned at once. In the example, we know from experience that we can allow up to eight simultaneous scanning processes without affecting our normal file-system operations:

```
root@solaris:~# vi /etc/opt/SUNWsamfs/migrationd.cmd
...
bufsize = ti 64
# For migration, use 8 drives in library 20, 6 in 30, and 2 in 40
max_drives = library 20 8 library 30 6 library 40 2
# Run up to 8 sam-migcopy processes simultaneously.
max_copy = 8
# Scan up to 8 VSNs simultaneously.
max_scan = 8
```

20. Save the file, and close the editor.

```
root@solaris:~# vi /etc/opt/SUNWsamfs/migrationd.cmd
...
max_copy = 8
# Scan up to 8 VSNs simultaneously.
max_scan = 8
:wq
root@solaris:~#
```

Check for Active Migration Jobs

The instructions in this section describe entering commands from the shell command prompt using the **samcmd** command. But please note that all commands can also be entered from the **samu** interface as well, in the form **:***command*, where *command* is the name of the command.

1. If you are not currently logged in to the Oracle HSM metadata server as **root**, log in now.

```
root@solaris:~#
```

2. Make sure that a previous migration is not currently active or incomplete. First, check current migration status. Use the command **samcmd x**.

   If another migration copy job is under way, the command lists the source and destination volumes, by media type and VSN, the copy mode, the percentage complete, and the current status of the copy:

```
root@solaris:~# samcmd x
Migration status   samcmd  version HH:MM:SS month day year
samcmd on hsm61sol
Status: Stop: Waiting for :migstart
source    dest    cmod perc status
li VOL004 li VOL042 -   60% Copy idled
```

   Otherwise, if no other migration copy jobs are running, the command lists no jobs:

```
root@solaris:~# samcmd x
Migration status     samu      ver  time date
Source Vsns - wait:  0 fsscan: 0 copy: 0 update ino: 0 log: 0 done:  0
Status: Idle: Waiting for :migstart
source  dest  cmod  perc  status
```

3. Next, check the status of any current source (**S**) and/or destination (**D**) volumes. Use the command **samcmd y**.

   In the first example, the job **end time** for the only source and destination volumes listed is **10/16 12:14**. The copy of the source volume is complete. So no jobs are currently running:

```
root@solaris:~# samcmd y
Migration vsn list   samcmd  version HH:MM:SS month day year
Status:  Run  Vsns:2 src:1 dest:1 maxcopy:2
ord m ty vsn     start time  end time    status  Inodes done/tot    bytes
  0 S li VOLa01  10/16 12:12 10/16 12:14 complete    35023/35023   550.00M
  0 D li VOLa80  10/16 12:12 10/16 12:14 avail                     550.00M
```

   In the second example, the job **end time** for the source and destination volumes is **none**. The source volume is still being copied to the destination volume. So a migration job is still running:

```
root@solaris:~# samcmd y
Migration vsn list   samcmd  version HH:MM:SS month day year
Status:  Run  Vsns:2 src:1 dest:1 maxcopy:2
ord m ty vsn     start time  end time  status  Inodes done/tot    bytes
  0 S li VOLa02  10/16 12:12 none      copy            0/35023   164.50M
  0 D li VOLa81  10/16 12:12 none      copy                      148.75M
```

4. Finally, check the volume listings in the library catalog. Use the command **samcmd v**. Look for the following flags in the output:

   - **R** means that the volume is read-only. When migration starts, source volumes are marked read-only.
   - **S** (*source*) means that data is still being copied from this volume.
   - **D** (*destination*) means that data is still being copied to this volume.
   - **m** means that the source volume has finished migration.
   - **e** means that the source volume failed to migrate due to an error.

   In the example, volume **VOLa01** migrated to **VOLa80** successfully. Volume **VOLa02** is still migrating to **VOLa81**. Volume **VOLa03** failed to migrate.

```
root@solaris:~# samcmd v
Robot catalog   samcmd  version HH:MM:SS month day year
Robot VSN catalog by slot       : eq 800
slot          access time count use flags         ty vsn
count 64
   0     2015/06/29 17:00    1  95%  -il---b--Rm-  li VOLa01
   1     2015/07/02 17:43    2  89%  -il-o-b--RS-  li VOLa02
   2     2015/07/02 18:31    2  89%  -il-o-b--Re-  li VOLa03
   ...
   51    2015/10/16 15:18    2  82%  -il-o-b-----  li VOLa80
   52    2015/10/16 15:25    2  84%  -il-o-b---D-  li VOLa81
```

5. If jobs are running, wait for them to complete.
6. Otherwise, once you are sure that no migrations are already underway, migrate volumes.

Migrate Volumes

The instructions in this section describe entering commands from the shell command prompt using the **samcmd** command. But please note that all commands can also be

entered from the **samu** interface as well, in the form **:***command*, where *command* is the name of the command.

1. If you are not currently logged in to the Oracle HSM metadata server as **root**, log in now.

   ```
   root@solaris:~#
   ```

2. Make sure that the source file system is mounted.

3. Activate the **migrationd.cmd** file. Use the command **samcmd migconfig**.

   If configuration is successful, the command displays the message **Configuring migration** and refers you to the log file for details:

   ```
   root@solaris:~# samcmd migconfig
   samcmd: migconfig: Configuring migration
   (see /var/opt/SUNWsamfs/sammig/logfile)
   root@solaris:~#
   ```

   Otherwise, the command stops with an error. Either you have failed to stop the migration process before issuing the configuration command, or you have stopped migration while a tape volume is still waiting to migrate:

   ```
   root@solaris:~# samcmd migconfig
   samcmd: migconfig: Can't configure migration, migration status is not stop, or
   migration job is pending
   root@solaris:~#
   ```

4. Display the migration configuration. Use the command **samcmd y**.

   If configuration was successful, the specified volumes are listed, the source volume's status is **sched_wait** (*scheduled*, *waiting*), and the destination volume's status is **avail** (*available*). In the example, configuration has succeeded:

   ```
   root@solaris:~# samcmd y
   Migration vsn list   samcmd  version HH:MM:SS month day year
   samcmd on hsm61sol
   Status: Stop: Waiting for :migstart  Vsns:2 src:1 dest:1 maxcopy:2
   ord m ty vsn    start time  end time    status  Inodes done/tot      bytes
     0 S li VOL001 none        none        sched_wait        0/0            0
     0 D li VOL501 none        none        avail                            0
   ```

5. If configuration was successful, start migration. Use the command **samcmd migstart**.

   ```
   root@solaris:~# samcmd migstart
   samcmd: migstart: State changed to start
   root@solaris:~#
   ```

6. Check the status of migration. Use the commands **samcmd x** and **samcmd y**.

   In the example, migration is just starting. the **Migration status** screen shows that the job status is now **Run**, **1** copy is underway using **s** (*server*) copy mode (**cmod**), the copy is **0%** complete, **0** inodes have been updated, and the source volume is still **Loading**:

   ```
   root@solaris:~# samcmd x
   Migration status    samcmd  version HH:MM:SS month day year
   Source Vsns - wait:  0 fsscan: 0 copy: 1 update ino: 0 log: 0 done:  0
   Status: Run
   source    dest      cmod perc status
   li VOL001 li VOL501 s    0%   Loading li.VOL001
   ```

The **Migration vsn list** screen shows that **2** volumes are currently being processed, **1** source and **1** destination. The status of both volumes is now **copy** to show that the source is being copied to the destination. At this point, **0** bytes have been copied from the source to the destination, and none of the **35023** inodes have been updated:

```
root@solaris:~# samcmd y
Migration vsn list   samcmd  version HH:MM:SS month day year
Status: Run  Vsns:2 src:1 dest:1 maxcopy:2
ord m ty vsn     start time  end time  status  Inodes done/tot    bytes
  0 S li VOL001  10/16 12:12 none      copy            0/35023        0
  0 D li VOL501  10/16 12:12 none      copy                           0
```

7. If you need to monitor I/O performance when using the StorageTek Direct Copy migration method, do so from the Storage Area Network switch that connects the drives.

   StorageTek Direct Copy bypasses the server host and operating system. So familiar tools like **iostat** cannot monitor the I/O.

8. Recheck the status of migration periodically, again using the commands **samcmd x** and **samcmd y**.

   In the example, the **Migration status** screen shows that the copy is now **23%** complete, and 560 (**0x00000230**) tape blocks have been read from the source:

```
root@solaris:~# samcmd x
Migration status    samcmd  version HH:MM:SS month day year
Source Vsns - wait:  0 fsscan: 0 copy: 1 update ino: 0 log: 0 done:  0
Status: Run
source    dest      cmod perc status
li VOL001 li VOL501 s     24% 0x00000230 blocks read
```

   The **Migration vsn list** screen shows that **164.50** megabytes have been read from the source volume and **148.75** megabytes have been written to the destination volume:

```
root@solaris:~# samcmd y
Migration vsn list   samcmd  version HH:MM:SS month day year
Status:  Run  Vsns:2 src:1 dest:1 maxcopy:2
ord m ty vsn     start time  end time  status  Inodes done/tot    bytes
  0 S li VOL001  10/16 12:12 none      copy            0/35023   164.50M
  0 D li VOL501  10/16 12:12 none      copy                      148.75M
```

9. When the migration completes, check the ending status. Use the commands **samcmd x** and **samcmd y** and examine the migration log file:

   In the example, source and destination volumes are no longer listed on the **Migration status** screen, which now shows that **1** copy is now done. Note that migration status is still **Run** and will remain so until we enter the **migidle** or **migstop** commands:

```
root@solaris:~# samcmd x
Migration status    samcmd  version HH:MM:SS month day year
Source Vsns - wait:  0 fsscan: 0 copy: 0 update ino: 0 log: 0 done:  1
Status: Run
source  dest  cmod perc status
```

   The **Migration vsn list** screen shows that **550.00** megabytes have been read from the source volume and **550.50** megabytes have been written to the destination

volume. All **35023** inodes have been updated to reflect the new locations of the migrated archive copies:

```
root@solaris:~# samcmd y
Migration vsn list   samcmd  version HH:MM:SS month day year
Status:  Run  Vsns:2 src:1 dest:1 maxcopy:2
ord m ty vsn     start time  end   time  status  Inodes done/tot    bytes
  0 S li VOL001  10/16 12:12 10/16 12:14 complete    35023/35023   550.00M
  0 D li VOL012  10/16 12:12 10/16 12:14 avail                     550.00M
```

The migration daemon's log file lists each stage of the migration and concludes with a summary. In the example, we use the Solaris **tail** command to view the most recent entries:

```
root@solaris:~# tail /var/opt/SUNWsamfs/sammig/logfile
date time Info: Schedule: Create VsnList file.
date time Info: Schedule: VsnList file created, source: 1, destination: 1.
date time Info: Schedule: Migration status changed to Start.
date time Info: 'li.VOL001' Filesystem scan: Started
date time Info: 'li.VOL001' Filesystem scan: Completed, total copy bytes:
517.2M, inodes: 35023, multi vsn copy: 0, removable-media file: 0, obsolete
copy: 0
date time Info: 'li.VOL001' Copy: Started, pid: 2459 destination 'li.VOL012'
date time Info: 'li.VOL001' Copy: Mode - server copy
date time Info: 'li.VOL001' Copy: Server copy started from position 0x4.
date time Info: 'li.VOL001' Copy: Tar header check started from position 0x4.
date time Info: 'li.VOL001' Copy: Tar header check succeeded, 5 inodes checked,
0 tar header error found.
date time Info: 'li.VOL001' Copy: Completed, pid: 2459, exit status: 12,
signal: 0
date time Info: 'li.VOL001' Update inode: Started, source position: 0
date time Info: 'li.VOL001' Update inode: Completed.
date time Info: 'li.VOL001' Log: Started, source position: 0
date time Info: 'li.VOL001' Log: Completed.
date time Summ: 'li.VOL001'
date time Summ: 'li.VOL001' ============== Summary ==============
date time Summ: 'li.VOL001' Status:    Complete
date time Summ: 'li.VOL001' Copy mode: Server copy
date time Summ: 'li.VOL001' Start at:  date time
date time Summ: 'li.VOL001' End at:    date time
date time Summ: 'li.VOL001' Bytes:     550.00M
date time Summ: 'li.VOL001' Archive copies:                  35023
date time Summ: 'li.VOL001' Read error copies:                   0
date time Summ: 'li.VOL001' Multi vsn copies:                    0
date time Summ: 'li.VOL001' Removable-Media file:                0
date time Summ: 'li.VOL001' ---Dest---   ---Bytes---   ---Copies---
date time Summ: 'li.VOL001' li VOL501        550.00M          35023
root@solaris:~#
```

10. Finally, make sure that you copy the volume-migration logs to a secure location.

    These logs record the destination volume and starting position for each archive file copy that was migrated off of each source volume. This information is critical when you need to recover files or file systems. So Oracle strongly recommends that you keep backup copies of these files with your recovery point and archiver log files, as described in Chapter 7, "Backing Up the Configuration and File Systems" and in the corresponding chapter of the *Oracle Hierarchical Storage Manager and StorageTek QFS Installation and Configuration Guide*.

The migration daemon creates migration log files in the directory that you specified in the **migrationd.cmd** file. For each volume migrated, it creates a file named *media_type.VSN* where:

- *media_type* is a two-letter code that identifies the kind of source media (see Appendix A, "Glossary of Equipment Types" for details).
- *VSN* is the unique volume serial number that identifies the source volume.

In the example, we copy the volume logs from specified log directory, **/var/adm/hsm_migration_logs/**, to the directory where we keep the file-system recovery resources on an NFS-mounted remote file system:

```
root@solaris:~# ls /var/adm/hsm_migration_logs/
li.VOL001  li.VOL002  li.VOL003  li.VOL004  li.VOL005  li.VOL006 ... ti.801 ...
root@solaris:~# cp /var/adm/hsm_migration_logs/*.* /zfs/recover/hsmfs1/2015mig/
```

11. If you have moved data off of media without creating an additional archive copy, dispose of the old media according to your requirements.

    See "Disposing of Old Media After Migration" on page 8-20.

12. Stop here. Migration is complete.

Staging Files and Rearchiving to Replacement Media

To migrate archive files from old to new media using the staging and rearchiving method, you need to identify the files that will migrate, stage them to the disk cache, and then write them to the new media, without interfering with normal file-system operations. This chapter covers the following stages in the process:

- estimating available resources
- configuring an archiving process to use the new media
- migrate the data to the replacement media.

Estimate The Resources Available for Staging and Re-Archiving

The details of the staging and rearchiving process depend largely on two factors: the amount of disk storage available and the number of removable media drives available. During media migration, the Oracle HSM stager loads the old removable volumes into drives that can read the old media format and restores archived files to the disk cache. Then the Oracle HSM archiver rearchives the files to new removable volumes using drives that can write the new media format. So, ideally, you would stage all the files on any given tape volume to disk at once and then immediately archive them to new media.

To do this, you would have to dedicate significant resources for the duration of the migration:

- disk space equivalent to the capacity of a full tape
- exclusive use of a drive that reads the old tape format
- exclusive use of a drive that writes the new format.

The above is not a problem if you can quiesce the file system until migration is complete. But migrating data in a production setting, without unduly interfering with ongoing file system and archive operations, requires some thought. If disk space or tape drives are in short supply, you need to identify the resources that you can

reasonably spare for migration and then adjust the migration process. So proceed as follows:

1. Estimate the amount of disk cache that you can use for migration without impeding normal file-system operations.
2. Estimate the number of tape drives that you can afford to dedicate to migration.

   If only a limited number of tape drives are available, plan on throttling the staging and archiving processes so that the migration process does not impede normal operations.
3. Based on your estimates above, decide on staging and archiving parameters. Determine the maximum number of migrating files that the available disk space will hold at any one time and the maximum rate at which files can be moved out of cache and on to new media.
4. Once you have estimated the resources, plan for the post-migration disposition of the old media.

Configure an Archiving Process to Use the New Media

Add the new media to the **archiver.cmd** file and modify the archive copy directives so that one copy is always made using the new media.

1. Open the **/etc/opt/SUNWsamfs/archiver.cmd** file in a text editor.

   The archiving policy specifies two copies, both of which are written to the media type that we want to replace. In the example, we open the file in the **vi** editor. We want to replace the DLT cartridges (type **lt**):

```
root@solaris: vi /etc/opt/SUNWsamfs/archiver.cmd
# ============================================
# /etc/opt/SUNWsamfs/archiver.cmd
# --------------------------------------------
...
# --------------------------------------------
# VSN Directives
vsns
allfiles.1 lt .*
allfiles.2 lt .*
endvsns
```

2. In the directives for copy **2**, change the specified media type to the identifier for the new media, save the file, and close the text editor.

   In the example, we want to migrate data from the old DLT tapes to new LTO cartridges. So, in copy **2**, we change the old media type, **lt** (DLT), to **li** (LTO):

```
root@solaris: vi /etc/opt/SUNWsamfs/archiver.cmd
# ============================================
# /etc/opt/SUNWsamfs/archiver.cmd
# --------------------------------------------
...
# --------------------------------------------
# VSN Directives
vsns
allfiles.1 lt .*
allfiles.2 li .*
endvsns
:wq
root@solaris:~#
```

3. Check the **archiver.cmd** file for syntax errors. Run the command **archiver -lv**, and correct errors until no errors are found.

   The **archiver -lv** command will print out the file line-by-line. If it encounters an error, it will stop running at the point where the error occurred.

```
root@solaris:~# archiver -lv
Reading '/etc/opt/SUNWsamfs/archiver.cmd'.
1: # ========================================
2: # /etc/opt/SUNWsamfs/archiver.cmd
3: # ----------------------------------------
4: # Global Directives
5: logfile = /var/opt/SUNWsamfs/archiver.log
6: # ----------------------------------------
7: # File System Directives:
8: fs = samqfsms
9: all .
10: 1 5m ...
root@solaris:~#
```

4. Once the modified **archiver.cmd** file is error-free, load it into the current configuration using the command **samd config**:

```
root@solaris:~# samd config
Configuring SAM-FS
root@solaris:~#
```

5. Next, migrate data from cartridge to cartridge.

Migrate the Data to the Replacement Media

The staging and archiving method for migrating data uses **sfind**, the Oracle HSM extension of the GNU **find** command. The **sfind** command is used to locate files on a specified tape volume and launch the **stage** and **rearchive** commands against all files found.

If you are unfamiliar with the **sfind**, **stage**, and/or **rearchive** commands, you should review their respective man pages now. Then, for each tape cartridge that holds data that must be migrated, proceed as follows:

Migrate Data from One Cartridge to Another

1. Log in to the file-system host as **root**.

```
root@solaris:~#
```

2. Move to the mount-point directory of the file system that holds the files that you are migrating.

   In the example, we are migrating archived copies of files stored in the **hsmfs1** file system mounted at **/hsm/hsmfs1**:

```
root@solaris:~# cd /hsm/hsmfs1
root@solaris:~#
```

3. Select a tape volume.

   When migrating data from media type to media type, work with one volume at a time. In the examples below, we work with volume serial number **VOL008**.

4. First, search the selected volume for damaged files that cannot be successfully staged. Use the Oracle HSM command **sfind . -vsn** *volume-serial-number* **-damaged**, where *volume-serial-number* is the alphanumeric string that uniquely identifies the volume in the library.

   In the example, we start the search from the current working directory (**.**). The **-vsn** parameter limits the search to files that are found on our current tape, **VOL008**. The **-damaged** flag limits the search to files that cannot be successfully staged:

   ```
   root@solaris:~# sfind . -vsn VOL008 -damaged
   ```

5. If the **sfind** search for damaged files returns any results, try to fix the file. Use the command **undamage -m** *media-type* **-vsn** *volume-serial-number file*, where:

   - *media-type* is one of the two-character media type codes listed in Appendix A.
   - *volume-serial-number* is the alphanumeric string that uniquely identifies the volume.
   - *file* is the path and name of the damaged file.

   Sometimes a transitory I/O error causes a copy to be marked damaged. The Oracle HSM **undamage** command clears this condition. In the example, the archive file copy **/hsm/hsmfs1/data0008/20131025DAT** is reported damaged. So we undamage it, and retry the search for damaged files:

   ```
   root@solaris:~# sfind . -vsn VOL008 -damaged
   /hsm/hsmfs1/data0008/20131025DAT
   root@solaris:~# undamage -m lt -vsn VOL008 /hsm/hsmfs1/data0008/20131025DAT
   root@solaris:~# sfind . -vsn VOL008 -damaged
   ```

6. If the **sfind** command again lists the file as damaged, the copy is unusable. See if the archive contains another, undamaged copy of the file. To list the available copies, use the command **sls -D** *file*, where *file* is the path and name of the file. To check the status of any copies found, use the command **sfind** *file* **-vsn** *volume-serial-number*.

   In the example, the **undamage** command could not fix the copy. So we use **sls** to list all copies of the file **/hsm/hsmfs1/data0008/20131025DAT**:

   ```
   root@solaris:~# undamage -m lt -vsn VOL008 /hsm/hsmfs1/data0008/20131025DAT
   root@solaris:~# sfind . -vsn VOL008 -damaged
   /hsm/hsmfs1/data0008/20131025DAT
   root@solaris:~# sls -D /hsm/hsmfs1/data0008/20131025DAT
   20131025DAT:
   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 07:12     1e4b1.1    lt VOL008
              copy 2: ---- May 21 10:29     109c6.1    lt VOL022
   ...
   ```

   Tape volume **VOL022** holds a second copy of the file. So we check the second copy with **sfind**:

   ```
   root@solaris:~# sfind /hsm/hsmfs1/data0008/20131025DAT -vsn VOL022 -damaged
   ```

7. If a copy is unusable and one undamaged copy of the file exists, rearchive the file. Then, once the archive holds two good copies, unarchive the damaged copy.

In the example, copy 1 of file **/hsm/hsmfs1/data0008/20131025DAT** on volume **VOL008** is unusable, but the **sfind** command did not find damage to copy 2. So we issue the **archive** command with the **-c** option to create a valid copy 1 before unarchiving the damaged copy on volume **VOL008**:

```
root@solaris:~# sfind /hsm/hsmfs1/data0008/20131025DAT -vsn VOL022 -damaged
root@solaris:~# archive -c 1 /hsm/hsmfs1/data0008/20131025DAT
...
root@solaris:~# unarchive -m lt -vsn VOL008 /hsm/hsmfs1/data0008/20131025DAT
```

8. If no usable copies exist, see if the file is resident in cache. Use the command **sfind . -vsn** *volume-serial-number* **-online**.

   In the example, both copy 1 on volume **VOL008** and copy 2 on volume **VOL022** are damaged and unusable. So we see if the file is available online, in the disk cache:

```
root@solaris:~# undamage -m lt -vsn VOL008 /hsm/hsmfs1/data0008/20131025DAT
root@solaris:~# sfind . -vsn VOL008 -damaged
/hsm/hsmfs1/data0008/20131025DAT
root@solaris:~# undamage -m lt -vsn VOL022 /hsm/hsmfs1/data0008/20131025DAT
root@solaris:~# sfind /hsm/hsmfs1/data0008/20131025DAT -vsn VOL022 -damaged
/hsm/hsmfs1/data0008/20131025DAT
root@solaris:~# sfind /hsm/hsmfs1/data0008/20131025DAT -online
```

9. If no usable copies exist, but the file is resident in cache, archive the file. Then, once the archive holds two good copies, unarchive the damaged copy.

   In the example, both copy 1 on volume **VOL008** and copy 2 on volume **VOL022** are unusable, so we issue the **archive** command to create two valid copies before unarchiving the damaged copy on volume **VOL008**:

```
root@solaris:~# undamage -m lt -vsn VOL008 /hsm/hsmfs1/data0008/20131025DAT
root@solaris:~# sfind . -vsn VOL008 -damaged
/hsm/hsmfs1/data0008/20131025DAT
root@solaris:~# undamage -m lt -vsn VOL022 /hsm/hsmfs1/data0008/20131025DAT
root@solaris:~# sfind /hsm/hsmfs1/data0008/20131025DAT -vsn VOL022 -damaged
/hsm/hsmfs1/data0008/20131025DAT
root@solaris:~# sfind /hsm/hsmfs1/data0008/20131025DAT -online
/hsm/hsmfs1/data0008/20131025DAT
root@solaris:~# archive /hsm/hsmfs1/data0008/20131025DAT
root@solaris:~# unarchive -m lt -vsn VOL008 /hsm/hsmfs1/data0008/20131025DAT
```

10. If no usable copies exist and if the file is not resident in the disk cache, the data has probably been lost. If the data is critical, consult a specialist data recovery firm for assistance. Otherwise, unarchive the damaged copy.

    In the example, both copy 1 on volume **VOL008** and copy 2 on volume **VOL022** are unusable. The **sfind** command could not find the file in the disk cache. The data is not critical. So we unarchive the damaged copy on volume **VOL008**:

```
root@solaris:~# undamage -m lt -vsn VOL008 /hsm/hsmfs1/data0008/20131025DAT
root@solaris:~# sfind . -vsn VOL008 -damaged
/hsm/hsmfs1/data0008/20131025DAT
root@solaris:~# undamage -m lt -vsn VOL022 /hsm/hsmfs1/data0008/20131025DAT
root@solaris:~# sfind /hsm/hsmfs1/data0008/20131025DAT -vsn VOL022 -damaged
/hsm/hsmfs1/data0008/20131025DAT
root@solaris:~# sfind /hsm/hsmfs1/data0008/20131025DAT -online
root@solaris:~# archive /hsm/hsmfs1/data0008/20131025DAT
root@solaris:~# unarchive -m lt -vsn VOL008 /hsm/hsmfs1/data0008/20131025DAT
```

11. When the **sfind** search for damaged files returns no results, stage the files from the current tape to the disk cache. Use the command **sfind . -vsn** *volume-serial-number* **-offline -exec stage {}\;**

    The **-vsn** parameter limits the search to files that are found on the current tape (always migrate data one tape at a time.).

    The **-offline** parameter further restricts the **sfind** output to files that are not already resident in cache, so that data is not overwritten.

    The **-exec stage {}\;** argument takes each path and file name that **sfind** returns and uses it as the argument to an Oracle HSM **stage** command. The **stage** command then restores the specified file to disk cache. The process repeats until all eligible files have been staged.

    In the example, the **sfind -vsn VOL008 -damaged** command returns no output. So we use **sfind** to stage all files that are found on **VOL008** and are not already in cache:

    ```
    root@solaris:~# sfind . -vsn VOL008 -damaged
    root@solaris:~# sfind . -vsn VOL008 -offline -exec stage {}\;
    ```

12. Once files have been staged from tape, selectively rearchive them. Use the command **sfind . -vsn** *volume-serial-number* **-online -exec rearch -r -m** *media-type* **{}\;** where *media-type* is the type of media from which you are migrating.

    The **-vsn** parameter limits the search to files that are also found on the current tape (always migrate data one tape at a time.).

    The **-online** parameter further restricts the **sfind** output to files that are resident in cache, so that data is not overwritten.

    The **-exec rearch -r -m** *media-type* **{}\;** argument takes each path and file name that **sfind** returns and uses it as the argument to an Oracle HSM **rearch -r -m** *media-type* command. The **-r** argument runs the process recursively through subdirectories. The **-m** argument rearchives only files that reside on the source media.

    In the example, the **-vsn** parameter value is **VOL008**, and the value of the **-m** parameter specifies **lt**, for DLT media:

    ```
    root@solaris:~# sfind . -vsn VOL008 -online -exec rearch -r -m lt {}\;
    ```

13. Repeat the preceding step until the **sfind** search finds no more files.
14. When all files have been rearchived, dispose of the old tape as planned.
15. Repeat this procedure until data has been migrated from all old media to new media.

Disposing of Old Media After Migration

Once you complete a migration, the old media do not necessarily lose all value. So carefully consider how you should dispose of them.

- At a minimum, retain the old media until you have accumulated enough new recovery point files to recover any file in the file system using only the new, replacement media.

- If storage space allows, keep the old media indefinitely. As long as compatible drives remain available, the old media can be an invaluable backup and recovery resource.
- If library space is at a premium, export the old media and retain them in off-site storage.
- If the old media can be reused and if you are sure that the data they contain are no longer useful, relabel the old volumes. For example, you could relabel the media for an older Oracle StorageTek T10000C drive and use it with a newer T10000D drive.
- Otherwise, if neither the data on the old volumes nor the media have any remaining value, export the volumes from the library and dispose of them appropriately.

A

Glossary of Equipment Types

The value of the **Equipment Type** field of the Master Configuration File (**mcf**) identifies devices and device configurations within the Oracle Hierarchical Storage Manager and StorageTek QFS Software. Equipment types are specified as two-character codes. This glossary lists the codes for quick reference when working with the samples or when interpreting an existing **mcf** (for full details see the **mcf(4)** man page).

For convenience, the codes are divided into two sections and then listed alphabetically:

Recommended Equipment and Media Types

This section describes all of the equipment codes that you normally need: the generic equipment codes (**rb**, **tp**, and **od**) and codes for identifying network-attached library interfaces, archival disk volumes, cloud resources, and the Oracle HSM historian.

The generic equipment codes **rb**, **tp**, and **od** are the preferred equipment type codes for all SCSI-attached libraries, tape drives, and optical disk devices. When you specify a generic equipment type, Oracle HSM can automatically set the correct type based on SCSI vendor codes.

cl
Cloud media, the abstract media type that organizes cloud resources into logical media volumes suitable for archiving. Oracle HSM labels each volume with a 31-character volume serial number (VSN) of the form *nameNumber*, where:

- *name* is the customer-defined string of 4 to 20 alphanumeric characters that identifies the logical library (type **cr**) that owns the media.
- *Number* is a randomly generated number in the range **[0000000-9999999]**.

For additional information, see the **cloud** (7) and **sam-cloudd** (1m) man pages.

cr
A cloud library, the abstract equipment type that manages cloud media (volumes of type **cl**) by emulating a network-attached tape library with a configurable number of drives.

A parameter file defines the characteristics of the equipment, including the **name** parameter value that prefixes and uniquely identifies the type **cl** volumes that belong to the library. See the **cloud** (7) and **sam-cloudd** (1m) man pages.

For additional information, see the **cloud** (7) and **sam-cloudd** (1m) man pages.

dk
A disk-based file system that the Oracle HSM software uses as an archival volume. UFS, ZFS, QFS, and NFS file systems can serve as disk archives.

Disk volumes and volume serial numbers (VSNs) are defined in the **/etc/opt/SUNWsamfs/diskvols.conf** file. See the **diskvols.conf** (4) man page.

g*XXX*
Where *XXX* is an integer in the range **[0-127]**, a striped group of disk devices that is part of an **ma** disk-cache family set.

hy
The Oracle HSM historian, an optional, virtual library that maintains a media catalog, but has no associated hardware. Used for tracking exported media.

ma
A high-performance QFS file system that maintains file-system metadata on one or more dedicated **mm** disk devices. File data resides on separate **md**, **mr**, or **g***XXX* data devices.

md
A disk device that stores file data for an **ma** file system or data and metadata for an **ms** file system. **md** devices store file data in small, 4-kilobyte Disk Allocation Units (DAUs) and large, 16-, 32-, or 64-kilobyte DAUs. The default DAU is 64-kilobytes.

mm
A disk device that stores file-system metadata for a high-performance **ma** file system.

mr
A disk device that stores file data for an **ma** file system. **mr** devices store file data in large, fully adjustable Disk Allocation Units (DAUs) that are multiples of 8 kilobytes in the range 8-65528 kilobytes. The default DAU is 64 kilobytes.

ms
A Oracle HSM file system that maintains file-system metadata on the same devices that store file data.

od
Any SCSI-attached optical disk. Oracle HSM sets the appropriate equipment type automatically using the SCSI vendor code.

of
An abstract media type that distinguishes foreign media from Oracle HSM file system media. Used when migrating files from a foreign file system to an Oracle HSM file system.

The **of** media type does not identify a physical equipment type. Never specify it in a master configuration file (**mcf**).

rb
Any SCSI-attached tape library. Oracle HSM sets the appropriate equipment type automatically using the SCSI vendor code.

rd
The SAM-Remote pseudo-device. In the Master Configuration File (**mcf**), the corresponding **Equipment Identifier** field has to contain the path to the pseudo-device (such as **/dev/samrd/rd2**). The corresponding **Family Set** field has to contain the hostname of the SAM-Remote server.

sc
A SAM-Remote client system. In the Master Configuration File (**mcf**), the corresponding **Equipment Identifier** field has to contain the path the SAM-Remote client-configuration file for the client. The corresponding **Family Set** field has to contain the family set name of the server. The **Additional Parameters** field must contain the full path to the client's library catalog file.

sk
An Oracle StorageTek ACSLS interface to a network-attached library. In the Master Configuration File (**mcf**), the corresponding **Equipment Identifier** field has to contain the path to the parameters file for the ACSLS interface. For more information, see the **stk(7)** man page.

ss
A SAM-Remote server. In the Master Configuration File (**mcf**), the corresponding **Equipment Identifier** field has to contain the path to the SAM-Remote server-configuration file. The corresponding **Family Set** field has to contain the family set name of the server, which must match the name used in the **Family Set** field of the **mcf** on the client.

tf
An abstract media type that distinguishes Oracle StorageTek T10000 and LTO volumes that are in Linear Tape File System (LTFS) format from volumes that contain Oracle HSM archive files.

The **tf** media type does not identify a physical equipment type. Never specify it in a master configuration file (**mcf**).

tp
Any SCSI-attached tape drive. Oracle HSM sets the appropriate equipment type automatically using the SCSI vendor code. No, however, that if you do use more specific equipment codes such as **li** and **ti**, you must do so consistently. If you specify **li** (LTO) tape equipment in the **mcf** file, for example, you cannot refer to the same equipment as **tp** equipment in the **archiver.cmd** file

z*X* (where *X* is a character in the range [0-9a-z])
An abstract media type that distinguishes foreign media from media controlled by Oracle HSM software.

Other Equipment and Media Types

The equipment types listed in this section are also supported.

Note that, in most cases, Oracle recommends identifying SCSI-attached libraries, tape drives, and optical disk devices using the generic equipment types **rb**, **tp**, and **od**. The generic equipment types tell Oracle HSM to identify the hardware dynamically, using SCSI vendor IDs. The type codes below are essential when migrating from one media type to another and may sometimes be useful for management purposes. But using them in a Master Configuration File (**mcf**), for example, hard-codes a static equipment configuration that may, at some point, no longer match the actual hardware.

ac
A Sun 1800, 3500, or L11000 tape library.

at
A Sony AIT-4 or AIT-5 tape drive.

cy
A Cygnet optical disk library.

d3
A StorageTek D3 tape drive.

dm
A Sony DMF library.

ds
A DocuStore or Plasmon optical disk library.

dt
A DAT 4-mm tape drive.

e8
An Exabyte X80 library.

fd
A Fujitsu M8100 128-track tape drive.

h4
An HP SL48 or SL24 library.

hc
An Hewlett Packard L9-/L20-/L60-series library.

i7
An IBM 3570 tape drive.

ic
An IBM 3570 media changer.

il
An IBM 3584 tape library.

li
An LTO-3 or later tape drive.

lt
A Digital Linear Tape (DLT), Super DLT, or DLT-S4 tape drive.

me
A Metrum library.

mf
An IBM Multi Function optical drive.

mo
A 5.25-in erasable optical drive.

o2
A 12-in WORM drive.

ov
An Overland Data Inc. Neo Series tape library.

pd
A Plasmon D-Series DVD-RAM library.

q8
A Qualstar 42xx, 62xx, or 82xx library.

s3
A StorageTek SL3000 library.

s9
An Oracle StorageTek 97xx series library.

se
A StorageTek 9490 tape drive.

sf
A StorageTek T9940 tape drive.

sg
A StorageTek 9840C or later tape drive.

sl
A Spectra Logic or Qualstar tape library.

st
A StorageTek 3480 tape drive.

ti
A StorageTek T10000 (Titanium) tape drive.

vt
A Metrum VHS (RSP-2150) tape drive.

wo
A 5.25-in optical WORM drive.

xt
An Exabyte (850x) 8-mm tape drive.

B

Media Status Flags

Media flags have the following meanings:

- **A** means that the slot needs an audit.
- **C** means that the slot contains cleaning cartridge.
- **D** means that the volume is a media migration destination.
- **E** means that the volume is bad or the cleaning media has expired.
- **L** means that the volume is a Linear Tape File System (LTFS) volume.
- **N** means that the volume is not in Oracle HSM format.
- **R** means that the volume is read-only (a software flag).
- **S** means that the volume is media migration source.
- **U** means that the volume is unavailable.
- **W** means that the volume is physically write-protected.
- **X** means that the slot is an export slot.
- **b** means that the volume has a bar code.
- **c** means that the volume is scheduled for recycling
- **f** means that the archiver found the volume full or corrupt.
- **d** means that the volume has a duplicate volume serial number (VSN)
- **l** means that the volume is labeled.
- **o** means that the slot is occupied.
- **p** means that the volume is a high priority volume.
- **-**, when used in displays, means that the corresponding flag is not set.

For example, the **samcmd v** lists catalog information, including media flags for each cataloged volume:

```
root@solaris:~# samcmd v 800
Robot catalog samcmd     6.1    16:45:25 Feb 14 2016
samcmd on samqfshost                          count 32
Robot VSN catalog by slot       : eq 800
slot          access time count use  flags         ty vsn
   0     2014/03/14 11:23  875   0%  -il-o-b-----  li VOL001
   1     2014/03/13 17:54  866   0%  -il-o-b-----  li VOL002
   2     2014/03/14 11:26    3   0%  -il-o-b-----  li VOL003
   3     2014/03/14 10:33    3   0%  -il-o-b-----  li VOL004
   4     2014/03/14 11:34    5   0%  -il-o-b-----  li VOL005
```

```
  5     2014/03/14 11:32    2   0%  -ilEo-b----f  li VOL006 MEDIA ERROR
  6     2014/03/13 18:07    2   0%  -il-o-b-----  li VOL007
  7     2014/03/13 18:07    1   0%  -il-o-b-----  li VOL008
 10     2014/03/13 18:10    2   0%  -il-o-b-----  li VOL011
```

C

Mount Options in a Shared File System

An Oracle Hierarchical Storage Manager and StorageTek QFS Software shared file system can be mounted with several mount options. This chapter describes some of these options within the context of their roles.

Shared File System Mount Options

You can specify most mount options by using the **mount** command, by entering them in the **/etc/vfstab** file, or by entering them in the **samfs.cmd** file. For example, the following **/etc/vfstab** file includes mount options for a shared file system:

```
sharefs - /sfs samfs - no shared,mh_write
```

You can change some mount options dynamically by using the **samu** operator utility. For more information about these options, see the *Oracle Hierarchical Storage Manager and StorageTek QFS samu Command Reference*.

For more information about any of these mount options, see the **mount_samfs** (1m) man page.

bg: Mounting in the Background

The **bg** mount option specifies that if the first mount operation fails, subsequent attempts at mounting should occur in the background. By default, **bg** is not in effect, and mount attempts continue in the foreground.

retry: Reattempting a File System Mount

The **retry** mount option specifies the number of times that the system should attempt to mount a file system. The default is 10000.

shared: Declaring an Oracle HSM Shared File System

The **shared** mount option declares a file system to be an Oracle HSM shared file system. This option must be specified in the **/etc/vfstab** file in order for the file system to be mounted as an Oracle HSM shared file system. The presence of this option in a **samfs.cmd** file or on the **mount** command does not cause an error condition, but it does not mount the file system as a shared file system.

minallocsz and maxallocsz: Tuning Allocation Sizes

The **minallocsz** and **maxallocsz** options to the **mount** command specify an amount of space, in kilobytes. These options set the minimum block allocation size. If a file is growing, the metadata server allocates blocks when an append lease is granted. Use

-o minallocsz=*n* to specify the initial size of this allocation. The metadata server can increase the size of the block allocation depending on the application's access patterns up to but not exceeding the **-o maxallocsz=***n* setting.

You can specify these **mount** options on the **mount** command line, in the **/etc/vfstab** file, or in the **samfs.cmd** file.

rdlease, wrlease, and aplease: Using Leases in an Oracle HSM Shared File System

By default, when hosts share files, the Oracle HSM metadata server maintains file-system consistency by issuing I/O *leases* to itself and its clients. A lease grants a shared host permission to perform an operation on a file for a specified period. A *read lease* lets a host read file data. A *write lease* lets a host overwrite existing file data. An *append lease* lets a host write additional data at the end of a file. The metadata server can renew leases as necessary.

Reads and writes to a Oracle HSM shared file system should thus provide near-POSIX behavior for data. For metadata, however, access time changes might not be seen immediately on other hosts. Changes to a file are pushed to disk at the end of a write lease. When a read lease is acquired, the system invalidates any stale cache pages so that the newly written data can be seen.

The following mount options set the duration of the leases:

- **-o rdlease=** *number-seconds* specifies the maximum amount of time, in seconds, for the read lease.
- **-o wrlease=** *number-seconds* specifies the maximum amount of time, in seconds, for the write lease.
- **-o aplease=** *number-seconds* specifies the maximum amount of time, in seconds, for the append lease.

In all three cases, *number-seconds* is an integer in the range [**15-600**]. The default time for each lease is **30** seconds. A file cannot be truncated if a lease is in effect. For more information about setting these leases, see the **mount_samfs** man page.

If you change the metadata server because the current metadata server is down, you must add the lease time to the changeover time because all leases must expire before an alternate metadata server can assume control.

Setting a short lease time causes more traffic between the client hosts and the metadata server, because the lease must be renewed after it has expired.

mh_write: Enabling Multiple Host Reads and Writes

The **mh_write** option controls write access to the same file from multiple hosts. If **mh_write** is specified as a mount option on the metadata server host, the Oracle HSM shared file system enables simultaneous reads and writes to the same file from multiple hosts. If **mh_write** is not specified on the metadata server host, only one host can write to a file at any one time.

By default, **mh_write** is disabled, and only one host has write access to a file for the duration of the **wrlease** mount option. If the Oracle HSM shared file system is mounted on the metadata server with the **mh_write** option enabled, simultaneous reads and writes to the same file can occur from multiple hosts.

When **mh_write** is enabled on the metadata server, Oracle HSM supports the following:

- Multiple reader hosts and paged I/O

- Multiple reader and/or writer hosts and direct I/O only if there are writers
- One append host (other hosts read or write) and direct I/O only if there are writers.

Mounting a file system with the **mh_write** option does not change locking behavior. File locks behave the same regardless of whether **mh_write** is in effect. But, in other respects, behavior might be less consistent. When there are simultaneous readers and writers, the Oracle HSM shared file system uses direct I/O for all host access to a file. Therefore, page-aligned I/O should be visible immediately to other hosts. However, non-page-aligned I/O can result in stale data being visible, or even written to the file, because the normal lease mechanism preventing such occurrences has been disabled.

For this reason, you should specify the **mh_write** option only when multiple hosts need to write the same file simultaneously and when hosted applications perform page-aligned I/O and coordinate conflicting writes. In other cases, data inconsistency could occur. Using **flock()** with **mh_write** to coordinate between hosts does not guarantee consistency. For more information, see the **mount_samfs** man page.

min_pool: Setting the Minimum Number of Concurrent Threads

The **min_pool** mount option sets the minimum number of concurrent threads for the Oracle HSM shared file system. The default setting is **min_pool=64** on Oracle Solaris systems. This setting means that at least 64 active threads will be in the thread pool on Oracle Solaris. You can adjust the **min_pool** setting to any value in the range [**8-2048**], depending on shared file-system activity.

The **min_pool** mount option must be set in the **samfs.cmd** file. It will be ignored if set in the **/etc/vfstab** file or at the command line.

meta_timeo: Retaining Cached Attributes

The **meta_timeo** mount option determines how long the system waits between checks on the metadata information. By default, the system refreshes metadata information every three seconds. For example, an **ls** command entered in a shared file system with several newly created files might not return information about all the files until three seconds have passed. The syntax for the option is **meta_timeo=***seconds* where *seconds* is an integer in the range [**0-60**].

stripe: Specifying Striped Allocation

By default, data files in a shared file system are allocated using the round-robin file allocation method. To specify that file data be striped across disks, you can specify the **stripe** mount option on the metadata host and all potential metadata hosts. Note that by default, unshared file systems allocate file data using the striped method.

In a round-robin allocation, files are created in a round-robin fashion on each slice or striped group. The maximum performance for one file will be the speed of a slice or striped group. For more information about file allocation methods, see the *Oracle Hierarchical Storage Manager and StorageTek QFS Installation and Configuration Guide* (*Oracle HSM Customer Documentation Library*, **docs.oracle.com/en/storage**).

sync_meta: Specifying the Frequency With Which Metadata Is Written

You can set the **sync_meta** option to **sync_meta=1** or **sync_meta=0**.

The default setting is **sync_meta=1**, which means that a Oracle HSM shared file system writes file metadata to disk every time the metadata changes. This setting slows data

performance but ensures data consistency. This setting must be in effect if you want to change the metadata server.

If you set **sync_meta=0**, the Oracle HSM shared file system writes the metadata to a buffer before writing it to disk. This delayed write delivers higher performance but decreases data consistency after an unscheduled machine interruption.

worm_capable and def_retention: Enabling WORM Functionality

The **worm_capable** mount option lets the file system support WORM files. The **def_retention** mount option sets the default retention time using the format **def_retention=***M***y***N***d***O***h***P***m**.

In this format, *M*, *N*, *O*, and *P* are non-negative integers and **y**, **d**, **h**, and **m** stand for years, days, hours, and minutes, respectively. Any combination of these units can be used. For example, **1y5d4h3m** indicates 1 year, 5 days, 4 hours, and 3 minutes; **30d8h** indicates 30 days and 8 hours; and **300m** indicates 300 minutes. This format is backward compatible with the formula in previous software versions, in which the retention period was specified in minutes.

For more information, see the *Oracle Hierarchical Storage Manager and StorageTek QFS Installation and Configuration Guide* (*Oracle HSM Customer Documentation Library*, **docs.oracle.com/en/storage**).

D

Configuration Directives and Parameters

This appendix lists directives and parameters used in Oracle Hierarchical Storage Manager configuration files. Each directive is a single text line composed of one or more comma-delimited fields. Related directives are stored together in Oracle HSM command (**.cmd**) files.

The remainder of this appendix provides an overview the three main configuration files and their associated directives:

- archiving
- staging
- preview requests.

See the Oracle HSM man pages for additional information.

Note that you can configure Oracle HSM command files from the command line, as described here, or by using the Oracle HSM Manager software. For information on Oracle HSM Manager, see the online help.

Archiving

Archiving directives and parameters define the archive sets that control copying of files, the media used, and the overall behavior of the archiving software. They are grouped together in a configuration file, **/etc/opt/SUNWsamfs/archiver.cmd**.

There are three groups of archiving-related directives and parameters, each of which has its own section in the **archiver.cmd** file:

- Archiving directives configure archiving operations.
- Copy parameters configure the archiving operations for a specific copy of the archive set.
- Volume Serial Number (VSN) association directives assign media to each specified copy operation.

  Volume Serial Number (VSN) association directives are contained in their own section at the end of the **archiver.cmd** file, immediately following the copy parameters section.

When both a global directive and a more granular, more context-specific directive or parameter are in scope for the same file system or copy, the more granular rule overrides the more general.

If duplicate directives are entered, the value of the first instance that the archiver encounters overrides all subsequent values set for the same directive.

The `archiver.cmd` File

An **`archiver.cmd`** file consists of five sections:

- Global directives specify archiver behavior for all configured Oracle HSM file systems.
- Archive set definitions identify files that are archived as a group, by file system and starting directory, specify the number of copies, and control how long files are retained in the disk cache following archiving.
- Copy parameters specify how each specified copy is made, either by archive set or for all archive sets, using the special directive **`allsets`**.
- The optional volume serial number (VSN) pools section organizes archival media into sets of media that can be assigned to a copy operation by name, as a group. Pool members are defined with a space-delimited list of regular expressions that match volume labels.
- The VSN directives section assigns media to copy operations, using a space-delimited list of pool names and/or regular expressions that match volume labels.

```
# /etc/opt/SUNWsamfs/archiver.cmd
#------------------------------------------------------------------------
# Global Directives
archivemeta = off
examine = noscan
setarchdone = off
scanlist_squash = off
#------------------------------------------------------------------------
# Archive Set Definitions
fs = hsmqfs
logfile = /var/adm/hsmqfs.archive.log
datafiles .
    1 -norelease 15m
    2 -norelease 15m
    3 -norelease 30m
    4 -norelease 30m
fs = dskvolqfs
logfile = /var/adm/dskvolqfs.archive.log
no_archive .
#------------------------------------------------------------------------
# Copy Parameter Directives
params
allsets -sort path -offline_copy stageahead
allsets.1 -startage 6h  -startsize 6G  -startcount 500000
allsets.2 -startage 24h -startsize 20G -startcount 500000 -drives 5
allsets.3 -rearch_stage_copy 1
allsets.4 -rearch_stage_copy 1
endparams
#------------------------------------------------------------------------
# VSN Pool Directives
dkarcpool dk DISKVOL[2-4][0-9]
tparcpool tp VOL[9-9][0-9][0-9]
crarcpool cr CLD0000[0-9][0-9][0-9
#------------------------------------------------------------------------
# VSN Directives
vsns
datafiles.1 dk ARQFS1 DISKVOL[0-1][0-9] DISKVOL20 dkarcpool
datafiles.2 tp VOL[0-5][0-9][0-9]
datafiles.3 tparcpool
```

```
datafiles.4 crarcpool
endvsns
:wq
```

Archiving Directives

The archiving directives are listed alphabetically below.

Archiving Directive: archivemeta

The **archivemeta** directive controls whether file system metadata is archived.

Syntax

archivemeta=*state*

where *state* is either **on** or **off**. The default is **off**.

Description

The exact effects of the **archivemeta** directive depend on whether you are using a Version 1 or a Version 2 superblock:

- For Version 1 file systems, the archiver archives directories, removable media files, segment index inodes, and symbolic links as metadata.
- For Version 2 file systems, the archiver archives directories and segment index inodes as metadata. Removable media files and symbolic links are stored in inodes rather than in data blocks. They are not archived. Symbolic links are archived as data.

By default, metadata is not archived.

Scope

Global.

The **archivemeta** directive is entered in the global directives section at the start of the **archiver.cmd** file.

Recommendations

If files are often moved around and there are frequent changes to the directory structures in a file system, archive the file system metadata. But if the directory structures are reasonably stable, you can disable metadata archiving and reduce the actions performed by removable media drives.

Archiving Directive: *archive-set-name path*

The combination of an archive set name and a path constitutes an *archive set assignment directive* that defines a group of files that should be archived together.

Syntax

***archive-set-name path* [-access *interval* [-nftv]] [-after *date-time*] [-minsize *size*] [-maxsize *size*] [-user *username*] [-group *groupname*] [-name *regex*]**

where:

- *archive-set-name* is an administrator-defined name for the archive set.

- *path* specifies the path to the top-most subdirectory that contains set members relative to the root directory of the file system (see the Options section for descriptions of the optional arguments).

Description

The archive set assignment directive, *archive-set-name path*, specifies the path to a group of files that should be archived together and, optionally, additional characteristics of files that belong in the group.

Scope

Per file system.

Archive set assignment directives are entered in the archive set definitions section of the **archiver.cmd** file following a file system directive of the form **fs =** *file-system-name* .

Options

- *archive-set-name* is an administrator-defined name for the archive set. It must start with an upper or lower case letter and can contain up to 28 additional characters in any combination of upper and lower case letters [**A-Za-z**], numerals [**0-9**], and underscores (_). But do not use the reserved names **no_archive** and **all**.
- *path* specifies the path to a starting subdirectory. All files in the starting directory and its subdirectories are archived as a group. To include all of the files in a file system, use the dot (**.**) character. The path must not include the root directory character, the leading slash (**/**).
- **-access** (optional) re-archives files that have not been accessed for the amount of time specified by *interval*, where *interval* is an integer followed by one of following units: **s** (seconds), **m** (minutes), **h** (hours), **d** (days), **w** (weeks), and **y** (years).

  This parameter lets you schedule rearchiving of less used files from higher to lower cost media. The software validates the access and modification times for files to ensure that they are greater than or equal to the file creation time and less than or equal to the time at which the file is examined. The **-nftv** (no file time validation) parameter disables this validation.
- **-after** archives only files that have been created or modified after *date-time*, where *date-time* is an expression of the form *YYYY-MM-DD* **[***hh***:***mm***:***ss***] [Z]** and where **YYYY**, *MM*, *DD*, *hh*, *mm*, and *ss* are integers representing the year, month, day, hour, minutes, and seconds, respectively. The optional **Z** parameter sets the time zone to Coordinated Universal Time (UTC). The defaults are **00:00:00** and local time.
- **-minsize** and **-maxsize** archive only those files that are over or under the specified *size*, where *size* is an integer followed by one of the following units: **b** (bytes), **k** (kilobytes), **M** (megabytes), **G** (gigabytes), **T** (terabytes), **P** (petabytes), **E** (exabytes).
- **-user** *username* and **-group** *groupname* archive only files that belong to the specified user and/or group.
- **-name** archives all files that have path and file names matching the pattern defined by the regular expression *regex*.

Recommendations

In general, you should configure the smallest of number of the most inclusive archive sets possible. Archive sets have exclusive use of a set of archival media. So large numbers of archive sets each defined by excessively restrictive assignment criteria cause poor media utilization, high system overhead, and reduced performance. In extreme cases, jobs may fail due to lack of usable media, even though ample capacity remains in the library.

Examples

In the example, we do not have any strong need to selectively group files within the file system. So, to maximize performance and media utilization while minimizing overhead, we define a single archive set, **allfiles**. This archive set includes all files in path that starts in the file system root directory:

```
fs = hsmqfs1
logfile = /var/adm/hsmqfs1.archiver.log
allfiles .
...
```

Archiving Directive: *archive-set-name.copy-number media-specification*

The volume serial number association directive, *archive-set-name.copy-number* assigns archival media volumes to archive sets, either directly by VSN or by named media pool.

Syntax

```
archive-set-name.copy-number media-type volume-specification
archive-set-name.copy-number -pool vsn-pool-name
```

where:

- *archive-set-name* is the name that an archive set assignment directive assigns to the archive set that you are associating with the specified volumes.
- *copy-number* is the number that an archive copy directive assigned to the copy that you are associating with the specified volumes. It is an integer in the range [**1-4**]. See the archive set copy directive.
- *media-type* is one of the two-character, Oracle HSM media type identifiers listed in Appendix A and in the **mcf** man page. See the archive set assignment directive.
- *volume-specification* is a space-separated list of one or more regular expressions that match volume serial numbers. See the Solaris **regcmp** man page for details on regular expression syntax.
- **-pool** *vsn-pool-name* is a previously specified, named collection of archival media volumes that can be specified as a unit. See the VSN pools and VSN pool definition directives.

Scope

Per archive set.

The VSN assignment directives are entered in the **archiver.cmd** file following a **vsn** directive and before either an **endvsns** directive or the end of the file.

Examples

```
vsns
all.1 dk ^DISKVOL00[0-1][0-9] ^DISKVOL020
all.2 li ^VOL[0-3][0-9][0-9]
```

```
all.3 li ^VOL[3-6][0-9][0-9]
findat.1 li VSN001 VSN002 VSN003 VSN004 VSN005
tstdat.1 -pool tests_pool
endvsns
```

Archiving Directive: `archmax`

The **archmax** directive sets a maximum size for the files that are written to archival media.

Syntax

archmax = *media maximum-size*

where:

- *media* is one of the media types defined in Appendix A and on the **mcf** man page.
- *maximum-size* is the maximum size of the archive file for a given media type.

Description

The **archmax** directive sets a maximum size for tape archive (**.tar**) files. When *maximum-size* is reached, the archiver stops adding copies of data files to the archive file. Archiving continues with a new archive file.

Maximum sizes depend on the media in use. By default, the maximum archive file size for Oracle StorageTek T10000 and all LTO magnetic tape media is 22 gigabytes. For disk archives and optical media, the default is one gigabyte.

Scope

Global, per archive set, or per copy.

An **archmax** directive can be entered in the global directives section at the start of the **archiver.cmd** file or in the archive set definitions section, following an **fs =** *file-system-name* file system directive or as part of an archive set copy directive.

Recommendations

The defaults are optimal for general use.

Setting larger or smaller sizes for archive files may have both advantages and disadvantages. For example, if you are archiving to tape and set **archmax** to a large size, the tape drive stops and starts less often. But, when a large archive file does not quite fit into the space remaining on a volume, a lot of media capacity is wasted. To avoid problems, do not set the **archmax** directive to be more than 5 percent of the media capacity.

Archiving Directive: `bufsize`

The **bufsize** directive changes the size of the archiving buffer, optionally, locks the buffer.

Syntax

bufsize=*media number-blocks* [lock]

where:

- *media* is one of the media types defined in Appendix A and in the **mcf** man page
- *number-blocks* is a number in the range [**2-1024**]. The default is **4**.

- **lock** indicates whether the archiver can use locked buffers when making archive copies.

Description

The **bufsize** directive lets you set a non-default size for the memory buffer that caches archival data for writing. It also lets you lock the buffer.

Buffering improves overall I/O performance by insuring that writes are made efficiently. To set the buffer size, the archiver multiples the *number-blocks* parameter by the **dev_blksize** that is specified for the media in the **defaults.conf** file. See the **defaults.conf** (4) man page for details.

If **lock** is specified, the archiver sets file locks on the archive buffer in memory for the duration of the copying operation. This action avoids the overhead associated with locking and unlocking the buffer for each I/O request and results in a reduction in system CPU time.

By default, the archiver sets the buffer for **4** blocks and lets the file system lock the buffer as needed.

Scope

Global.

The **bufsize** directive is entered in the global directives section at the start of the **archiver.cmd** file. But the **-bufsize** and **-lock** copy parameters can be used to provide similar functionality on a per-copy basis.

Recommendations

For general use, the default buffering is usually optimal, so be careful when making changes.

If direct I/O is enabled and if large amounts of memory are available, using the **bufsize** directive with the **lock** argument can significantly reduce CPU overhead by eliminating the need to lock and unlock the buffer for each I/O request. Note, however, that **-lock** can cause an out-of-memory condition on systems that lack adequate memory. For information on enabling direct I/O, see the **setfa** (1), **sam_setfa** (3), and **mount_samfs** (1m) man pages.

Archiving Directive: *copy-number*

The **copy-number** or *archive set copy directive*, tells the archiver to make an archival copy of the files specified by the immediately preceding archive set assignment directive, *archive-set-name path*.

Syntax

copy-number **[*archive-age*] [-release [*attribute*]] [-norelease] [-stage [*attribute*]] [*unarchive-age*]**

where:

- The archive set copy directive appears immediately after the corresponding archive set assignment directive.
- *copy-number* is **1**, **2**, **3**, or **4** (see the Options section for definitions of the optional arguments).

Description

The archive set copy directive tells the archiver to make a copy of the corresponding archive set and, optionally, sets conditions for when the copy is made and how it is archived. Archive set copy directives begin with a *copy-number*, **1**, **2**, **3**, or **4**. The digit is followed by one or more arguments that specify archive characteristics for that copy.

By default, the archiver writes a single archive copy for files in the archive set when the archive age of the file is four minutes.

Scope

Per archive set.

The archive set copy directive immediately follows an archive set assignment directive in the archive set definitions section of the **archiver.cmd** file.

Options

archive-age
The optional *archive-age* parameter is the time that a new or modified file must spend in the disk cache before it becomes eligible for archiving.

Specify *archive-age* as one or more combinations of an integer and a unit of time, where units include **s** (seconds), **m** (minutes), **h** (hours), **d** (days), **w** (weeks), and **y** (years). The default is **4m** (4 minutes).

-release
The optional **-release** parameter clears the Oracle HSM releaser software to free the disk space used by files as soon as an archive copy has been made.

The optional release *attribute* is **-a**, **-n**, or **-d**, where:

- The **-a** (*associative staging*) attribute requires that the software stage all files that have been released from the archive set when any one of them is accessed.
- The **-n** attribute requires that the software read directly from the archive media and never stage files.
- The **-d** attribute resets the default staging behavior.

-norelease
The optional **-norelease** parameter keeps the Oracle HSM releaser software from freeing the disk space used by archived files until all copies marked with **-norelease** have been made.

-release -norelease
Used together, **-release -norelease** require that the Oracle HSM software free the disk space used by files immediately after all copies that are flagged **-release -norelease** are made. Oracle HSM does not wait for the releaser process to run.

-stage
The optional **-stage** parameter is **-a**, **-c** *copy-number*, **-n**, **-w**, or **-d**, where:

- **-a** requires staging of all files from the archive set when any one of them is accessed.
- **-c** *copy-number* requires that the software stage from the specified copy number.
- **-n** requires that the software read directly from the archive media and never stage files.
- **-w** requires that the software wait for each file to be successfully staged before proceeding (not valid with **-d** or **-n**).

- **-d** resets the default staging behavior.

unarchive-age
The *unarchive-age* parameter specifies the amount of time that an archival copy of a file spends in the archive before it is unarchived to free space on the media for reuse. Time is expressed as one or more combinations of an integer and a unit of time, where units include **s** (seconds), **m** (minutes), **h** (hours), **d** (days), **w** (weeks), and **y** (years).

Examples

The example below contains two copy directives for archive set **allsamma1**. The first directive does not release copy **1** until it reaches an archive age of five minutes (**5m**). The second directive does not release copy **2** until it reaches an archive age of one hour (**1h**) and unarchives copy **2** once it reaches the unarchive age of seven years and six months (**7y6m**):

```
# Archive Set Assignments
fs = samqfs1
logfile = /var/adm/samqfs1.archive.log
allfiles .
    1 -norelease 5m
    2 -norelease 1h 7y6m
```

Archiving Directive: *copy-number*[*archive-age*]

A **copy-number** and, optionally, an *archive-age* that directly follow an **fs** directive constitute a *metadata copy directive*. The metadata copy directive tells the archiver to make an extra copy of the file system metadata.

Syntax

copy-number **[*archive-age*]**

where:

- The directive immediately follows the **fs** directive that identifies the file system.
- *copy-number* is the ordinal number of the extra copy (**1** for the first, **2** for the second, etc.)
- *archive-age* is an optional time that must elapse before the copy is made, expressed as one or more combinations of an integer and a unit. Units include **s** (seconds), **m** (minutes), **h** (hours), **d** (days), **w** (weeks), and **y** (years).

Description

Each **copy-number** directive tells the archiver to make an extra copy of the file system metadata.

By default, Oracle HSM makes only a single copy.

Scope

Per file system.

The **copy-number** directive immediately follows the **fs =** *file-system-name* directive that identifies the file system.

Recommendations

Accept the default. Be cautious about specifying extra copies of file system metadata. If directories change frequently, specifying multiple metadata copies can cause excessive numbers of tape mount.

Example

In the example, copy **1** of the metadata for the **hsmqfs1** file system is made after 4 hours (**4h**) and copy **2** is made after twelve hours and 30 minutes(**12h30m**):

```
# General Directives
archivemeta = off
examine = noscan
# Archive Set Assignments
fs = hsmqfs1
1 4h
2 12h30m
logfile = /var/adm/hsmqfs1.archiver.log
allfiles .
    1 -norelease 15m
    2 -norelease 15m
...
```

Archiving Directive: `drives`

The **drives** directive limits the number of drives that the archiver can use in a specified robotic library.

Syntax

drives = *media-library count*

where:

- *media-library* is the family set name of the automated library as defined in the **mcf** file.
- *count* is the number of drives that the archiver can use.

Description

The **drives** directive lets you control how many drives the archiver uses, so that some can be reserved for staging or other uses. By default, the archiver uses all of the drives in an automated library for archiving.

You can also use the archive set copy parameters **-drivemax**, **-drivemin**, and **-drives** for this purpose.

Scope

Global or per copy.

A **drives** directive can be entered in the global directives section at the start of the **archiver.cmd** file or in the archive set definitions section, as part of an archive set copy directive. The related copy parameters **drivemax** and **drivemin** provide similar per-copy functionality.

Archiving Directive: `endparams`

The copy parameters directive **endparams** delimits the end of the copy parameters section of the **archiver.cmd** file. See **params**.

Syntax

```
copy-parameters ...
endparams]
```

where *copy-parameters* is one or more copy parameters.

Examples

```
# Copy Parameters
params
allsets -sort path -offline_copy stageahead
allsets.1 -startage 6h  -startsize 6G  -startcount 500000
allsets.2 -startage 24h -startsize 20G -startcount 500000 -drives 5
allsets.2 -rearch_stage_copy 1
endparams]
```

Archiving Directive: `endvsnpools`

The **vsnpools** directive marks the end of the section of the **archiver.cmd** file that defines groups of media that will be used for the same purposes. See **vsnpools**.

Syntax

```
vsn-pool-name-directives
endvsnpools
```

where *vsn-pool-directives* is one or more VSN pool directives.

Examples

```
vsnpools
hsmqfs1pool li VOL[0-4][0-9][0-9]
hsmqfs2pool li VOL[5-9][0-9][0-9]
endvsnpools
```

Archiving Directive: `endvsns`

The **endvsns** directive marks the end of the media-assignment section of the **archiver.cmd** file.

Syntax

```
vsn-association-directives
endvsns
```

where *vsn-association-directives* is one or more volume serial number association directives.

Example

```
vsns
all.1 dk ^DISKVOL00[0-1][0-9] ^DISKVOL020
all.2 li ^VOL[0-3][0-9][0-9]
all.3 li ^VOL[3-6][0-9][0-9]
findat.1 li VSN001 VSN002 VSN003 VSN004 VSN005
tstdat.1 -pool tests_pool
endvsns
```

Archiving Directive: `examine`

The **`examine`** directive tells the archiver how to identify files that are ready for archiving.

Syntax

```
examine = method
```

where *`method`* is one of the following directives:

- **`noscan`**
- **`scan`**
- **`scandirs`**
- **`scaninodes`**

Description

The **`examine`** directive can specify one of four ways of detecting files that require archiving:

- **`noscan`**, the default, specifies continuous archiving. After an initial scan, the archiver scans directories only when their contents change, requiring archiving. The archiver does not directory and inode information. This archiving method performs better than scan archiving, particularly for file systems with more than 1,000,000 files.
- **`scan`** specifies legacy scan archiving. The archiver scans directories once and always scans inodes thereafter.
- **`scandirs`** specifies scan archiving. The archiver always scans directories that do not have the **`no_archive`** attribute set, but never scans inodes.
- **`scaninodes`** specifies scan archiving. The archiver always scans inodes, but never scans directories.

Scope

Global.

The **`examine`** directive is entered in the global directives section at the start of the **`archiver.cmd`** file.

Recommendations

In general, accept the default, **`noscan`**, for best performance.

The archiver does not scan directories that are marked **`no_archive`**. So, to reduce overhead when using the **`scandirs`** method, set the **`no_archive`** attribute on directories that contain files that do not change.

Archiving Directive: `fs`

The **`fs`** directive limits the scope of a set of archiving and copy parameters to the file system specified in the directive.

Syntax

```
fs = file-system-name
```

where *`file-system-name`* is a the name of a file system defined in the **`mcf`** file.

Description

The **fs** directive identifies a file system and marks the start of a list of archiver directives and parameters that apply to that file system only.

Scope

Per file system.

The **fs** directive is entered in the archive set definitions section of the **archiver.cmd** file.

Example

In the example, all directives between the lines **fs = hsmqfs1** and **fs = hsmqfs2** apply only to the file system named**hsmqfs1**:

```
fs = hsmqfs1
logfile = /var/adm/hsmqfs1.archiver.log
allfiles .
    1 -norelease 15m
    2 -norelease 15m
fs = hsmqfs2
...
```

Archiving Directive: interval

If a file system has not been configured for continuous archiving (the default), the **interval** directive defines the amount of time that the archiver waits after checking for unarchived files before it checks again.

Syntax

interval = ***elapsed-time***

where *elapsed-time* is the number of seconds that must elapse between one file system scan and the next.

Description

If the **examine** directive is not set to **noscan** (continuous archiving) and if copy parameters do not set a start time for archiving, the **interval** directive supplies a default start time. When the number of seconds specified by the **interval** directive have elapsed since the archiver last scanned the file system, it scans again, using the method specified by the **examine** directive (**scan**, **scandirs**, or **scaninodes**).

The default is **600** seconds (10 minutes).

arrun and **arscan** commands issued via **samcmd** or the **samu** utility override the **interval** directive and start the specified action immediately.

The **hwm_archive** mount option can also override the **interval** directive and force archiving whenever the file system utilization passes the high-water mark for a given file system.

For more information about specifying the archive interval, see the **archiver.cmd** and **mount_samfs** man pages.

Scope

Global.

The **interval** directive is entered in the global directives section at the start of the **archiver.cmd** file.

Archiving Directive: `logfile`

The **logfile** directive defines the path and name of the archiver log file.

Syntax

```
logfile = path-and-name
```

where *path-and-name* is the absolute path and name of the file.

Description

The **logfile** directive tells the archiver to log every file that is archived, re-archived, or unarchived. The log file is thus a continuous record of archival action.

By default, the archiver does not maintain log files.

Scope

Global or per file system.

A **logfile** directive can be entered in the global directives section at the start of the **archiver.cmd** file or in the archive set definitions section, following a file system directive of the form **fs =** *file-system-name*.

Recommendations

Archiver log files are essential for recovering damaged or lost file systems and can be valuable for monitoring and analysis. So you should enable archiver logs, rotate them, and back them up frequently. For more information, see the *Oracle Hierarchical Storage Manager and StorageTek QFS Installation and Configuration Guide*.

Archiving Directive: `notify`

The **notify** directive identifies a script file that the archiver should use when notifying the administrator of events, notices, and alarm conditions.

Syntax

```
notify = path-and-name
```

where *path-and-name* is the path and name of a suitable script file.

Description

The archiver executes the specified script when it encounters one of the alarm conditions that you have specified. These could include **emerg**, **alert**, **crit**, **err**, **warning**, **notice**, **info**, **debug**, or others.

You can either edit the default script to meet your notification requirements or you can substitute an alternative. For full information, see the **archiver.sh** (1m) man page.

The default path and file name to the notification script is **/etc/opt/SUNWsamfs/scripts/archiver.sh**.

Scope

Global.

The **notify** directive is entered in the global directives section at the start of the **archiver.cmd** file.

Archiving Directive: ovflmin

The **ovflmin** directive enables or disables the Oracle HSM volume overflow feature.

Syntax

```
ovflmin = media minimum-file-size
```

where:

- *media* is one of the media types defined in Appendix A and in the **mcf** man page.
- *minimum-file-size* is the size of the smallest file that will be written to more than one archival volume.

Description

When volume overflow is enabled, for a given media type, the archiver can create large archive files that span multiple volumes. When the size of an archive file exceeds the specified minimum, the archiver writes the remaining portion of the file to another volume of the same media type. The portion of the file written to each volume is called a *section*.

Volume overflow files do not generate checksums. For more information on using checksums, see the **ssum** man page.

By default, volume overflow is disabled.

Scope

Global or per file system.

The **ovflmin** directive can be entered in the global directives section at the start of the **archiver.cmd** file or in the archive set definitions section, as part of an archive set copy directive.

Recommendations

Use volume overflow with caution after assessing its effects. Disaster recovery and recycling are significantly more difficult with files that span volumes.

Archiving Directive: params

The copy parameters directive **params** delimit the start of the copy parameters section of the **archiver.cmd** file. See **endparams**.

Syntax

```
params
copy-parameters ...]
```

where *copy-parameters* is one or more copy parameters.

Examples

```
# Copy Parameters
params
allsets -sort path -offline_copy stageahead
allsets.1 -startage 6h  -startsize 6G  -startcount 500000
```

```
allsets.2 -startage 24h -startsize 20G -startcount 500000 -drives 5
allsets.2 -rearch_stage_copy 1
endparams]
```

Archiving Directive: `scanlist_squash`

The **`scanlist_squash`** directive enables or disables recursive searches for unarchived files.

Syntax

`scanlist_squash =` ***`state`***

where *`state`* is either **`off`** or **`on`**.

Description

The **`scanlist_squash`** directive tells the archiver if it should look for unarchived files by scanning from each parent directory down through the subdirectories in the directory tree.

The default value is **`off`**.

Scope

Global or per archive set.

The **`scanlist_squash`** directive can be entered in the global directives section at the start of the **`archiver.cmd`** file or in the archive set definitions section, as part of an archive set copy directive.

Recommendations

Avoid enabling recursive scans for any file system that contains many modified files and/or subdirectories. In such cases, setting **`scanlist_squash = on`** can significantly reduce archiving performance.

Archiving Directive: `setarchdone`

The **`setarchdone`** global directive sets the **`archdone`** flag on files that will never be archived.

Syntax

`setarchdone =` ***`state`***

where *`state`* is either **`on`** or **`off`**.

Description

The **`setarchdone`** directive tells the archiver to set the **`archdone`** flag to **`on`** on all unarchived files that meet no archiving criteria will never be archived.

Normally, once the archiver has created all specified copies of an unarchived file, it sets an **`archdone`** flag on the file **`on`**. The flag tells subsequent archiving operations that the file has been archived and should thus be skipped until it is again modified. A file that meets none of the specified archiving criteria will never be archived, so the archiver never sets its **`archdone`** flag **`on`**.

When the **`setarchdone`** directive is **`on`**, the archiving process finds files that will never be archived and sets their **`archdone`** flags **`on`**. Such files will never be archived. While

this can reduce future archiving overhead, the evaluation of files increases overhead immediately and may adversely affect performance.

The default is **off** if the **examine** directive is set to **scandirs** or **noscan**, **on** if the **examine** directive is set to **scan** or **scaninodes**.

Scope

Global.

The **setarchdone** directive is entered in the global directives section at the start of the **archiver.cmd** file.

Recommendations

Accept the default. The archiver does not use the **archdone** flag during directory scans, and flagging files that will never be archived can be a time consuming operation, particularly when directories are large. This can hurt performance.

Archiving Directive: *vsn-pool-name media-type volume-specification*

The volume serial number (VSN) pool directive defines a named collection of archival media volumes that a volume serial number (VSN) association directive can specify as a unit.

Syntax

vsn-pool-name media-type volume-specification

where:

- *vsn-pool-name* is the name that you assign to the pool.
- *media-type* is one of the two-character, Oracle HSM media type identifiers listed in Appendix A and in the **mcf** man page.
- *volume-specification* is a space-separated list of one or more regular expressions that match volume serial numbers. See the Solaris **regcmp** man page for details on regular expression syntax.

Scope

Media usage.

The VSN pools directives are entered in the **archiver.cmd** file following a **vsnpools** directive and before either an **endvsnpools** directive or the end of the file.

Examples

The example below specifies three pools, one for file system **hsmqfs1**, one for file system **hsmqfs2**, and a **scratch** pool. A scratch pool is a set of volumes used when specific volumes in a VSN association are exhausted or when another VSN pool is exhausted. If one of the three specific pools is out of volumes, the archiver selects the scratch pool VSNs:

```
vsnpools
hsmqfs1_pool li ^VOL7[0-9][0-9]
hsmqfs2_pool li ^VOL8[0-9][0-9]
scratch li ^VOL9[0-5]
endvsnpools
```

Archiving Directive: vsnpools

The **vsnpools** directive marks the start of the section of the **archiver.cmd** file that defines groups of media that will be used for the same purposes. See **endvsnpools**.

Syntax

```
vsnpools
vsn-pool-name-directives
```

where *vsn-pool-directives* is one or more VSN pool directives.

Examples

```
vsnpools
hsmqfs1pool li ^VOL[0-4][0-9][0-9]
hsmqfs2pool li ^VOL[5-9][0-9][0-9]
endvsnpools
```

Archiving Directive: vsns

The **vsns** directive marks the start of the media-assignment section of the **archiver.cmd** file.

Syntax

```
vsns
vsn-association-directives
```

where *vsn-association-directives* is one or more volume serial number association directives.

Example

```
vsns
all.1 dk ^DISKVOL00[0-1][0-9] ^DISKVOL020
all.2 li ^VOL[0-3][0-9][0-9]
all.3 li ^VOL[3-6][0-9][0-9]
findat.1 li VSN001 VSN002 VSN003 VSN004 VSN005
tstdat.1 -pool tests_pool
endvsns
```

Archiving Directive: wait

The **wait** directive delays the start of archiving until an administrator issues a start signal.

Syntax

```
wait
```

Description

The **wait** directive delays the start of archiving until an administrator issues a start signal using the **samcmd** command, the **samu** interface, or the Oracle HSM graphical user interface. Once the administrator gives the signal, archiving proceeds as specified by the remaining directives and parameters in the **archiver.cmd** file.

By default, the archiver starts automatically when the **sam-fsd** initialization command runs.

Scope

Global or per file system.

The **wait** directive can be entered in the global directives section at the start of the **archiver.cmd** file or in the archive set definitions section, following an **fs =** *file-system-name* directive.

Recommendations

Use the **wait** directive when you need time to perform other actions before archiving begins or when you need to temporarily exclude an archive set from archiving.

Copy Parameters

All copy parameters take the same basic form:

Copy Parameter: *archive-set-name*[.*copy-number*] [*options*]

Copy parameters define how the copies specified by an archive set are created.

Syntax

***archive-set-name*[.*copy-number*] [*options*]**

where:

- *archive-set-name* is either the special directive **allsets** or the name of an archive set defined in the archive set definitions section of the **archiver.cmd** file.
- The optional **.** operator followed by a *copy-number* limits the application of the specified copy parameters to the archive copy specified by *copy-number* in the archive set definitions. *copy-number* is an integer in the range **1-4**.
- *options* is one or more of the copy parameter options listed below.

Description

The special **allsets** directive applies the specified copy parameters to all defined archive sets. It lets you simplify management and minimize configuration conflicts by applying a consistent, core set of copy options across all of your archive sets. Always set the **allsets** copy parameter first, so that its provisions are not overridden by later parameters.

Options

R
Limits application of the parameters to re-archived copies.

-startage *time*
Specifies interval between the moment when the first file is added to an archive request and the moment when archiving actually begins.

Specify *time* as one or more combinations of an integer and a unit of time, where units include **s** (seconds), **m** (minutes), **h** (hours), **d** (days), **w** (weeks), and **y** (years). The default is **2h** (two hours).

-startcount *count*
Specifies the minimum number of files in an archive request. Archiving begins when the number of files awaiting archiving reaches this threshold.

By default, a **-startcount** value is not set.

-startsize *size*
Specifies the minimum size, in bytes, of an archive request. Archiving begins when the total size of the files awaiting archiving reaches this threshold.

By default, *size* is not set.

-archmax *maximum-size*
Limits the size of an archive file to no more than *maximum-size*, where *maximum-size* is media-dependent.

The defaults are as follows:

- for magnetic tape media, 512 megabytes.
- for optical disc media, 5 megabytes.

-bufsize= *media-type number-blocks*
Sets the size of the write buffer that holds the archive file to *number-blocks*dev_blksize*, where:

- *number-blocks* is the number of tape blocks buffered, an integer in the range **2-32**.
- *dev_blksize* is the block size specified for the media type in the **defaults.conf** file.

The default is **4**.

-drivemax *maximum-size*
Limits the amount of data archived using a single drive to no more than *maximum-size* megabytes, where *maximum-size* is an integer.

When multiple drives are specified using the **-drives** parameter, limiting the amount of data written to any one drive can help to balance workloads and improve overall drive utilization.

By default, *maximum-size* is not specified.

-drivemin *minimum-size*
Limits the amount of data archived using one drive to at least *minimum-size* megabytes, where *minimum-size* is an integer.

Use the **-drivemin** option to optimize drive utilization. Set *minimum-size* large enough for the transfer time to significantly exceed the time required to load, position, and unload media and large enough to insure that multiple drives are only used when actually needed.

The default is the value of **-archmax** (if specified) or the value listed for the media type in the **defaults.conf** file.

-drives *number*
Limits the number of drives used for archiving to at most *number*, where *number* is an integer.

Setting a higher maximum number of drives can improve performance when archive sets contain large files or large numbers of files. If the available drives operate at different speeds, specifying multiple drives can also balance these variations and increase archiving efficiency.

The default is **1**.

-fillvsns
Forces the archiving process to use smaller archive files that fill archival media volumes more completely.

By default, the archiver selects a volume with enough space to hold the all files in an archive copy. This results in larger archive files that may not fit into the remaining capacity on many cartridges. As a result, media is under-utilized overall. The **-fillvsns** parameter addresses this issue, but at the cost of more media mounts, positioning operations, and unmounts, all of which reduce archiving and staging performance.

-lock
Mandates the use of locked buffers when making archive copies using direct I/O. Locked buffers prevent paging of the buffer and improve direct I/O performance.

The **-lock** parameter can cause an out-of-memory condition if specified on systems that have limited memory available.

By default, locked buffers are not mandated, and the file system retains control over the archiving buffer.

-offline_copy *method*
Specifies how archive copies are made when files have already been released from the disk cache.

Files can be released from the disk cache once a single archive copy is made. In such cases, any additional copies specified must be made from the archive copy. Use the **-offline_copy** *method* option to balance the amount of extra space that you can afford to provide in the disk cache against the number of extra drives that you can afford to make available for use in copying archival media. Each *method* has advantages in some circumstances:

- **direct**

  The **direct** method copies files directly from the volume that holds the first copy to another volume.

  This approach requires two drives.

  This approach may require additional buffer space. So increase the value set by the **stage_n_window** mount option when using this method.

- **stageahead**

  The **stageahead** method stages the next file needed for the current archive copy into the disk cache while writing the current file out to archival media.

  This approach requires two drives.

- **stageall**

  The **stageall** method stages all files needed for the current archive copy into the disk cache before writing any files out to archival media.

  This approach requires only one drive.

  This approach requires more space in the disk cache.

- **none**

  none stages files to the disk cache as needed before copying them out to archival media.

  none is the default method.

-sort *criterion*
Arranges files by *criterion* before archiving them. The sorting *criterion* can be one of the following:

- **age** specifies sorting by modification time, from oldest to most recent.
- **path** (the default) specifies sorting by full path name and thus keeps files that reside in the same directories together on the archive media.
- **priority** specifies sorting by archiving priority, from highest to lowest.
- **size** sorts files by file size, from smallest to largest.
- **none** specifies no sorting and archives files in the order in which they are encountered in the file system.

By default, the archiver sorts by **path**.

-rsort *criterion*
Sorts files by *criterion* like **-sort**, but in reverse order.

-recycle_dataquantity *size*
Limits the amount of data that the recycler will schedule for rearchiving to *size* bytes, where *size* is an integer.

The recycler schedules rearchiving when it needs to drain archival volumes of valid archive files. Note that the actual number of volumes selected for recycling may also depend on the **-recycle_vsncount** parameter.

The default is **1073741824** (one gigabyte).

-recycle_hwm *percent*
Sets the maximum percent media utilization (the high water mark or **hwm**) that initiates recycling of removable media. The parameter is ignored for disk media (see **-recycle_minobs** below).

The default is **95** percent.

-recycle_ignore
Prevents actual recycling of any media in the archive set, while allowing recycling processes to run normally. Use the **-recycle_ignore** option to test recycling policies non-destructively before committing them to production use.

-recycle_mailaddr *mail-address*
Directs informational recycler messages to *mail-address*.

By default, recycler messages are not sent.

-recycle_mingain
Limits selection of volumes for recycling to those which would increase their free space by at least the specified *percentage*.

The default value is **50** percent.

-recycle_vsncount
Limits the number of volumes that the recycler schedules for rearchiving to *count*.

The actual number of volumes selected for recycling may also depend on the **-recycle_dataquantity** option.

The default is **1** percent. The option is ignored for disk media.

-recycle_minobs
Sets the *percentage* of obsolete files in a disk-resident archive file that triggers rearchiving of the valid files and eventual deletion of the original **tar** file.

The default is **50**. The parameter is ignored for removable media (see **-recycle_hwm** above).

-unarchage
Sets the reference time for computing the unarchive age to *time_ref*, where *time_ref* is either **access** for the file access time (the default) or **modify** for the modification time.

-tapenonstop
Writes a single tape mark and an end-of-file (EOF) label at the end of the archive file without closing the removable media file. This speeds transfer of multiple archive files, but the tape cartridge cannot be unloaded until the entire archive set has been written to tape.

By default, Oracle HSM software closes the tape file by writing two additional tape marks after the end-of-file label at the end of the archive file.

-reserve *keyword*
Reserves a removable media volume for the exclusive use of a specified archive set.

When a volume is first used to hold files from the archive set, the software assigns the volume a unique reserve name based on one or more specified keywords: **fs**, **set**, and/or one of the following: **dir** (directory), **user**, or **group**.

fs
Includes the file system name in the reserve name: **arset.1 -reserve fs**.

set
Includes the archive set name from the archive set assignment directive in the reserve name: **all -reserve set**.

dir
Includes the first 31 characters of the directory path specified in the archive set assignment directive in the reserve name.

user
Includes the user name associated with the archive file: **arset.1 -reserve user**.

group
Includes the group name associated with the archive file: **arset.1 -reserve group**.

-priority *multiplier ranking*
Changes the archiving priority of files when used with the **sort priority** parameter listed above.

See the **archiver** and **archiver.cmd** man pages for a full explanation of priorities.

Recommendations

Exercise caution when using the **-reserve** *keyword* option. Reserving volumes by set can be advantageous in some situations. But be aware that it is inherently less efficient than allowing the software to select the media. When volumes are reserved, the system must mount, unmount, and position cartridges more often, increasing overhead and reducing performance. Highly restrictive reservation schemes under-utilize available

media and, in extreme cases, may cause archiving failures due to lack of available media.

Staging

Staging directives configure archiving operations. They may apply globally, to all file systems or they may apply to a specific file system or archive set, depending on the directive.

The stager starts when the **samd** daemon runs. The stager has the following default behavior:

- The stager attempts to use all the drives in the library.
- The stage buffer size is determined by the media type, and the stage buffer is not locked.
- No log file is written.
- Up to 1000 stage requests can be active at any one time.

You can customize the stager's operations for your site by inserting directives into the **/etc/opt/SUNWsamfs/stager.cmd** file.

The stager.cmd File

The directives in the **stager.cmd** file override the default behaviors. You can configure the stager to stage files immediately, to never stage files, to partially stage files, and to specify other staging actions. For example, specifying the never-stage attribute benefits applications that access small records from large files because the data is accessed directly from the archive media without staging the file online.

If you are using the Oracle HSM Manager software, you can control staging from the File System Summary or File System Details page. You can browse the file system and see the status of individual files, use filters to view certain files, and select specific files to stage. You can select which copy to stage from or let the system choose the copy.

The example shows a **stager.cmd** file after all possible directives have been set.

```
drives=dog 1
bufsize=od 8 lock
logfile=/var/adm/stage.log
maxactive=500
```

Staging Directives

Staging directives are listed alphabetically below. For additional information about stager directives, see the **stager.cmd** man page.

Staging Directive: drives

The **drives** directive specifies the number of drives that the stager can use when copying back files from archival media to the disk cache.

Syntax

drives=*library count*

where:

- *library* is the family set name of the library as it appears in the **mcf** file.

- *count* is the maximum number of drives used.

Description

The **drives** directive sets limits to the number of drives that the stager can use.

By default, *count* is the number of drives listed for the library in the **mcf** file, so the stager uses all available drives unless otherwise directed.

Recommendations

Use the **drives** directive if users and applications frequently request staging. Under these conditions, the default value can let the stager may monopolize all available drives and interfere with archiving.

Examples

The example specifies that only one drive from library **SL150_500** is used for staging files:

```
drives = SL150_500 1
```

Staging Directive: bufsize

The **bufsize** directive changes the size of the archiving buffer, optionally, locks the buffer.

Syntax

bufsize=*media number-blocks* **[lock]**

where:

- *media* is one of the media types defined in Appendix A and in the **mcf** man page
- *number-blocks* is a number in the range [**2-8192**].
- **lock** indicates whether the archiver can use locked buffers when making archive copies.

Description

The **bufsize** directive lets you set a non-default size for the memory buffer where the stager caches file data that it reads from archival media before writing to disk cache. It also lets you lock the buffer.

Buffering improves overall I/O performance by insuring that writes are made efficiently. To set the buffer size, the archiver multiples the *number-blocks* parameter by the **dev_blksize** that is specified for the source media in the **defaults.conf** file. See the **defaults.conf** (4) man page for details.

If **lock** is specified, the archiver sets file locks on the archive buffer in memory for the duration of the copying operation.

By default, the stager sets the buffer for **16** blocks and lets the file system lock the buffer as needed.

Recommendations

For general use, the default buffering is usually optimal, so be careful when making changes.

If you do set a new buffer size, specify a number of blocks that is consistent with the amount of memory installed in the system. The higher the number specified for *number-blocks*, the more memory the stager uses.

If direct I/O is enabled and if large amounts of memory are available, using the **bufsize** directive with the **lock** argument can significantly reduce CPU overhead by eliminating the need to lock and unlock the buffer for each I/O request. Note, however, that **-lock** can cause an out-of-memory condition on systems that lack adequate memory. For information on enabling direct I/O, see the **setfa** (1), **sam_setfa** (3), and **mount_samfs** (1m) man pages.

Staging Directive: `logfile`

The **logfile** directive defines the path and name of the stager log file.

Syntax

```
logfile = path-and-name [event-list]
```

where:

- *path-and-name* is the absolute path and name of the file.
- *event-list* is an optional, space-delimited list of event types.

Description

The **logfile** directive tells the stager to log every file that is staged. The log file typically contains the name of the file, the date and time, and the volume serial number (VSN) of the source media. The optional event list can specify one or more of the following event types for inclusion in the log:

all
Logs all staging events.

start
Logs the time when the archiver starts staging the file.

finish
Logs the time when the archiver finishes staging a file.

cancel
Logs cancellations of staging operations.

error
Logs staging errors.

By default, the archiver does not maintain log files. If logging is specified without an event list, **finish**, **cancel**, and **error** events are logged by default.

Log Fields

Stager log entries take the following form:

```
status date time media-type volume position.offset inode filesize filename copy
user group requestor equipment-number validation
```

where:

- *status* is **S** for starting, **C** for canceled, **E** for error, **F** for finished.

- *date* is the date in the form *yyyy*/*mm*/*dd*, where *yyyy* is a four-digit number representing the year, *mm* is a two-digit number representing the month, and *dd* is a two-digit number representing the day of the month.
- *time* is the time in the form *hh*:*mm*:*ss* format, where *hh*, *mm*, and *ss* are a two-digit numbers representing the hour, minute, and seconds, respectively.
- *media-type* is one of the two-character, Oracle HSM media type identifiers listed in Appendix A and in the **mcf** man page.
- *volume* is the volume serial number (VSN) of the media that holds the file being staged.
- *position*.*offset* is a pair of hexadecimal numbers separated by a dot that represent position of the start of the archive (**tar**) file on the volume and the offset of the staged file relative to the start of the archive file.
- *inode* is the inode number and generation number of the staged file, separated by a dot.
- *filesize* is the size of the staged file.
- *filename* is the name of the staged file.
- *copy* is the archive copy number of the copy that contains the staged file.
- *user* is the user that owns the file.
- *group* is the group that owns the file.
- *requestor* is the group that requested the file.
- *equipment-number* is the equipment ordinal number defined in the **mcf** file for the drive from which the file was staged.
- *validation* indicates whether the staged file is being validated (**V**) or not validated (**-**).

Examples

The first example shows a **logfile** directive that creates a stage log in the **/var/adm/** directory:

```
logfile=/var/adm/stage.log
```

The second example shows part of a typical stager log:

```
S 2016/11/09 14:06:27 dk disk01 e.76d 2557.1759 1743132 /hsmfs/dat0/3f 1 root
other root 0 -
F 2016/11/09 14:06:27 dk disk01 e.76d 2557.1759 1743132 /hsmfs/dat0/b9 1 root
other root 0 -
S 2016/11/09 14:06:27 dk disk02 4.a68 1218.1387 519464 /hsmfs/dat1/a0 1 root other
root 0 -
S 2016/11/09 14:06:43 dk disk01 13.ba5 3179.41 750880 /hsmfs/dat0/cl 1 root other
root 0 -
F 2016/11/09 14:06:43 dk disk01 13.ba5 3179.41 750880 /hsmfs/dat0/cf 1 root other
root 0 -
```

Staging Directive: maxactive

The **maxactive** directive lets you specify the number of stage requests that can be active at any one time.

Syntax

maxactive=*number*

where *number* is an integer in the range [**1-500000**].

Description

The **maxactive** directive limits the number of staging requests that the stager can handle at any one time.

The default is **4000**.

Examples

The example specifies that no more than 500 stage requests can be in the queue simultaneously:

```
logfile = /var/opt/SUNWsamfs/log/stager
drives = li 1
maxactive=500
```

Staging Directive: copysel

The copy selection directive, **copysel**, lets you specify the order in which the stager selects the archive set copy that it will use when staging a requested file to the disk cache.

Syntax

copysel=*selection-order*

where *selection-order* is a colon-delimited list of copy numbers in first-to-last order.

Description

The **copysel** directive lets you override the stager's usual selection order.

Normally, the stager first looks at the first copy of the corresponding archive set. If it finds usable file, the stager copies it back to the disk cache. If not, it moves on to the next copy, until it either finds a usable copy of the file or exhausts all available copies. For more information, see the **stager.cmd** (4) man page.

So, by default, the copy selection order **1:2:3:4**.

Example

The example shows a **stager.cmd** file that sets non-default copy-selection orders for file systems **samfs1** and **samfs2**:

```
logfile = /var/opt/SUNWsamfs/log/stager
drives = li 1
fs = samfs1
copysel = 4:3:2:1
fs = samfs2
copysel = 3:1:4:2
```

Adjusting the Preview Queue

When an Oracle HSM process requests a removable media volume that is not currently loaded into a drive, the request is added to the *preview queue*. Queued requests are satisfied in first-in-first-out (FIFO) order by default. But you can override the default

behavior by editing the file **/etc/opt/SUNWsamfs/preview.cmd**. The Oracle HSM library-control daemon (**sam-amld**) reads these directives when it starts and uses them until it stops. You cannot change queue priorities dynamically.

There are three types of directives:

- Global directives are placed at the top of the file and apply to all file systems.
- File-system directives take the form **fs=***directive* and are specific to individual file systems.
-

This section lists the global and file system-specific preview directives and concludes with a sample **preview.cmd** file.

The preview.cmd file

The aggregate priority for any given media mount request is determined using the values set by all weighting factors, according to the following formula:

priority = vsn_priority + *wm_priority* + (age_priority * *time-waiting-in-queue*)

where *wm_priority* is the water mark priority currently in effect (**hwm_priority**, **lwm_priority**, **hlwm_priority**, or **lhwm_priority**) and *time-waiting-in-queue* is the number of seconds that the volume request has been queued. For a full explanation of priority calculation, see the **PRIORITY CALCULATION** section of the **preview.cmd** man page.

Under special conditions—when access to data is critically important or when removable media drives are in short supply—the directives in the **preview.cmd** file let you better match file-system activity to operational requirements and available resources. The integrity of stored data is unaffected by the settings in the **preview.cmd** file, so you can freely experiment until you find the proper balance between archiving and staging requests.

You may need to adjust the default priority calculation for either or both of the following reasons:

- to insure that staging requests are processed before archive requests, so that files are available when users and applications access them.
- to insure that archive requests gain top priority when a file system is about to fill up

The sample **preview.cmd** file below addresses the conditions highlighted above:

```
# /etc/opt/SUNWsamfs/preview.cmd
# Use default weighting value for vsn_priority:
vsn_priority=1000.0
age_priority = 1.0
# Insure that staging requests are processed before archive requests:
lwm_priority = -200.0
lhwm_priority = -200.0
hlwm_priority = -200.0
# Insure that archive requests gain top priority when a file system is about to
fill up:
hwm_priority = 500.0
```

Negative weighting values for **lwm_priority**, **lhwm_priority**, and **hlwm_priority** insure that stage requests have priority over archive requests whenever space is available in the disk cache, so that data is always accessible when requested. If several

requests are sitting in the queue for 100 seconds and the file system is below the low water mark, then:

- An archiving mount request for a priority volume has the aggregate priority **1000+(-200)+(1x100)=900**
- A staging mount request for a priority volume has the aggregate priority **1000+0+(1x100)=1100**
- A staging mount request for a non-priority volume has the aggregate priority **0+0+(1x100)=100**

But when the disk cache is near capacity, archiving requests need to take priority. If too few files are archived as the file system fills, there is no space available for staging archived files or ingesting new ones. If several requests are sitting in the queue for 100 seconds and the file system is above the high water mark, then:

- An archiving mount request for a priority volume has the aggregate priority **1000+500+(1x100)=1600**
- A staging mount request for a priority volume has the aggregate priority **1000+0+(1x100)=1100**
- A staging mount request for a non-priority volume has the aggregate priority **0+0+(1x100)=100**

Preview Queue Directives

The following are purely global directives:

- **vsn_priority**
- **age_priority**.

Preview Queue Directive: age_priority

The **age_priority** directive changes the relative priority given to the amount of time that a request spends in the queue.

Syntax

age_priority=*weighting-factor*

where *weighting-factor* is a real number greater, less than, or equal to **1.0**.

Description

The **age_priority** directive adjusts the priorities of files so that you can you keep older requests from being indefinitely superseded by newer, higher-priority, requests or, conversely, keep newer, higher priority requests from being blocked by older requests. The directive specifies a multiplier that changes the relative weighting of the time spent in the queue.

Values greater than **1.0** increase the weight given to time spent in the queue when calculating the aggregate priority. Values less than **1.0** reduce the weight given to time spent in the queue when calculating the total priority. Values equal to **1.0** do not change the relative weight given to time spent in the queue.

The default is **1.0**.

Scope

Global.

Example

```
root@hsmmds1:~# vi /etc/opt/SUNWsamfs/preview.cmd
vsn_priority = 1000.0
age_priority = 1.5
lwm_priority = 0.0
lhwm_priority = 0.0
hlwm_priority = 0.0
hwm_priority = 0.0
```

Preview Queue Directive: `hlwm_priority`

The **`hlwm_priority`** directive adjusts the relative weight given to archiving requests versus staging requests when the disk cache is emptying.

Syntax

`hlwm_priority=`*`weighting-factor`*

where *`weighting-factor`* is a real number.

Description

The **`hlwm_priority`** directive adjusts the relative weight given to archiving requests versus staging requests when the disk cache is emptying, and cache utilization is between the high and low water marks (**`hwm`** and **`lwm`**). In this situation, reducing the relative weight given to archiving and thereby raising the priority of staging requests places more files in the disk cache, reduces demand for media mounts, and increases file system performance.

The default is **`0.0`**.

Scope

Global or per file system.

Example

```
root@hsmmds1:~# vi /etc/opt/SUNWsamfs/preview.cmd
vsn_priority = 1000.0
age_priority = 0.0
lwm_priority = 0.0
lhwm_priority = 1.0
hlwm_priority = 0.0
hwm_priority = 0.0
```

Preview Queue Directive: `hwm_priority`

The **`hwm_priority`** directive adjusts the relative weight given to archiving requests versus staging requests when file system disk cache is nearly full.

Syntax

`hwm_priority=`*`weighting-factor`*

where *`weighting-factor`* is a real number.

Description

The **hwm_priority** directive adjusts the relative weight given to archiving requests versus staging requests when file system utilization exceeds the high water mark (**hwm**), the point where the releaser process starts and begins reclaiming the disk space occupied by files that have copies on archival media. In this situation, increasing the relative weight given to archiving lets the releasing process free more space for staged archive copies and new files.

The default is **0.0**.

Scope

Global or per file system.

Example

```
root@hsmmds1:~# vi /etc/opt/SUNWsamfs/preview.cmd
vsn_priority = 1000.0
age_priority = 0.0
lwm_priority = 0.0
lhwm_priority = 0.0
hlwm_priority = 0.0
hwm_priority = 1.0
```

Preview Queue Directive: lhwm_priority

The **hlwm_priority** directive adjusts the relative weight given to archiving requests versus staging requests as the disk cache fills.

Syntax

lhwm_priority=*weighting-factor*

where *weighting-factor* is a real number.

Description

The **hlwm_priority** directive adjusts the relative weight given to archiving requests versus staging requests when the disk cache is filling up, and cache utilization is between the low and high water marks (**lwm** and **hwm**). In this situation, increasing the relative weight given to archiving lets the releasing process free more space for staged archive copies and new files. The directive takes the following form:

The default is **0.0**.

Scope

Global or per file system.

Example

```
root@hsmmds1:~# vi /etc/opt/SUNWsamfs/preview.cmd
vsn_priority = 1000.0
age_priority = 0.0
lwm_priority = 0.0
lhwm_priority = 1.0
hlwm_priority = 0.0
hwm_priority = 0.0
```

Preview Queue Directive: `lwm_priority`

The **`lwm_priority`** directive adjusts the relative weight given to archiving requests versus staging requests when the file system disk cache is nearly empty.

Syntax

```
lwm_priority=weighting-factor
```

where *`weighting-factor`* is a real number.

Description The **`lwm_priority`** directive adjusts the relative weight given to archiving requests versus staging requests when file system utilization drops below the low water mark (**`lwm`**), the point where the releaser process stops. In this situation, reducing the relative weight given to archiving and thereby raising the priority of staging requests places more files in the disk cache, reduces demand for media mounts, and increases file system performance.

The default is **`0.0`**.

Scope

Global or per file system.

Example

```
root@hsmmds1:~# vi /etc/opt/SUNWsamfs/preview.cmd
vsn_priority = 1000.0
age_priority = 0.0
lwm_priority = 1.0
lhwm_priority = 0.0
hlwm_priority = 0.0
hwm_priority = 0.0
```

Preview Queue Directive: `vsn_priority`

The **`vsn_priority`** directive increases the priority of volumes (VSNs) that are flagged as high-priority volumes by a specified value.

Syntax

```
vsn_priority =value
```

where *`value`* is a real number.

Description

The **`vsn_priority`** directive increases the priority of previously identified volumes by a specified amount. Priority flags are set using the **`chmed`** (1m) command:

```
chmed +p media-type.volume-serial-number
```

where:

- *`media-type`* is one of the two-character, Oracle HSM media types listed in Appendix A and on the **`mcf`** man page.
- *`volume-serial-number`* is the alphanumeric string that uniquely identifies the high-priority volume in the library.

The default priority value for a volume is **`1000.0`**.

Scope

Global.

Example

```
root@hsmmds1:~# chmed +p li.VOL011
root@hsmmds1:~# chmed +p li.VOL031
root@hsmmds1:~# chmed +p li.VOL074
root@hsmmds1:~# vi /etc/opt/SUNWsamfs/preview.cmd
vsn_priority = 1500.0
age_priority = 0.0
lwm_priority = 0.0
lhwm_priority = 0.0
hlwm_priority = 0.0
hwm_priority = 0.0
```

E

Understanding Archiver and Migration Logs

Archiver and migration logs record the exact locations of files that have been copied to tape. Should you need to recover a file system, these log files contain information that lets you restore any files that cannot be found using the available recovery point files.

The following table defines each field in the archiver log.

Field	Typical Value	Meaning
1	**A**	The type of archive activity logged: **A** (*archived*), **R** (*re-archived*), or **U** (*unarchived*)
2	**2015/03/23**	The date of the archive action, in the form *yyyy*/*mm*/*dd*.
3	**18:42:06**	The time of the archive activity, in the form *hh*:*mm*:*ss*.
4	**ti**	The archive media type. Media types, are discussed in Appendix A, "Glossary of Equipment Types" and on the **mcf(4)** man page.
5	**VOL004**	The volume serial number (VSN) of a removable media volume or the volume name and **tar**(1) path to a file archived on disk media.
6	**arset0.1**	The Oracle HSM archive set name and copy number.
7	**9a089.132**	The physical position of the start of the archive file (tar file) on media and the file offset within the archive file, in hexadecimal format.
8	**hsm1**	The name of the file system that holds the file.
9	**118.51**	The inode number and generation number of the file. The generation number is used in addition to the inode number for uniqueness because inode numbers are reused.
10	**162514**	The length of the file if the file is written on only one volume. Length of the section if the file is written on multiple volumes.
11	**t0/fdn**	The path and name of the file relative to the mount point of the file system.
12	**f**	The type of file: **d** (*directory*), **f** (*file*), **l** (*symbolic link*), **R** (*removable-media file*), **I** (*segment index*), or **S** (*data segment*)
13	**0**	The section number of an overflowed file or segment. If the file is an overflowed file, the value is non-zero. Otherwise the value is 0.
14	**56**	The equipment ordinal of the drive on which the file was archived.

The following example shows sample lines from an archiver log file.

```
A 2014/03/23 18:42:06 ti VOL004 arset0.1 9a089.1329 hsm1 118.51 162514 t0/fdn f 0 54
A 2014/03/23 18:42:10 ti VOL004 arset0.1 9aac2.1 hsm1 189.53 1515016 t0/fae f 0 56
A 2014/03/23 18:42:10 ti VOL004 arset0.1 9aac2.b92 hsm1 125.53 867101 t0/fai f 0 50
```

```
A 2014/03/24 13:30:24 dk DISK01/d8/d16/f2 arset4.1 810d8.1 hsm1 11971.30 1136048 t1/dat0 f 0 0
A 2014/03/24 13:30:25 dk DISK01/d8/d16/f2 arset4.1 810d8.8d hsm1 11973.9 1849474 t1/dat9 f 0 0
A 2014/03/24 13:30:25 dk DISK01/d8/d16/f3 arset4.1 810d8.96 hsm1 119576.6 644930 t1/file7 f 0 0
```

Media migration logs are very similar to archive logs. The archive-activity and the archive-set/copy-number fields are omitted, and the last three fields are unique to migration logs. All other fields are the same. Note, however, that there is a separate migration log for each volume migrated. Each log file is in the logging directory specified by the **migrationd.cmd** file.

Field	Typical Value	Meaning
1	**2015/03/23**	The date of the migration action, in the form *yyyy*/*mm*/*dd*.
2	**18:42:06**	The time of the migration activity, in the form *hh*:*mm*:*ss*.
3	ti	The archive media type. Media types, are discussed in Appendix A, "Glossary of Equipment Types" and on the **mcf(4)** man page.
4	**VOL004**	The volume serial number (VSN) of a removable media volume or the volume name and **tar**(1) path to a file archived on disk media.
5	**9a089.19**	The physical position of the start of the archive file (tar file) on media and the file offset within the archive file, in hexadecimal format.
6	**hsm1**	The name of the file system that holds the file.
7	**118.51**	The inode number and generation number of the file. The generation number is used in addition to the inode number for uniqueness because inode numbers are reused.
8	**162514**	The length of the file if the file is written on only one volume. Length of the section if the file is written on multiple volumes.
9	**dat0/datA**	The path and name of the file relative to the mount point of the file system.
10	**f**	The type of file: **d** (*directory*), **f** (*file*), **l** (*symbolic link*), **I** (*segment index*), or **S** (*data segment*)
11	**s**	The copy mode used for the migration: either **s** (*server copy*) or **x** (*xcopy*).
12	**801**	The equipment ordinal of the drive that mounted the source volume.
13	**804**	The equipment ordinal of the drive that mounted the destination volume.

The following example shows representative lines from the migration log file **hsm_migration_logs/li.VOL001**, which logs migration of the LTO tape **VOL001** to new media:

```
2015/10/16 12:14:12 li VOL012 2 4.1 hsmfs1 1026.1 0 .domain f s 804 801
2015/10/16 12:14:12 li VOL012 2 4.2 hsmfs1 1025.1 0 .fuid f s 804 801
2015/10/16 12:14:12 li VOL012 2 6.1 hsmfs1 1040.1 14971 data0/dat0A f s 804 801
2015/10/16 12:14:12 li VOL012 2 6.20 hsmfs1 1041.1 14971 data0/dat0B f s 804 801
2015/10/16 12:14:12 li VOL012 2 6.3f hsmfs1 1042.1 14971 data0/dat0C f s 804 801
2015/10/16 12:14:12 li VOL012 2 6.5e hsmfs1 1043.1 14971 data0/dat0D f s 804 801
```

F

Product Accessibility Features

Users with low vision, blindness, color blindness, or other visual impairments can access the Oracle Hierarchical Storage Manager and StorageTek QFS Software (Oracle HSM) via the commandline interface. This text-based interface is compatible with screen readers, and all functions are controlled using a keyboard.

Glossary

This glossary focuses on terms specific to Oracle Hierarchical Storage Manager and StorageTek QFS Software and file systems. For industry standard definitions, please refer to the dictionary maintained by the Storage Networking Industry Association at **`http://www.snia.org/education/dictionary/`**.

active metadata server

See **metadata server (MDS)**.

addressable storage

All storage space that is user-referenced through an Oracle HSM file system. See **online storage**, **nearline storage**, and **offsite storage**.

admin set

A set of user- and/or group-owned storage that administrators use. Admin sets are typically created to administer storage for projects that involve users from several groups and span multiple files and directories.

archival media

Media that stores copies of the files in an Oracle HSM file system. Archival media can include removable tape cartridges, magneto-optical cartridges, disk file systems configured as archival volumes, and cloud storage volumes.

archival storage

Data storage space created on archival media.

archive set

A collection of files that are copied to archival media together, using a common set of policies and parameters. Set membership determines the number of copies made, the parameters of the copying process, and the media used.

The **`archiver.cmd`** file defines archive sets by file system, directory location, size, and/or user and group ownership. See the **`archiver.cmd`** (4) man page for additional details.

archiver

The Oracle HSM program that manages the process of copying files to archival media. See the **`archiver`** (1m) man page for additional details.

associative staging

Copying a group of files that are no longer resident in the disk cache from archival media back to the Oracle HSM disk cache when a user or application accesses any one

member of the group. Associative staging insures that files that are used together are staged together. File owners can associate any files that reside in the same directory by setting the associative-staging attribute on the related files. See the **`staging`** (1) man page, **staging**, and **disk cache** for additional details.

audit (full)

The process of loading cartridges to verify their volume serial numbers (VSNs). For magneto-optical cartridges, the capacity and space information is determined and entered into the automated library's catalog.

automated library

A device that stores removable media cartridges, loads them into drives, and unloads them without operator intervention. An automated library contains cartridge storage slots, one or more drives, a transport mechanism for the cartridges, and, often, a mechanism for ingesting and exporting cartridges. See **direct-attached automated library**, **network-attached automated library**, **robot**, **transport**.

backup

A snapshot of a collection of files for the purpose of preventing inadvertent loss. A backup includes both the file's attributes and associated data.

block allocation map

A bitmap representing each available block of storage on a disk and indicating whether the block is in use or free.

block size

The size of the smallest addressable data unit on a block device, such as a magnetic tape cartridge or hard disk. The block size for a Linear Tape Open (LTO) cartridge is 256 kilobytes. For an Oracle StorageTek T10000 tape cartridge, the block size is 2048 kilobytes. For disk devices, block size is equivalent to *sector size*, which is typically 512 bytes.

buffered I/O

Writing and reading data to storage media, such as magnetic tape or disk, via an intervening segment of host memory, called the *buffer*. When an application writes to the storage device, the host lets the required changes accumulate in memory before writing them out to the media in a single operation, a process called *flushing the buffer*. When an application reads from the media, the host reads more data from the media than the application requested and stores it all in memory, in case the application subsequently requests the additional data.

By consolidating a large number of application I/O requests into a smaller number of hardware I/O operations, buffering improves I/O performance and uses storage hardware more efficiently, even when applications send or request data in suboptimal amounts or at inconsistent rates. Compare **direct I/O**.

cartridge

A container for data-storage media, such as magnetic tape or optical media. Also called a *volume*, a *tape*, a *piece of media*, or, loosely, a *VSN*. See **volume**, **volume serial number (VSN)**.

catalog

The Oracle Hierarchical Storage Manager software's record of the removable media volumes in an automated library. Volumes are identified and tracked using a *volume*

serial number. See **volume serial number (VSN)**, **historian**.

client-server

Adjective describing a distributed computer application that divides work between *servers* that specialize in providing files or services and *clients* that request files and services when performing particular tasks.

cloud library

Cloud storage that is accessed and managed as if it were a network-attached tape library containing a set of labeled media volumes. For additional information, see the **`cloud`** (7) man page.

cloud storage

Storage provided as an abstract, network service, without reference to any particular physical implementation or location. Cloud storage supplies users with an agreed level of service rather than a set of defined physical resources. Users and applications store and access data by addressing logical containers rather than physical locations.

A **cloud library** is the Oracle Hierarchical Storage Manager interface to cloud storage.

Cryptographic Framework

The Oracle Solaris Cryptographic Framework, a common store of algorithms and libraries that handle cryptographic requirements for users and applications. The Cryptographic Framework implements the PKCS #11 Cryptographic Token Interface (Cryptoki). See **Public Key Cryptography Standards #11 (Cryptoki)**.

Cryptoki

The PKCS #11 Cryptographic Token Interface. See **Public Key Cryptography Standards #11 (Cryptoki)**.

data device

In a file system, a device or group of devices upon which file data is stored.

Data Integrity Validation (DIV)

Data Integrity Validation, a feature of Oracle StorageTek tape drives that works with the Oracle HSM software to calculate and compare checksums during I/O.

During write operations, Oracle HSM calculates a four-byte checksum for each data block and passes the checksum to the drive along with the data. The tape drive then recalculates the checksum and compares the result to the value supplied by Oracle HSM. If the values agree, the drive writes both the data block and the checksum to tape. Optionally, when the write operation is complete, Oracle HSM can ask the tape drive to rescan the data, recalculate checksums, and compare the results to the checksums stored on the tape.

During read operations, both the drive and Oracle HSM read a data block and its associated checksum from tape. Each recalculates the checksum from the data block and compares the result to the stored checksum. If checksums do not match at any point, the drive notifies Oracle HSM that an error has occurred.

data mover

In an Oracle HSM shared file system, a client that is connected to tape drives and performs tape I/O on behalf of the metadata server. See **distributed I/O**.

DAU

See **disk allocation unit (DAU)**.

device logging

A configurable feature of Oracle HSM that provides specific error information for the hardware devices that support file systems.

device scanner

Software that periodically monitors the presence of all manually mounted removable devices and detects the presence of mounted cartridges that can be requested by users or by other processes.

direct access

Access to files on archival media without preliminary staging to the disk cache. The **-n** (*stage never*) staging attribute marks files for direct access. See **removable media file** and the **stage** (1) man page for more information.

direct-attached automated library

An **automated library** that is connected directly to the host via a SCSI interface. Oracle HSM software can directly control SCSI-attached libraries.

direct I/O

Reading from and writing to a storage device without using memory buffers on the host. Direct I/O can improve performance when transferring large amounts of block-aligned, sequential data. But otherwise, **buffered I/O** generally provides the best results.

directory

A file data structure that points to other files and directories within the file system.

disk allocation unit (DAU)

In QFS file systems, the minimum amount of contiguous space that each I/O operation consumes, regardless of the amount of data written. The disk allocation unit thus determines the minimum number of I/O operations needed when transferring a file of a given size. The DAU should always be a multiple of the block size of the disk device.

The size of the disk allocation unit varies depending upon the QFS device type selected and user requirements. The **md** device type uses dual-allocation units: the DAU is **4** kilobytes for the first eight writes to a file and then a user-specified **16**, **32**, or **64** kilobytes for any subsequent writes, so that small files are written in suitably small blocks, while larger files are written in larger blocks. The **mr** and striped group device types use a DAU that is adjustable in increments of **8** within the range **[8-65528]** kilobytes. Files are thus written in large, uniform blocks that can closely approximate the size of the large, uniformly sized files.

See **block size.**

disk buffer

In Oracle Hierarchical Storage Manager SAM-Remote configurations, the buffer on the SAM-Remote server host that is used for archiving data from the client to the server.

disk cache

The disk-resident portion of an Oracle HSM file system where files are written, modified, and read. New and modified files are copied from the disk cache to archive

media and may eventually be released from the disk. When users subsequently request non-resident files, the files are staged (copied) from the archival media to the disk. Individual disk partitions or an entire disk can be used as disk cache.

disk space threshold

The maximum or minimum level of disk cache utilization, as defined by an administrator. The releaser controls disk cache utilization based on these predefined disk space thresholds. See **high-water mark**, **low-water mark**, and **releaser**.

disk striping

Writing a file across several disks, thereby improving access performance and increasing overall storage capacity. See also **striping**.

distributed I/O

A feature of Oracle Hierarchical Storage Manager that lets the metadata server of a shared QFS file system delegate tape I/O to file system clients that are connected to tape drives. This reduces loads on the server and makes more efficient use of drives and SAN bandwidth. See **data mover**.

DIV

See **Data Integrity Validation (DIV)**.

drive

1. A electromechanical mechanism for transferring data to and from a removable-media volume, such as a magnetic tape cartridge.
2. An electromechanical, magnetic hard disk drive.
3. A solid-state device that emulates a disk drive. See **solid-state device**.

Ethernet

A packet-switched, local area network technology.

extent array

An array within a file's inode that defines the disk location of each data block assigned to the file.

family set

In Oracle HSM and QFS file-system configurations, a group of physical devices that function as a single logical devices, such as a set of data and metadata disks or an automated library and its associated drives.

Fibre Channel

The ANSI standard that specifies high-speed serial communication between devices. Fibre Channel is used as one of the bus architectures in SCSI-3.

file system

A logical structure that organizes data into a hierarchy of directories and files.

file system directives

In the Oracle HSM **`archiver.cmd`** file, archiver and releaser directives that are specific to a particular file system. File system directives follow global directives and include all directives between an **`fs =`** *`filesystem-specifier`* directive and the next **`fs =`**

directive or the end of the file. File system directives override any global directives that may also apply. See the **archiver.cmd** (4) man page for details.

ftp

File Transfer Protocol, a network protocol for transferring files between two hosts. For a more secure alternative, see **sftp**.

global directives

In the Oracle HSM **archiver.cmd** file, archiver and releaser directives that apply to all file systems. Global directives appear before the first **fs =** *filesystem-specifier* directive. See the **archiver.cmd** (4) man page for details.

grace period

In a disk **quota**, the amount of time that the file system allows the total size of files belonging to specified user, group, and/or **admin set**s to exceed the **soft limit** specified in the quota.

HA-COTC

High-Availability Clients Outside the Cluster, the failover configuration for the metadata servers of a shared QFS file system that includes clients.

In an HA_COTC configuration, the file system is shared between active and potential QFS metadata servers and file-system clients. The metadata servers are hosted on a two-node, failover cluster. Clients are not hosted on cluster nodes. Solaris Cluster thus software ensures that the metadata servers remain available so that clients can access metadata and obtain I/O licenses. But clients are not configured for failover and cannot therefore compromise the integrity of the file system following a failure.

HA-QFS

High Availability QFS, the failover configuration that insures that a QFS unshared, standalone file system remains accessible in the event of a host failure.

In an HA-QFS configuration, the file system is configured on both nodes of a two-node cluster managed by Solaris Cluster software. At any given time, only one node mounts the QFS file system. If the node that is mounting the file system fails, the clustering software automatically initiates fail over and re-mounts the file system on the remaining node.

HA-SAM

High-Availability Oracle Hierarchical Storage Manager, the failover configuration for an archiving QFS file system.

In an HA-SAM configuration, Oracle Solaris Cluster software maintains the availability of the file system by insuring that the QFS metadata server and the Oracle Hierarchical Storage Manager application continue to operate even if a server host fails. The file system is shared between active and potential QFS metadata servers hosted on a two-node cluster that is managed by Solaris Cluster software and Data Services.

hard limit

In a quota, the amount of time that the file system allows the total size of files belonging to specified user, group, and/or admin set IDs to exceed the soft limit specified in the quota. See **quota**, **admin set**, **soft limit**.

high-water mark

1. The percentage disk-cache utilization at which Oracle HSM starts the releaser process, deleting previously archived files from disk. A properly configured high-water mark insures that the file system always has enough space available for new and newly staged files. For more information, see the **sam-releaser** (1m) and **mount_samfs** (1m) man pages. Compare **low-water mark**.
2. In a removable media library that is part of an archiving file system, the percentage media-cache utilization at which Oracle HSM starts the recycler process. Recycling empties partially full volumes of current data so that they can replaced by new media or relabeled.

historian

The Oracle HSM historian is a catalog of volumes that have been exported from the automated media libraries defined in the **/etc/opt/SUNWsamfs/mcf** file. By default, it is located at **/var/opt/SUNWsamfs/catalog/historian** on the Oracle HSM file-system host. For additional information, see **catalog** and the **historian** (7) man page.

hosts file

The **hosts.***filesystem-name* file that identifies the hosts that can mount a shared QFS file system. See the **hosts.fs** (4) man page for details.

identity domain

In the Oracle Cloud, the authorization and authentication domain for a single tenant of the multiple-tenancy cloud environment. Within the identity domain, the tenant can define role-based user accounts and make subscribed Cloud services available to the specified users. Each identity domain is identified by a unique Identity Domain ID.

For additional details, see "Oracle Cloud Terminology", in Chapter 1 of *Getting Started with Oracle Cloud*, available at **docs.oracle.com**.

indirect block

A disk block that contains a list of storage blocks. Oracle HSM and QFS file-system metadata can contain up to three levels of indirect blocks. A first-level indirect block contains a list of blocks used for data storage. A second-level indirect block contains a list of first-level indirect blocks. A third-level indirect block contains a list of second-level indirect blocks.

inode

An index node, a 512-byte metadata structure that defines a file for the file system. An inode describes all the attributes associated with a file other than the name. The attributes include ownership, access, permissions, size, and the location of the file.

inode file

In a QFS file system, a metadata file called **.inodes** that contains the inode structures for all files resident in the file system.

Java Cryptography Extension (JCE)

A provider-based application program interface (API) that provides a uniform framework for implementing security features in Java applications. It is part of the Java Developer's Kit (JDK).

kernel

The program that provides basic operating system facilities. The UNIX kernel creates and manages processes, provides functions to access the file system, provides general security, and supplies communication facilities.

key label

A name that uniquely identifies a cryptographic key that is contained within a keystore. See **keystore**.

Key Management Appliance (KMA)

In an **Oracle Key Manager (OKM)** environment, a security-hardened server that manages and provisions encryption keys and authentication of the storage devices in accordance with prescribed policies. KMAs are clustered for high-availability.

Key Management Framework (KMF)

The Oracle Solaris Key Management Framework, a set of tools and application programming interfaces (APIs) for securely creating, exchanging, storing, and using public-key encryption objects, such as X.509 certificates and public/private key pairs. KMF supports both **Key Management Interoperability Protocol (KMIP)** and **Public Key Cryptography Standards #11 (Cryptoki)**.

For full information, see the encryption- and certificate-management information in the *Oracle Solaris Information Library* and the Solaris **`pktool`** (1) and **`kmscfg`** (1) man pages.

Key Management Interoperability Protocol (KMIP)

An extensible protocol that defines message formats for communicating with cryptographic key-management servers. KMIP lets client applications maintain cryptographic keystores on a server and lets the clients encrypt and decrypt data indirectly, without local access to the key.

The **Oracle Key Vault (OKV)** product implements KMIP, and the Solaris **Key Management Framework (KMF)** can create and manage KMIP keystores. See the **`pktool`** (1) man page for details.

For information on a related standard, see **Public Key Cryptography Standards #11 (Cryptoki)**.

Key Management Service (KMS)

A software service that lets client applications remotely create and store cryptographic keys on dedicated cryptographic servers. Clients can encrypt and decrypt data without direct access to the key. The key servers can track and manage all keys under a single, uniform set of policies. See **Public Key Cryptography Standards #11 (Cryptoki)**, **Key Management Interoperability Protocol (KMIP)**, and the **`kmscfg`** (1) man page.

keystore

1. In the Oracle Solaris Cryptographic Framework, persistent storage for token objects that serve as cryptographic keys.
2. Generally, any repository that holds security certificates, public keys, and private keys for encrypting and decrypting information.

LAN

Local area network.

lease

In a shared QFS file system, a function that grants a client permission to perform an operation on a file for a specified period of time. The metadata server issues leases to each client. Leases can be renewed as necessary.

library

See **automated library**.

library catalog

See **catalog**.

libsam

An application programing interface (API) library that lets applications manipulate Oracle Hierarchical Storage Manager operations and files stored in StorageTek QFS file systems. With the **`libsam`** library, software applications that run on the file-system metadata server can access and manipulate file systems using local function calls. See the **`intro_libsam`** (3) and **`rest_libsam`** (3) man pages for details. Compare **libsamrpc**, **rest_libsam**.

libsamrpc

An application programing interface (API) library that lets applications manipulate Oracle Hierarchical Storage Manager operations and files stored in StorageTek QFS file systems. The **`libsamrpc`** library makes remote procedure calls, so calling applications can run on any host on the network. It supports a subset of the **`libsam`** functions. Compare **libsam**, **rest_libsam**.

Linear Tape File System (LTFS)

An open standard for file systems on magnetic tape media. LTFS provides directory and file metadata that let users and applications use data as if it were stored on magnetic or solid-state disk.

local file system

1. A QFS file system that is not shared with other hosts.
2. A file system that is installed on a server for use by the operating system.
3. A file system that is installed on one node of a Solaris Cluster system and is not made highly available.

low-water mark

In an archiving file system, the percentage disk-cache utilization at which Oracle HSM stops the releaser process and stops deleting previously archived files from disk. A properly configured low-water mark insures that the file system retains as many file in cache as possible, for best performance, while making space available for new and newly staged files. For more information, see the **`sam-releaser`** (1m) and **`mount_samfs`** (1m) man pages. Compare **high-water mark**.

LTFS

See **Linear Tape File System (LTFS)**.

LUN

A Logical Unit Number, a logical partition of a physical device that is used as if it were an independent device.

mcf

The Master Configuration File that defines QFS file systems, data and metadata devices, and Oracle HSM archival data devices.

media

Material that stores data. Common storage media include magnetic tape, magnetic disks, solid-state devices, cloud services, and optical disks.

media migration

1. Copying files from one type or generation of archival tape media to a different type or a newer generation of the same type.
2. Copying files from old, worn archival tape media to new, replacement media.
3. A feature of Oracle Hierarchical Storage Manager 6.1 and later that automates the above processes.

metadata

Literally, data about data. In a file system, metadata is information about files and directories. It includes the locations of each file's data on storage media, file attributes such as file type (directory, regular file, character special file, block special file, etc), modification times, ownership, access permissions, and checksums. See **inode**, **indirect block**.

For additional details, see the Oracle HSM **`sls`** (1) and Solaris **`ls`** (1) man pages.

metadata device

In an Oracle HSM high-performance (type **`ma`**) file system, a dedicated storage device type (type **`mm`**) that stores only file system metadata. See the **`mcf`** (4) man page.

You can use solid-state disk, electromechanical magnetic disk, or hardware or software mirrored devices as metadata devices.

metadata server (MDS)

The host that controls Oracle Hierarchical Storage Manager and StorageTek QFS file systems. The metadata server manages the file-system metadata, maintains configuration information for file systems and related processes, such as archiving, participates in file-system I/O, and, in shared configurations, makes the file system available to clients.

Only one metadata server can be *active* at a time. But, in shared configurations, you can configure some or all clients as standby, *potential* metadata servers that can be activated should the active server fail or require disruptive maintenance.

mount point

The directory on which a file system is mounted.

multireader file system

An Oracle HSM file system configuration in which all file system hosts can read files but only one host can write them. For more information, see the **`mount_samfs`** (1m) man page.

nearline storage

Removable media storage that requires robotic mounting before it can be accessed. Nearline storage is usually less expensive than online storage, but it takes somewhat longer to access. See **automated library**.

network-attached automated library

A library that is controlled by a software package supplied by the vendor. A parameter file identifies network-attached libraries to the Oracle HSM software, and a special Oracle HSM media changer daemon provides the interface to the vendor software. Oracle StorageTek ACSLS software controls Oracle StorageTek network-attached libraries. See **automated library**.

NFS

Network File System, a file system that provides transparent access to remote file systems on heterogeneous networks.

OKM

See **Oracle Key Manager (OKM)**.

OKV

See **Oracle Key Vault (OKV)**.

offsite storage

Storage that is remote from the server and is used for disaster recovery.

online storage

Storage that is immediately available. See **disk cache**.

Oracle HSM

1. A common abbreviation for Oracle Hierarchical Storage Manager.
2. An adjective describing a QFS file system that is configured for archiving and managed by Oracle HSM software.

Oracle Solaris Key Management Framework (KMF)

See **Key Management Framework (KMF)**.

Oracle Key Manager (OKM)

Oracle's scalable, device-independent system for securely creating, exchanging, storing, and using storage encryption keys. See also **Key Management Appliance (KMA)**.

Oracle Key Vault (OKV)

A pre-configured, secured software appliance that centrally manages Oracle Advanced Security Transparent Data Encryption (TDE) master keys, other encryption keys, Oracle Wallets, Java Keystores, and credential files for Oracle database and application servers and clients that implement the Key Management Interoperability Protocol.

See **Key Management Interoperability Protocol (KMIP)**.

parameters file

The file that defines a network attached or cloud library for Oracle HSM. In the Oracle HSM master configuration file, mcf, the path to the parameters file is the equipment identifier for the library, **`/etc/opt/SUNWsamfs/family-set-name`**, where **`family-set-name`** is the value of the family set name field in the **`mcf`**.

partition

A portion of a device or a side of a magneto-optical cartridge.

PKCS #11

See **Public Key Cryptography Standards #11 (Cryptoki)**.

pkcs11_kms

An Oracle Solaris implementation of a PKCS #11 Key Management Service that lets Oracle Key Manager clients communicate with remote servers using a private protocol. See **Oracle Key Manager (OKM)**, **Key Management Service (KMS)**, and the **`pkcs11_kms`** (5) and **`kmscfg`** (1M) man pages.

plugin

See **provider**.

potential metadata server

See **metadata server (MDS)**.

preallocation

The process of reserving a contiguous amount of space on the Oracle HSM disk cache for writing a file. Preallocation can be specified only for a file that is size zero. For more information, see the **`setfa`** (1) man page. See **disk cache**.

provider

In the context of the Oracle Solaris Cryptographic Framework, a supplier of cryptographic services that are used by *consumers*, such Oracle HSM and other applications, end users, or kernel operations. Providers include PKCS #11 libraries, kernel cryptographic modules, and hardware accelerators. Providers are also called Cryptographic Framework *plugins*.

pseudo device

A software subsystem or driver with no associated hardware.

Public Key Cryptography Standards #11 (Cryptoki)

A generic application programming interface (API) for working with cryptographic tokens, such as hardware security modules (HSM) and smart cards, public-key cryptography, and storage encryption. The PKCS #11 standard is currently maintained by the OASIS PKCS 11 Technical Committee.

The Solaris **Key Management Framework (KMF)** can create and manage PKCS #11 keystores. See the **`pktool`** (1) man page for details.

For information on a related standard, see **Key Management Interoperability Protocol (KMIP)**.

QFS

1. A QFS file system, an Oracle UNIX file system that offers high performance and high capacity. QFS file systems can be used on their own or with Oracle Hierarchical Storage Manager.
2. The StorageTek QFS product, which includes the file system without the Oracle Hierarchical Storage Manager software.

qfsdump

See **samfsdump (qfsdump)**.

qfsrestore

See **samfsrestore (qfsrestore)**.

quota

The amount of storage resources that specified user, group, or **admin set**s are allowed to consume. See **hard limit** and **soft limit**.

RAID

Redundant Array of Independent Disks, a disk technology that uses several independent disks to reliably store files. Depending on the architecture, it can protect against data loss should one or more disks fail and can provide higher throughput than individual disks.

recovery point

A compressed file that stores a point-in-time backup copy of the metadata for a Oracle HSM file system.

In the event of a data loss—anything from accidental deletion of a user file to catastrophic loss of a whole file system—an administrator can recover to the last known-good state of the file or file system almost immediately by locating the last recovery point at which the file or file system remained intact. The administrator then restores the metadata recorded at that time and either stages the files indicated in the metadata to the disk cache from archival media or, preferably, lets the file system stage files on demand, as users and applications access them. See **samfsdump (qfsdump)**.

recycler

An Oracle HSM utility that reclaims space when an archival tape volume is largely filled with *stale* copies (copies that no longer reflect the current state of the file). The recycler moves any remaining current file copies to other media and then relabels the volume. The relabeled volume can then be over-written with new files.

recycling

The process of moving current files from an archival media volume and relabeling the media for re-use. For details, see the **`sam-recycler`** (1m), **`recycler.cmd`** (1m), and **`recycler.sh`** (1m) man pages.

regular expression

A string of characters in a standardized pattern-matching language that is designed for searching, selecting, and editing other character strings, such as file names and configuration files. For full details of the regular expression syntax used in Oracle HSM file-system operations, see the Solaris **`regex`** and **`regcmp`** man pages.

release priority

The priority according to which the Oracle HSM **releaser** deletes a file from the file **disk cache** once the file has been successfully archived. See the **`sam-releaser`** (1m) man page for details.

releaser

A Oracle HSM component that identifies archived files and releases their disk cache copies, thus making more disk cache space available. The releaser automatically regulates the amount of online disk storage according to high and low thresholds. See the **`sam-releaser`** (1m) and **`releaser.cmd`** (4) man page for details.

remote procedure call

See **RPC**.

removable media file

In an Oracle HSM file system, a special type of user file that can be accessed directly from where it resides on a removable media cartridge, such as magnetic tape or optical disk cartridge. See **direct access**.

REST

Representational state transfer, the style of software architecture pioneered by the World Wide Web. REST emphasizes the roles of components, their interactions, and the ways that they represent data, rather than the internal implementation of components. Typical REST applications communicate via Hypertext Transfer Protocol (HTTP). REST is an alternative to RPC. See **RPC**.

rest_libsam

An application programing interface (API) that lets applications control Oracle Hierarchical Storage Manager operations and access files stored in StorageTek QFS file systems. The **`rest_libsam`** library provides a lightweight **REST** interface that operates over an authenticated HTTPS connection. It is implemented on top of the existing **`libsam`** library and supports a subset of the **`libsam`** functions. Compare **libsam**, **libsamrpc**.

robot

An **automated library** component that moves cartridges between storage slots and drives. Also called a **transport**.

round-robin

A data access method in which entire files are written to logical disks in a sequential fashion. When a single file is written to disk, the entire file is written to the first logical disk. The second file is written to the next logical disk, and so on. The size of each file determines the size of the I/O. See also **disk striping** and **striping**.

RPC

Remote procedure call, a mechanism that lets a client application execute subroutines that run under an independent server application.

SAM

A common abbreviation for Storage Archive Manager, the former name of the Oracle Hierarchical Storage Manager product.

SAM-Remote

An Oracle HSM client-server configuration that lets an Oracle HSM metadata server access an automated tape library that is controlled by another Oracle HSM metadata server. The client is configured with pseudodevices that represent the devices that the server makes available and uses a specified subset of the archive media on the server.

SAM-QFS

1. A common abbreviation for older versions of the Oracle Hierarchical Storage Manager product.
2. An adjective describing a QFS file system that is configured for archiving and managed by Oracle HSM software.

samfsdump (qfsdump)

An Oracle HSM command that backs up file system metadata to a dump file. See **recovery point**.

If the Oracle Hierarchical Storage Manager packages are not installed, the command is called **qfsdump**.

samfsrestore (qfsrestore)

A program that restores inode and directory information from a recovery point. See **samfsdump (qfsdump)**, **recovery point**.

SAN

Storage Area Network.

SAS

Serial-Attached SCSI.

SC-RAC

Solaris Cluster-Oracle Real Application Cluster (SC-RAC), a high-availability Oracle Database solution that use QFS file systems.

In an SC-RAC solution, Oracle RAC software coordinates I/O requests, distributes workload, and maintains a single, consistent set of database files for multiple Oracle Database instances running on the nodes of a cluster. In the SC-RAC configuration, Oracle Database, Oracle Real Application Cluster (RAC), and QFS software run on two or more of the nodes of a cluster managed by Oracle Solaris Cluster software. One node is configured as the metadata server (MDS) of a QFS shared file system. The remaining nodes are configured as potential metadata servers that share the file system as clients. If the active metadata server node fails, Solaris Cluster software automatically activates a potential metadata server on a healthy node and initiates failover. Since the QFS file system is shared and already mounted on all nodes, access to the data remains uninterrupted.

SCSI

Small Computer System Interface, an electrical communication specification commonly used for peripheral devices such as disk and tape drives and automated libraries.

seeking

Moving the read/write heads of a disk device from one disk location to another during random-access I/O operations.

sftp

Secure File Transfer Protocol, a secure implementation of **ftp**. See **ssh**.

shared hosts file

When you create a shared file system, the system copies information from the hosts file to the shared hosts file on the metadata server. You update this information when you issue the **samsharefs -u** command

Small Computer System Interface

See **SCSI**.

soft limit

In a quota, the maximum amount of storage space that a specified user, group, and/or admin set IDs can fill for an indefinite period. Files can use more space than the soft limit allows, up to the hard limit, but only for a short grace period defined in the quota. See **grace period**, **hard limit**, **quota**, **admin set**.

solid-state device

A storage device that uses electronically rewritable, non-volatile, NAND flash memory as the storage medium, such as a SAS-attached, solid-state disk drive (SSD).

Solid-state drives can provide significantly higher inputs and outputs per second (IOPS) and significantly lower latency compared to traditional magnetic hard drives. They are thus particularly good choices for use as the metadata devices of Oracle Hierarchical Storage Manager and StorageTek QFS, high-performance, **ma** file systems.

ssh

Secure Shell, an encrypted network protocol that allows secure, remote command-line login and command execution.

staging

In Oracle HSM file systems, the process of copying an archived file that is no longer resident in the disk cache from archival storage back to the disk cache. See the **staging** (1) and **stager.cmd** (4) man pages, **disk cache**, and **associative staging** for additional information.

Storage Archive Manager

The former name of the Oracle Hierarchical Storage Manager product.

storage slots

In an automated library, the storage bays that hold media cartridges that are not mounted in drives.

stripe size

During striped device access, the number of disk allocation units (DAUs) that a QFS file system writes before moving to the next device in the stripe, as specified by the **stripe=** mount option.

striped group

In a QFS file system, a collection of devices configured as a single logical device of type **gXXX**. See the **mcf** (4) man page for additional information.

striping

Writing files to multiple devices in parallel, so that each file is spread across all the devices.

QFS file systems support two types of striping:

- *Hard striping* is a permanent feature of the file system that you enable when you specify striped group (type **gXXX**) devices in the Master Configuration File (**mcf**) entries that define the file system.
- *Soft striping* is an optional feature that enable or disable when you mount the file system with the **stripe=** mount parameter.

Compare **round-robin**.

SUNW.hasam

A Solaris Cluster Data Services resource type that supports failover for the Oracle Hierarchical Storage Manager application. **SUNW.hasam** is included with the Oracle HSM software. See HA-SAM.

SUNW.HAStoragePlus

A Solaris Cluster Data Services resource type that manages failover of a server's local storage, so that critical state and dynamic configuration information remains available. **`SUNW.HAStoragePlus`** is included in the Solaris Cluster software as a standard resource type. See HA-QFS, HA-SAM.

SUNW.qfs

A Solaris Cluster Data Services resource type that supports failover for the metadata servers of a high-availability, StorageTek QFS file system. **`SUNW.qfs`** is included with the Oracle Hierarchical Storage Manager and StorageTek QFS software. See HA-QFS, HA-SAM, and HA-COTC.

superblock

A data structure in the file system that defines the basic parameters of the file system. The superblock is written to all partitions in the storage family set and identifies the partition's membership in the set.

tar

Tape archive. A standard file and data recording format used for archive images.

TCP/IP

Transmission Control Protocol/Internet Protocol. The internet protocols responsible for host-to-host addressing and routing, packet delivery (IP), and reliable delivery of data between application points (TCP).

timer

Quota software that keeps track of the period starting when a user reaches a soft limit and ending when the hard limit is imposed on the user.

transport

See **robot**.

`vfstab` file

The **`vfstab`** file contains mount options for the file system. Mount options specified on the command line override those specified in the **`/etc/vfstab`** file, but mount options specified in the **`/etc/vfstab`** file override those specified in the **`samfs.cmd`** file.

volume

1. On storage media, a single, accessible, logical storage area, usually addressed by a volume serial number (VSN) and/or volume label. Storage disks and magnetic tape cartridges can hold one or more volumes. For use, volumes are *mounted* on a file system at a specified mount point. See **volume serial number (VSN)**, **mount point**.
2. A magnetic tape cartridge that holds a single logical volume. See **cartridge**.
3. On a random-access disk device, a file system, directory or file that is configured and used as if it were a sequential-access, removable-media cartridge, such as a tape.

volume overflow

A capability that enables the system to span a single file over multiple **volume**s. Volume overflow is useful for sites using very large files that exceed the capacity of their individual cartridges.

volume serial number (VSN)

1. A serial number assigned to a tape or disk storage volume. A volume serial number can consist of up to six uppercase, alphanumeric characters, must start with a letter, and must identify the volume uniquely within a given context, such a tape library or partition. The volume serial number is written on the volume label.
2. Loosely, a specific storage volume, especially a removable media cartridge. See **cartridge**, **volume**.

WORM

Write-Once-Read-Many. A storage classification for media that can be written only once but read many times.