Oracle® Hierarchical Storage Manager and StorageTek QFS Software

Installation and Configuration Guide

Release 6.1.4

E78138-07

July 2019

Oracle Hierarchical Storage Manager and StorageTek QFS Software Installation and Configuration Guide, Release 6.1.4

E78138-07

Contents

Preface

This document addresses the needs of system administrators, storage and network administrators, and service engineers who are tasked with installing and configuring 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.
- Bold type represents graphical user interface elements.
- `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 a Linux command shell prompt.

Available Documentation

The *Oracle Hierarchical Storage Manager and StorageTek QFS Software Installation and Configuration Guide* is part of the multivolume *Oracle HSM Customer Documentation Library*, available from **`docs.oracle.com/en/storage`**.

Additional, supporting documentation can be found at the following online locations:

- the Oracle HSM wiki:

  `https://community.oracle.com/community/server_%26_storage_systems/systems-io/oracle-tape-storage/oracle_HSM`

- the Oracle HSM forum on the My Oracle Support Community:

  `https://community.oracle.com/community/support/oracle_sun_technologies/sam_qfs_storage_archive_manager_and_sun_qfs`

The above resources provide important product alerts and support information. Oracle recommends that you subscribe to the corresponding news feeds so that you receive notice when new information becomes available.

Documentation for the Oracle Solaris operating system, the Solaris Cluster software, and Solaris Cluster Data Services (HA-NFS, HA-SAMBA, etc.) is available at **`http://docs.oracle.com/en/operating-systems/`**.

1 Deploying Oracle HSM Solutions

Deploying Oracle Hierarchical Storage Manager and StorageTek QFS Software (Oracle HSM) is a straightforward process. You install software packages, edit a couple of configuration files, run a couple of commands, and then mount and use the new file system(s). Nonetheless, Oracle HSM offers a wide range of options and tuning parameters. These extra features can let you address almost any special need.

Accordingly, this document is designed to guide you through a deployment of an Oracle HSM deployment that closely follows detailed solution requirements. It starts from the workings, installation, and configuration of basic QFS and Oracle HSM file systems. These file systems will either meet all of your requirements on their own or form the foundation for a more specialized solution. Once the basics are in place, you can then branch to procedures for configuring additional features that support particular environments and specialized business needs. You will carry out the following core tasks:

- Configure hardware and operating system software to meet requirements.
- Configure the basic QFS and/or Oracle HSM file systems required, accepting defaults wherever possible.
- Configure any additional Oracle HSM features demanded by requirements.
- Back up your finished configuration and hand it over for testing and production use.

Throughout the planning and deployment process, remember that QFS and Oracle HSM 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 that is implemented on a mix of disk arrays, tape libraries, and public and private storage clouds as if it were an ordinary local file system installed on a single hard 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. Overly complex implementations and overzealous micromanagement of file systems and storage resources thus undermine key goals of an Oracle HSM deployment and may impair performance, capacity utilization, and data protection.

The remainder of this introduction provides brief, descriptive overviews of QFS file systems and Oracle HSM archiving file systems. Basic familiarity with this information makes it easier to understand the purpose of subsequent configuration steps.

QFS File Systems

QFS file systems let you combine fully optimized, custom storage solutions with standard UNIX interfaces. Internally, they manage physical storage devices to meet exacting, often highly specialized performance requirements. But they present themselves to external users, applications, and the operating system as ordinary UNIX file systems. So you can meet specialized performance and data-protection requirements using a complex range of storage hardware and still insure straightforward integration with existing applications and business processes.

QFS file systems manage their own physical storage using an integral QFS volume manager. QFS software organizes standard physical storage devices into highly optimized logical devices that remain fully compatible with standard interfaces. The software encapsulates special features and customizations so that they remain hidden from the operating system and applications. To the latter, the QFS software presents a logical *family-set device* that processes I/O requests like a single disk, via standard, Solaris device drivers. This combination of standards compliance and tunability distinguishes QFS from other file systems.

The remainder of this section starts with a brief discussion of QFS defaults and I/O performance-tuning, and then describes the core tools that let you control the I/O behavior of the file systems that you create:

- Flexible disk allocation units and logical device types let you match the sizes of reads and writes to the sizes of files.
- Striped and round-robin file-allocation methods let you control how file I/O interacts with devices.
- Fully configurable storage allocation and integrated volume management let you control how file systems interact with underlying physical storage.
- A choice of general-purpose and high-performance file systems gives you the option of performing data and metadata I/O on separate devices.

QFS Defaults and I/O Performance-Tuning Objectives

Disk I/O (input/output) involves CPU-intensive operating system requests and time-consuming mechanical processes. So I/O performance tuning focuses on minimizing I/O-related system overhead and keeping the mechanical work to the absolute minimum necessary for transferring a given amount of data. This means reducing both the number of separate I/Os per data transfer (and thus the number of operations that the CPU performs) and minimizing *seeking* (repositioning of read/write heads) during each individual I/O. The basic objectives of I/O tuning are thus as follows:

- Read and write data in blocks that divide into the average file size evenly.
- Read and write large blocks of data.
- Write blocks in units that align on the 512-byte sector boundaries of the underlying media, so that the disk controller does not have to read and modify existing data before writing new data.
- Queue up small I/Os in cache and write larger, combined I/Os to disk.

Oracle HSM default settings provide the best overall performance for the range of applications and usage patterns typical of most general-purpose file systems. But when necessary, you can adjust the default behavior to better suit the types of I/O that your applications produce. You can specify the size of the minimum contiguous read or write. You can optimize the way in which files are stored on devices. You can choose between from file systems optimized for general use or for high-performance.

Disk Allocation Units and Logical Device Types

File systems allocate disk storage in blocks of uniform size. This size—the disk allocation unit (DAU)—determines the minimum amount of contiguous space that each I/O operation consumes, regardless of the amount of data written, and the minimum number of I/O operations needed when transferring a file of a given size. If the block size is too large compared to the average size of the files, disk space is wasted. If the block size is too small, each file transfer requires more I/O operations, hurting performance. So I/O performance and storage efficiency are at their highest when file sizes are even multiples of the basic block size.

For this reason, the QFS software supports a range of configurable DAU sizes. When you create a QFS file system, you first determine the average size of the data files that you need to access and store. Then you specify the DAU that divides most evenly into the average file size.

You start by selecting the QFS device type that is best suited to your data. There are three types:

- **`md`** devices
- **`mr`** devices
- **`g`***XXX* striped-group devices (where *XXX* is an integer in the range **`[0-127]`**.

When file systems will contain a predominance of small files or a mix of small and large files, **`md`** devices are usually the best choice. The **`md`** device type uses a flexible, dual-allocation scheme. When writing a file to the device, the file system uses a small DAU of **`4`** kilobytes for the first eight writes. Then it writes any remaining data using a large, user-selected DAU of **`16`**, **`32`**, or **`64`** kilobytes. Small files are thus written in suitably small blocks, while larger files are written in larger blocks tailored to their average size.

When file systems will contain a predominance of large and/or uniformly sized files, **`mr`** devices may be a better choice. The **`mr`** device type uses a DAU that is adjustable in increments of 8 kilobytes within the range **`[8-65528]`** kilobytes. Files are written in large, uniform blocks that closely approximate the average file size, thus minimizing read-modify-write overhead and maximizing performance.

Striped groups are aggregates of up to 128 devices that are treated as a single logical device. Like the **`mr`** device type, striped groups use a DAU that is adjustable in increments of 8 kilobytes within the range **`[8-65528]`** kilobytes. The file system writes data to the members of a striped group in parallel, one DAU per disk. So the aggregate write can be very large. This makes striped groups potentially useful in applications that must handle extremely large data files.

File Allocation Methods

By default, unshared QFS file systems use striped allocation, and shared file systems use round-robin allocation. But you can change allocation when necessary. Each approach has advantages in some situations.

Striped Allocation

When striped allocation has been specified, the file system allocates space in parallel, on all available devices. The file system segments the data file and writes one segment to each device. The size of each segment is determined by the *stripe width*—the number of DAUs written per device—times the number of devices in the family set. Devices may be **md** disk devices, **mr** disk devices, or striped groups.

Striping generally increases performance because the file system reads multiple file segments concurrently rather than sequentially. Multiple I/O operations occur in parallel on separate devices, thus reducing the seek overhead per device.

However, striped allocation can produce significantly more seeking when multiple files are being written at once. Excessive seeking can seriously degrade performance, so you should consider round-robin allocation if you anticipate simultaneous I/O to multiple files.

Round-Robin Allocation

When round-robin allocation has been specified, the file system allocates storage space serially, one file at a time and one device at a time. The file system writes the file to the first device that has space available. If the file is larger than the space remaining on the device, the file system writes the overage to the next device that has space available. For each succeeding file, the file system moves to the next available device and repeats the process. When the last available device has been used, the file system starts over again with the first device. Devices may be **md** disk devices, **mr** disk devices, or striped groups.

Round-robin allocation can improve performance when applications perform I/O to multiple files simultaneously. It is also the default for shared QFS file systems (see "Accessing File Systems from Multiple Hosts Using Oracle HSM Software" on page 8-1 and the **mount_samfs** man page for more information on shared file systems).

Storage Allocation and Integrated Volume Management

Unlike UNIX file systems that address only one device or portion of a device, QFS file systems do their own volume management. Each file-system handles the relationships between the devices that provide its physical storage internally and then presents the storage to the operating system as a single *family set*. I/O requests are made via standard, Solaris device-driver interfaces, as with any UNIX file system.

File System Types

There are two types of QFS file-system. Each has its own advantages:

General-Purpose ms File Systems

QFS **ms** file systems are the simplest to implement and are well suited for most common purposes. They store the file-system metadata with the file data, on the same dual-allocation, **md** disk devices. This approach simplifies hardware configuration and meets most needs.

High-Performance ma File Systems

QFS **ma** file systems can improve data transfer rates in demanding applications. These file systems store metadata and data separately, on dedicated devices. Metadata is kept on **mm** devices, while data is kept on a set of **md** disk devices, **mr** disk devices, or striped groups. As a result, metadata updates do not contend with user and application I/O, and device configurations do not have to accommodate two different kinds of I/O workload. For example, you can place your metadata on RAID-10 mirrored disks for high redundancy and fast reads and keep the data on a more space-efficient, RAID-5 disk array.

Oracle HSM Archiving File Systems

Archiving file systems combine one or more QFS **ma**- or **ms**-type file systems with archival storage and Oracle Hierarchical Storage Manager software. The Oracle HSM software automatically copies modified files from the file system's primary disk cache to less expensive media that is optimized for archival storage. The software manages the copies as an integral part of the file system. So the file system offers both continuous data protection and the ability to flexibly and efficiently store extremely large files that might otherwise be too expensive to store on disk or solid-state devices.

Archival media can include a mix of secondary disk volumes, removable tape volumes, and Oracle Storage Cloud or private cloud volumes. Tape and cloud volumes can be encypted, making them ideal for secure, off-site storage (for information on the encryption capabilities of your tape drives, see the vendor documentation). When enabled, Oracle HSM cloud encryption feature encrypts data files using an AES 256 symmetric-key cypher before writing them to cloud volumes (for details, see the **cloud** (7) man page).

A properly configured, Oracle HSM file system provides continuous data protection without separate backup applications. The software backs up file data as they change, in accordance with user-defined policies. You can maintain up to four archival copies of each file, using a mix of media types and local, remote, and cloud storage resources. Archival copies are stored in standard, POSIX-compliant **tar** (tape archive) files.

When necessary, you can quickly restore individual files, an Oracle HSM file system, or an entire Oracle HSM server and primary storage system using saved file-system metadata and the archived file data. The Oracle HSM software records the location of all copies of file data in the file-system metadata. I/O errors are dynamically detected and corrected, so that the metadata remains consistent at all times. If the primary copy of a file’s data is lost or corrupted, the file system can seamlessly supply a replacement copy from the archive. If you back up this file-system metadata regularly—using the Oracle HSM command samfsdump—and save the backup metadata files in a secure location, you can restore lost file systems without copying data back to primary storage and without time-consuming integrity checks—a major consideration when hundreds of thousands of files and petabytes of data are being stored. The restored file system is ready for use as soon as you restore the metadata to the primary storage. As users and applications access files in the file system, Oracle HSM copies the required data from the archive to the replacement disk and updates the file-system metadata accordingly. Since files are restored to disk only as requested, the recovery process makes efficient use of network bandwidth and has minimal impact on normal operations.

This ability to simultaneously manage files on high-performance, primary disk or solid-state media and on lower-cost, higher-density secondary disk, tape, or Cloud-resident media makes Oracle HSM file systems ideal for economically storing unusually large and/or little used files. Very large, sequentially accessed data files, like satellite images and video files, can be stored exclusively on magnetic tape. When users or applications access a file, the file system automatically stages it back to disk or reads it into memory directly from tape, depending on the chosen file configuration. Records that are retained primarily for historical or compliance purposes can be stored hierarchically, using the media that is best aligned with user access patterns and cost constraints at a given point in the life of the file. Initially, when users still occasionally access a file, you can archive it on lower-cost, secondary disk devices. As demand diminishes, you can maintain copies only on tape or optical media. Yet, when a user needs the data—in response to a legal discovery or regulatory process, for example—the file system can automatically stage the required material to primary disk with minimal delay, much as if it had been there all along.

For legal and regulatory purposes, Oracle HSM archival file systems can be WORM-enabled. WORM-enabled file systems support default and customizable file-retention periods, data and path immutability, and subdirectory inheritance of WORM settings. Long-term data integrity can be monitored using manual and/or automated media validation.

Four, basic, Oracle HSM processes configure, manage, and maintain archiving file systems:

- Archiving
- Staging
- Releasing
- Recycling.

Archiving

The archiving process copies files from a file system to *archival media* that are reserved for storing copies of active files. Archival media may include removable media volumes, such as magnetic tape cartridges, and/or one or more file systems that reside on magnetic disk or solid-state storage devices. Archival file copies may provide backup redundancy for the active files, long-term retention of inactive files, or some combination of both.

In an Oracle HSM archiving file system, the active, online files, the archival copies, and the associated storage resources form a single, logical resource, the *archive set*. Every active file in an archiving file system belongs to exactly one archive set. Each archive set may include up to four archival copies of each file plus policies that control the archiving process for that archive set.

The archiving process is managed by a UNIX daemon (service), **`sam-archiverd`**. The daemon schedules archiving activities and calls the processes that perform the required tasks, **`archiver`**, **`sam-arfind`**, and **`sam-arcopy`**.

The **`archiver`** process reads the archiving policies in an editable configuration file, **`archiver.cmd`**, and sets up the remaining archiving processes as specified. Directives in this file control the general behavior of the archiving processes, define the archive sets by file system, and specify the number of copies made and the archival media used for each.

The **`sam-archiverd`** daemon then starts a **`sam-arfind`** process for each currently mounted file system. The **`sam-arfind`** process scans its assigned file system for new files, modified files, renamed files, and files that are to be re-archived or unarchived. By default, the process scans continuously for changes to files and directories, since this offers the best overall performance. But, if you must maintain compatibility with older, StorageTek Storage Archive Manager implementations, for instance, you can edit the archive set rules in the **`archiver.cmd`** file to schedule scanning using one of several methods (see the **`sam-archiverd`** man page for details).

Once it identifies candidate files, **`sam-arfind`** identifies the archive set that defines the archiving policies for the file. The **`sam-arfind`** process identifies the archive set by comparing the file's attributes to the selection criteria defined by each archive set. These criteria might include one or more of the following file attributes:

- the directory path to the file and, optionally, a regular expression that matches one or more of the candidate file names
- a specified user name that matches the owner of one or more candidate files
- a specified group name that matches the group associated with the file

- a specified minimum file size that is less than or equal to the size of the candidate file
- a specified maximum file size that is greater than or equal to the size of the candidate file.

Once it has located the correct archive set and the corresponding archiving parameters, **`sam-arfind`** checks whether the **`archive age`** of the file equals or exceeds the threshold specified by the archive set. The archive age of a file is the number of seconds that have elapsed since the file was created, last modified (the default), or last accessed. If the archive age meets the age criteria specified in the policy, **`sam-arfind`** adds the file to the *archive request* queue for the archive set and assigns it a priority. Priorities are based on rules specified in the archive set and on factors such as the number of archive copies that already exist, the size of the file, any outstanding operator requests, and any other operations that depend on the creation of an archive copy.

Once **`sam-arfind`** has identified the files that need archiving, prioritized them, and added them to archive requests for each archive set, it returns the requests to the **`sam-archiverd`** daemon. The daemon **`composes`** each archive request. It arranges data files into archive files that are sized so that media is efficiently utilized and files are efficiently written to and, later, recalled from removable media. The daemon honors any file-sorting parameters or media restrictions that you set in the **`archiver.cmd`** file (see the **`archiver.cmd`** man page for details), but be aware that restricting the software's ability to select media freely usually reduces performance and media utilization. Once the archive files have been assembled, **`sam-archiverd`** prioritizes the archive requests so that the copy process can transfer the largest number of files in the smallest number of mount operations (see the scheduling section of the **`sam-archiverd`** man page for details). Then **`sam-archiverd`** schedules copy operations so that, at any given time, they require no more than the maximum number of drives allowed by the archive set policies and/or the robotic library.

Once the archive requests are scheduled, **`sam-archiverd`** calls an instance of the **`sam-arcopy`** process for each archive request and drive scheduled. The **`sam-arcopy`** instances then copy the data files to archive files on archival media, update the archiving file system's metadata to reflect the existence of the new copies, and update the archive logs.

When the **`sam-arcopy`** process exits, the **`sam-archiverd`** daemon checks the archive request for errors or omissions caused by read errors from the cache disk, write errors to removable media volumes, and open, modified, or deleted files. If any files have not been archived, **`sam-archiverd`** recomposes the archive request.

The **`sam-arfind`** and **`sam-arcopy`** processes can use the **`syslog`** facility and **`archiver.sh`** to create a continuous record of archiving activity, warnings, and informational messages. The resulting archiver log contains valuable diagnostic and historical information, including a detailed record of the location and disposition of every copy of every file archived. So, during disaster recovery, for example, you can often use an archive log to recover missing data files that would otherwise be irrecoverable (for details, see the *Oracle Hierarchical Storage Manager and StorageTek QFS Software File System Recovery Guide* in the *Customer Documentation Library*). File-system administrators enable archiver logging and define log files using the **`logfile=`** directive in the **`archiver.cmd`** file. For more information about the log file, see the **`archiver.cmd`** man page.

Staging

The staging process copies file data from archival storage back into the primary disk cache. When an application tries to access an *offline file*—a file that is not currently available in primary storage—an archive copy is automatically *staged*—copied back to primary disk. The application can then access the file quickly, even before complete data is written back to disk, because the read operation tracks along directly behind staging. If a media error occurs or if a specific media volume is unavailable, the staging process automatically loads the next available archive copy, if any, using the first available device. Staging thus makes archival storage transparent to users and applications. All files appear to be available on disk at all times.

The default staging behavior is ideal for most file systems. However, you can alter the defaults by inserting or modifying directives in a configuration file, **`/etc/opt/SUNWsamfs/stager.cmd`**, and you can override these directives on a per-directory or per-file basis from the command line. To access small records from large files, for example, you might choose to access data directly from the archival media without staging the file. Or you might stage a group of related files whenever any one of them is staged, using the *associative* staging feature. See the **`stage`** and **`stager.cmd`** man pages for details.

Releasing

The releasing process frees up primary disk cache space by deleting the online copies of previously archived files that are not currently in use. Once a file has been copied to archival media, such as a disk archive or tape volume, it can be staged when and if an application accesses it. So there is no need to retain it in the disk cache when space is needed for other files. By deleting unneeded copies from disk cache, releasing insures that primary cache storage is always available for newly created and actively used files, even if the file system grows without any corresponding increase in primary storage capacity.

Releasing occurs automatically when the cache utilisation exceeds the *high-water mark* and remains above a *low-water mark*, two configurable thresholds that you set when you mount an archiving file system. The high-water mark insures that enough free space is always available, while the low-water mark insures that a reasonable number of files are always available in cache and that media mount operations are thus kept to the minimum necessary. Typical values are 80% for the high value and 70% for the low.

Releasing by water mark using the default behavior is ideal for most file systems. However, you can alter the defaults by modifying or adding directives in a configuration file, **`/etc/opt/SUNWsamfs/releaser.cmd`**, and you can override them on a per-directory or per-file basis from the command line. You can, for example, partially release large, sequentially accessed files, so that applications can start reading a part of the file that is always retained on disk while the remainder stages from archival media. See the **`release`** and **`releaser.cmd`** man pages for details.

Recycling

The recycling process frees up space on archival media by deleting archive copies that are no longer in use. As users modify files, the archive copies associated with older versions of the files eventually expire. The recycler identifies the media volumes that hold the largest proportion of expired archive copies. If the expired files are stored on an archival disk volume, the recycler process deletes them. If the files reside on removable media, such as a tape volume, the recycler re-archives any unexpired copies that remain on the target volume to other media. It then calls an editable script, **`/etc/opt/SUNWsamfs/scripts/recycler.sh`**, to relabel the recycled volume, export it from the library, or perform some other, user-defined action.

By default, recycling process does not run automatically. You can configure the Solaris **`crontab`** file to run it at a convenient time. Or you can run it as needed from the command line using the command **`/opt/SUNWsamfs/sbin/sam-recycler`**. To modify default recycling parameters, edit the file **`/etc/opt/SUNWsamfs/archiver.cmd`** or create a separate **`/etc/opt/SUNWsamfs/recycler.cmd`** file. See the corresponding man pages for details.

2 Configuring Host Systems

Configure host operating systems for Oracle Hierarchical Storage Manager and StorageTek QFS Software before proceeding further with installation and configuration. The chapter outlines the following tasks:

- Configuring Oracle Solaris hosts for Oracle HSM
- Configuring Linux hosts as Oracle HSM clients (if required)
- Configuring a key manager for cloud data encryption (if required)

Configuring Oracle Solaris Hosts for Oracle HSM

To configure Solaris hosts for use with Oracle HSM software and QFS file systems, carry out the following tasks:

- Install the latest operating system updates.
- Tune Solaris system and driver parameters for anticipated file system I/O.
- If you plan to use encrypted cloud storage, install the required supplementary software.

Install the Latest Operating System Updates

If possible, always install the latest patches and updates for the Solaris operating system. If you need to use the latest features available in Oracle Hierarchical Storage Manager and StorageTek QFS Software Release 6.1.4, you must have Oracle Solaris 11 operating system software installed on all Solaris hosts. For full information on the minimum recommended operating system releases for use with software, consult the release notes and **`support.oracle.com`**.

For installation and update instructions for the chosen version of Solaris, consult the installation and administration documents in the corresponding customer documentation library, the Oracle Technical Network (OTN), and the knowledgebase on **`support.oracle.com`**. If you are new to the Image Packaging System (IPS), the following OTN articles may prove especially helpful:

- *Introducing the Basics of Image Packaging System (IPS) on Oracle Solaris 11* by Glynn Foster (November 2011)
- *How to Update Oracle Solaris 11 Systems From Oracle Support Repositories* by Glynn Foster (March 2012)
- *More Tips for Updating Your Oracle Solaris 11 System from the Oracle Support Repository* by Peter Dennis (May 2012).

Tune Solaris System and Driver Parameters for Anticipated File System I/O

End-to-end input/output (I/O) performance through a system is highest when the operating system, drivers, file systems, and applications transfer data in units that do not have to be fragmented and re-cached unnecessarily. So set up Solaris for the largest data transfers that your the applications and file systems are likely to make. Proceed as follows:

1. Log in to the Oracle HSM file-system host as **`root`**.

```
root@solaris:~#
```

2. Make a backup copy of the **`/etc/system`** file, and then open **`/etc/system`** in a text editor.

   In the example, we use the **`vi`** editor.

```
root@solaris:~# cp /etc/system /etc/system.backup
root@solaris:~# vi /etc/system
*ident  "%Z%%M% %I%     %E% SMI" /* SVR4 1.5 */
* SYSTEM SPECIFICATION FILE
...
```

3. In the **`/etc/system`** file, set **`maxphys`**, the size of the largest physical I/O request that any driver can process as a single unit, equal to the largest data transfers that your applications and file systems will make. Enter a line of the form **`set maxphys = 0x`***`value`*, where *`value`* is a hexadecimal number representing a number of bytes. Then save the file and close the editor.

   Drivers break up requests that exceed **`maxphys`** into **`maxphys`**-sized fragments. The default value can vary depending on the operating system release, but it is typically around 128 kilobytes. In the example, we set **`maxphys`** to **`0x800000`** (8,388,608 bytes or 8 megabytes):

```
root@solaris:~# vi /etc/system
*ident  "%Z%%M% %I%     %E% SMI" /* SVR4 1.5 */
* SYSTEM SPECIFICATION FILE
...
set maxphys = 0x800000
:wq
root@solaris:~#
```

4. Open the **`/kernel/drv/sd.conf`** file in a text editor.

   In the example, we use the **`vi`** editor:

```
root@solaris:~# vi /kernel/drv/sd.conf
# Copyright (c) 1991, 2010, Oracle and/or its affiliates. All rights reserved.
name="sd" class="scsi" target=0 lun=0;
name="sd" class="scsi" target=1 lun=0;
...
# Associate the driver with devid resolution.
ddi-devid-registrant=1;
```

5. In the **`/kernel/drv/sd.conf`** file, set **`sd_max_xfer_size`**, the size of the largest data transfer that the SCSI disk (**`sd`**) driver can process, to the value that you set for **`maxphys`**. Enter a line of the form **`sd_max_xfer_size=0x`***`value`***`;`**, where *`value`* is a hexadecimal number representing a number of bytes. Save the file, and close the editor.

The default is **0x100000** (1048576 bytes or one megabyte). In the example, we add a comment and set **sd_max_xfer_size** to **0x800000** (8,388,608 bytes or 8 megabytes):

```
...
# Associate the driver with devid resolution.
ddi-devid-registrant=1;
# Set SCSI disk maximum transfer size
sd_max_xfer_size=0x800000;
:wq
root@solaris:~#
```

6. Open the **/kernel/drv/ssd.conf** file in a text editor.

   In the example, we use the **vi** editor.

```
root@solaris:~# vi /kernel/drv/ssd.conf
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
name="ssd" parent="sf" target=0;
name="ssd" parent="fp" target=0;
...
name="ssd" parent="ifp" target=127;
```

7. In the **/kernel/drv/ssd.conf** file, set **ssd_max_xfer_size**, the size of the largest data transfer that the Fibre Channel disk (**ssd**) driver can process to the value that you set for **maxphys**. Enter a line of the form **ssd_max_xfer_size=0x***value***;**, where *value* is a hexadecimal number representing a number of bytes. Then save the file and close the editor.

   The default is **0x100000** (1048576 bytes or one megabyte). In the example, we add a comment and set **ssd_max_xfer_size** to **0x800000** (8,388,608 bytes or 8 megabytes):

```
...
name="ssd" parent="ifp" target=127;
# Set Fibre Channel disk maximum transfer size
ssd_max_xfer_size=0x800000;
:wq
root@solaris:~#
```

8. Restart the system. Use the command **init 6**.

```
root@solaris:~# init 6
```

9. If you plan to use encryption with cloud storage, install the additional required software now.

10. If you are preparing a solution that includes additional Solaris hosts, repeat the tasks specified in "Configuring Oracle Solaris Hosts for Oracle HSM" on page 2-1 until all Solaris hosts have been configured.

11. If you are preparing a solution that includes one or more Linux clients, go to "Configuring Linux Hosts for Oracle HSM Clients".

12. Otherwise, go to "Configuring Storage Hosts and Devices" on page 3-1.

Enable Oracle HSM Cloud Libraries

Cloud libraries (equipment type cr) and cloud media volumes (media type cl) are the Oracle HSM interface to public and private storage clouds. Storage clouds are abstract, network services that provide an agreed level of service rather than a set of defined physical resources. Oracle HSM cloud libraries and media let you use cloud resources in the same way that you use a removable media library.

If you intend to make use of the Oracle HSM cloud library, carry out the following preliminary tasks:

- Install Java Development Kit 7.
- If you plan on using the cloud drive encryption feature, enable cloud encryption by installing cryptography software and configuring a key-management solution.

Install Java Development Kit 7

If you intend to make use of the Oracle HSM cloud library (cr) equipment type, you must install Java Development Kit 7 (JDK 7) on the Oracle HSM metadata server. JDK 7 is the most recent release of the software to include compatible, 32-bit executables. If you have an active Oracle support agreement, you can obtain the most recent version, updated with all current security enhancements.

To make sure that the required software is installed, carry out the tasks listed below:

- Uninstall older versions of Java Development Kit 7 (older versions may not include the latest security updates).
- Install the latest version of Java Development Kit 7 by using the Solaris Image Packaging System, by downloading the SVR4 packages and installing them with pkgadd, or by downloading the executables and copying them to the required location.

Uninstall Older Versions of Java Development Kit 7

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

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

2. List the contents of the directory /usr/jdk/instances/, and see if the listing contains a directory named jdk1.7.0.

   In the example, the listing does include a JDK 7 directory:

   ```
   root@mds:~# ls /usr/jdk/instances
   jdk1.5.0 jdk1.6.0 jdk1.7.0 jdk1.8.0
   root@mds:~#
   ```

3. If JDK 7 is already installed, uninstall it using the method that was used to install it. Use the IPS pkg uninstall command, use the SVR 4 pkgrm command, or delete the installation directory with the rm -R /usr/jdk/instances/jdk1.7.0 command.

   In the example, we uninstall an earlier IPS installation:

   ```
   root@mds:~# pkg uninstall //solaris/developer/java/jdk-7
   ...
   root@mds:~#
   ```

4. Now install JDK 7 by using Solaris IPS, by downloading and installing SVR4 packages, or by downloading and installing executables.

Install Java Development Kit 7 Using the Solaris Image Packaging System (IPS)

To install Java Development Kit 7 (JDK 7) using IPS, proceed as follows:

1. If you have not already done so, log in to the Oracle HSM metadata server as root.

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

2. If you have already made the Oracle support repository the default Solaris 11 publisher, go to step 14 and install the JDK.
3. Otherwise, check your Oracle support entitlements. Open a web browser window to the URL **`https://pkg-register.oracle.com.`**
4. Log in to the page using your Oracle Single Sign On user name and password, just as you would on My Oracle Support (**`https://support.oracle.com`**).

   The Oracle SSO user name is usually an e-mail address registered with Oracle. Once you log in, you see a list of entitlements.
5. Click the **CERTIFICATE** link on the upper left part of the screen left under the page-header bar.

   The link URL is **`https://pkg-register.oracle.com/register/certificate`**
6. When the Your Certificate page appears, press the Download Key button to download the cryptographic key for the entitlement. Save the key file to a convenient working directory.
7. While still on the Your Certificate page, press the Download Certificate button to download the cryptographic key for the entitlement. Save the key file to your working directory.
8. Click the **REPOSITORIES** link on the upper left part of the screen left under the page-header bar.

   The link URL is **`https://pkg-register.oracle.com/register/repos/`**
9. Locate the Oracle Solaris 11 Support repository and press the corresponding **Show Details** button.
10. Configure your local repository and install the certificate. Use the command:

    ```
    pkg set-publisher -k dir/pkg.oracle.com.key.pem
    -c dir/pkg.oracle.com.certificate.pem
    -G "*" -g https://pkg.oracle.com/solaris/support/ solaris
    ```
    where **`dir`** is the working directory where you downloaded the files.

    The packaging system creates copies of the key and certificate files and installs them in the required location. In the example, we have downloaded the files to the directory **`/root/mos/`**:

    ```
    root@mds:~# pkg set-publisher -k /root/mos/pkg.oracle.com.key.pem
    -c /root/mos/pkg.oracle.com.certificate.pem -G "*"
    -g https://pkg.oracle.com/solaris/support/ solaris

    Refreshing catalog 1/1 solaris 87.76 MB...
    root@mds:~#
    ```
11. You can now delete the downloaded key and certificate files from the working directory.

    ```
    root@mds:~# rm /root/mos/*.pem
    root@mds:~#
    ```
12. Check your IPS publisher settings, making sure that the configuration does not include mirrors. Use the command **`pkg publisher solaris`**.

    In the example, the string **`Mirror`** does not appear, so there are no mirrors:

    ```
    root@mds:~# pkg publisher solaris
    Publisher: solaris
    ```

```
Alias:
Origin URI: https://pkg.oracle.com/solaris/support/
SSL Key: /var/pkg/ssl/57df...
SSL Cert: /var/pkg/ssl/30d4...
Cert. Effective Date: Thu Jun 08 22:12:19 2017
Cert. Expiration Date: Sun Jun 16 22:12:19 2019
Client UUID: f07e...
Catalog Updated: Tue Jun 06 23:17:46 2017
Enabled: Yes
root@mds:~#
```

13. If the **pkg publisher solaris** output lists mirrors, remove them. Use the command **pkg set-publisher -M mirrorURL solaris**, where **mirrorURL** is the URL of the mirror.

14. Install the JDK 7 software. Use the command **pkg install --accept //solaris/developer/java/jdk-7**.

    The pkg install command installs JDK 7 in the **/usr/jdk/instances/jdk1.7.0/** directory. The **--accept** switch indicates that you agree to the license. The string **//solaris/developer/java/jdk-7** is the Solaris fault management resource identifier (FMRI) that identifies the package within the Solaris Image Packaging System.

```
root@mds:~# pkg install --accept //solaris/developer/java/jdk-7
         Packages to install: 2 ...
DOWNLOAD                        PKGS FILES    XFER (MB)  SPEED
Completed                       2/2  655/655  43.8/43.8  2.4M/s
PHASE                           ITEMS
Installing new actions          738/738
Updating package state database Done
Updating package cache          0/0
Updating image state            Done
Creating fast lookup database   Done
Updating package cache          1/1
root@mds:~#
```

15. If you intend to use the cloud drive encryption feature, enable encryption now.

16. If you are preparing a solution that includes additional Solaris hosts, repeat the tasks specified in "Configuring Oracle Solaris Hosts for Oracle HSM" on page 2-1 until all Solaris hosts have been configured.

17. If you are preparing a solution that includes one or more Linux clients, go to "Configuring Linux Hosts for Oracle HSM Clients".

18. Otherwise, go to "Configuring Storage Hosts and Devices" on page 3-1.

Download and Manually Install Java Development Kit 7 Executables

1. If you have not already done so, log in to the Oracle HSM metadata server as root.

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

2. In a web browser, log in to My Oracle Support (**https://support.oracle.com**).

   You must have a support account.

3. In a web browser tab or window, open the Information Center article "Installation & Configuration for Oracle Java SE" (Doc ID 1412103.2).

   The document can be found at the following URL:

`https://support.oracle.com/epmos/faces/DocContentDisplay?_afrLoop=493538228111468&id=1412103.2&_afrWindowMode=0&_adf.ctrl-state=17bvnyv0r_29`

4. Scroll down the page to the section titled "Downloads (Latest JDK/JRE)", and click on the link corresponding to the latest JDK/JRE 7 build.
5. When the page Patch 13079846: Oracle JDK 1.7.0 opens, use the **Platform** control to select the 32-bit software bundle for the Solaris platform that you use for your Oracle HSM metadata server.

   For example, applicable selections might be named **Oracle Solaris on SPARC (32-bit)** or **Oracle Solaris on x86 (32-bit)**.
6. Click the **Download** button.
7. When the File Download panel appears, click the link to the patch archive (zip) file and save the file to your working directory.

   JDK 1.7 patch archive files are named **`pidnumber_17000_platform.zip`**, where:

   - **`idnumber`** is a string of numerals
   - **`platform`** is the operating system and hardware platform that you selected

   For example, the download file might be **`p13079846_17000_SOLARIS.zip`** for SPARC platforms or **`p13079846_17000_Solarisx86.zip`** for x86 platforms.
8. If you require 64-bit JDK 7 executables for other uses, repeat this procedure, this time selecting the 64-bit software bundle for the Solaris platform that you use for your Oracle HSM metadata server.
9. Next, install JDK 7, either by using SVR4 packages or by manually installing the executables.
10. In the patch archive (.zip) file, locate the compressed tape archive file **`jdk-7ubuild-platform.tar.gz,`** where:

    - **`build`** is the build number
    - **`platform`** is the operating system and hardware platform that you selected

    For example, **`jdk-7u141-solaris-sparc.tar.gz`** would contain 32-bit SPARC executables, while **`jdk-7u141-solaris-i586.tar.gz`** would hold the equivalents for the x86 platform.
11. Decompress the tape archive (.tar) file. Use the command **`gunzip jdk-7ubuild-platform.tar.gz`**.

    In the example, we extract Solaris SPARC packages for JDK 7 build 141:

    ```
    root@mds:~# gunzip jdk-7u141-solaris-sparc.tar.gz
    root@mds:~# ls
    jdk-7u141-solaris-i586.tar
    ```
12. Extract the directory that holds the executables, **`jdk1.7.0_build`**, from the tape archive (tar) file to the working directory. Use the command **`gunzip jdk-7ubuild-platform.tar.gz`** and **`tar -xf jdk-7ubuild-platform.tar`**

    ```
    root@mds:~# tar -xf jdk-7u141-solaris-sparc.tar
    ...
    root@mds:~#
    ```

13. If you require 64-bit JDK 7 executables for other uses, repeat this procedure, this time using the 64-bit software bundle for the Solaris platform that you use for your Oracle HSM metadata server.
14. Move the directory that holds the executables, jdk1.7.0_build to the JDK installation directory and rename it, omitting the build number string. Use the command mv jdk1.7.0_141 /usr/jdk/instances/jdk1.7.0.

    JDK 7 is now installed.

    ```
    root@mds:~# mv jdk1.7.0_141 /usr/jdk/instances/jdk1.7.0
    root@mds:~#
    ```

15. If you intend to use the cloud drive encryption feature, enable encryption now.
16. If you are preparing a solution that includes additional Solaris hosts, repeat the tasks specified in "Configuring Oracle Solaris Hosts for Oracle HSM" on page 2-1 until all Solaris hosts have been configured.
17. If you are preparing a solution that includes one or more Linux clients, go to "Configuring Linux Hosts for Oracle HSM Clients".
18. Otherwise, go to "Configuring Storage Hosts and Devices" on page 3-1.

Download and Install Java Development Kit 7 Using SVR4 Packages

1. In the patch archive (.zip) file, locate the compressed tape archive file **jdk-7ubuild-platform.tar.gz,** where:
   - **build** is the build number
   - **platform** is the operating system and hardware platform that you selected

   For example, **jdk-7u141-solaris-sparc.tar.gz** would contain 32-bit SPARC executables, while **jdk-7u141-solaris-i586.tar.gz** would hold the equivalents for the x86 platform.
2. Decompress the tape archive (tar) file, and extract the SVR4 packages from the tape archive (tar) to the working directory. Use the command **zcat jdk-7ubuild-platform.tar.Z | tar xf -**.

   In the example, we extract Solaris SPARC packages for JDK 7 build 141:

   ```
   root@mds:~# zcat jdk-7u141-solaris-sparc.tar.Z | tar xf -
   ```

3. Install the SVR4 packages, answering **yes** to all questions. Use the command **yes | pkgadd -d . package-list,** where **package-list** is a space delimited list of package names.

   In the example, the list of packages includes **SUNWj7dev**, **SUNWj7rt**, **SUNWj7cfg**, **SUNWj7jmp**, and **SUNWj7man**.

   ```
   root@mds:~# yes | pkgadd -d . SUNWj7dev SUNWj7rt SUNWj7cfg SUNWj7jmp SUNWj7man
   ```

4. If you intend to use the cloud drive encryption feature, enable encryption now.
5. If you are preparing a solution that includes additional Solaris hosts, repeat the tasks specified in "Configuring Oracle Solaris Hosts for Oracle HSM" on page 2-1 until all Solaris hosts have been configured.
6. If you are preparing a solution that includes one or more Linux clients, go to "Configuring Linux Hosts for Oracle HSM Clients".
7. Otherwise, go to "Configuring Storage Hosts and Devices" on page 3-1.

Enable Cloud Encryption

If you intend to use the cloud-drive encryption feature of the cloud library (cr) equipment type, carry out the following tasks:

- Install Java Cryptography Extensions Unlimited Strength Jurisdiction Policy Files 7.
- If you plan to manage encryption keys using a Public Key Cryptography Standards #11 (Cryptoki) key manager, configure the Oracle HSM metadata server to use the selected key manager, Oracle Key Manager (OKM) or Oracle Key Vault (OKV).
- If you plan to manage encryption keys using a local file, create the keystore file now.

Install Java Cryptography Extensions Unlimited Strength Jurisdiction Policy Files 7

Java Cryptography Extensions (JCE) jurisdiction policy files specify the cryptographic strength that the JDK/JCE software can provide. The default jurisdiction policy files distributed with the JDK 7 development kit limit cryptographic strength in keeping with the import restrictions in force in some countries. The JCE Unlimited Strength Jurisdiction Policy Files 7 allow unlimited strength encryption.

1. If you have not already done so, log in to the Oracle HSM metadata server as root.

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

2. In a web browser window, open the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 7 Download page.

   The page is available at the following URL:

   http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html

3. Click the **Accept License Agreement** radio button.
4. In the download field, click on the link **`UnlimitedJCEPolicyJDK7.zip`**.
5. When prompted, save the **`UnlimitedJCEPolicyJDK7.zip`** file in a convenient working directory.
6. Decompress and extract the contents of the **`UnlimitedJCEPolicyJDK7.zip`** file.

   The file contains a directory, **`jce/`**, that holds a **`README.txt`** file and local and export policy Java Archive (jar) files:

   ```
   root@solaris:~# ls
   jce
   root@solaris:~# ls jce
   README.txt
   local_policy.jar
   US_export_policy.jar
   root@solaris:~#
   ```

7. Read the **`README.txt`** file carefully, so that you understand the jurisdictional issues surrounding import and export of strong cryptography.
8. Install the policy files, **`US_export_policy.jar`** and **`local_policy.jar`**. Copy the files to the directory **`/usr/jdk/instances/jdk1.7.0/jre/lib/security/`**.

   ```
   root@mds:~# cp US_export_policy.jar
   /usr/jdk/instances/jdk1.7.0/jre/lib/security/
   root@mds:~# cp local_policy.jar /usr/jdk/instances/jdk1.7.0/jre/lib/security/
   ```

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

9. If you plan on managing cryptographic keys using a PKCS #11 key manager rather than a local text file, enroll Oracle HSM with the key manager now. You can enroll the Oracle HSM metadata server with Oracle Key Manager or with Oracle Key Vault.
10. If you plan on managing cryptographic keys using a local keystore file, create the file now.
11. If you are preparing a solution that includes additional Solaris hosts, repeat the tasks specified in "Configuring Oracle Solaris Hosts for Oracle HSM" on page 2-1 until all Solaris hosts have been configured.
12. If you are preparing a solution that includes one or more Linux clients, go to "Configuring Linux Hosts for Oracle HSM Clients".
13. Otherwise, go to "Configuring Storage Hosts and Devices" on page 3-1.

Configure Oracle HSM Metadata Server for Use with Oracle Key Manager (OKM)

1. If you plan on using Oracle Key Manager (OKM) as your key manager, have an OKM administrator register Oracle HSM as an agent and provide the following access information:
   - the name of the profile that applies to Oracle HSM
   - the OKM Agent ID for Oracle HSM
   - the password for logging in as Oracle HSM
   - the IP address of the Key Management Appliance (KMA) that Oracle HSM should try to access first.

     A Key Management Appliance (KMA) is a host node in an OKM cluster.

   For more information on OKM, see the *Oracle Key Manager Online Documentation Library* at the following URL:

   http://www.oracle.com/technetwork/documentation/tape-storage-curr-187744.html#crypto
2. If you have not already done so, log in to the Oracle HSM metadata server as root.

   ```
   root@mds:~#
   ```
3. Install the **pkcs11_kms** package on the Oracle HSM metadata server. Use the IPS command **pkg install system/library/security/crypto/pkcs11_kms**.

   The Oracle Solaris **pkcs11_kms** encryption services provider lets Oracle HSM use encryption and decryption keys by requesting application-defined *key labels*. Using a private protocol, the **pkcs11_kms** provider passes this label to Oracle Key Manager (OKM) for inclusion in the metadata associated with encrypted data. The client application can then encrypt and decrypt data via the key management server, without direct access to encryption keys.

   ```
   root@mds:~# pkg install system/library/security/crypto/pkcs11_kms
   Packages to install: 1
   Create boot environment: No
   Create backup boot environment: No
   DOWNLOAD PKGS FILES XFER (MB) SPEED
   Completed 1/1 16/16 0.6/0.6 1.6M/s
   PHASE ITEMS
   Installing new actions 43/43
   ```

```
...
Updating package cache 1/1
root@mds:~#
```

4. Plug **pkcs11_kms** into the Solaris Cryptographic Framework. Use the command **cryptoadm install provider='/usr/lib/security/$ISA/pkcs11_kms.so.1'**

```
root@mds:~# cryptoadm install provider='/usr/lib/security/$ISA/pkcs11_kms.so.1'
root@mds:~#
```

5. Start the **pkcs11_kms** configuration process. Enter the command **kmscfg**.

   The **Profile Name:** prompt appears:

```
root@mds:~# kmscfg
Profile Name:
```

6. At the **Profile Name:** prompt, enter the name of the profile that the OKM administrator specified for use with Oracle HSM.

   In the example, the specified profile is named **HSMcloud**:

```
root@mds:~# kmscfg
Profile Name: HSMcloud
```

7. When prompted for an **Agent Name,** enter the agent ID that the OKM administrator assigned to Oracle HSM.

   In the example, the Agent ID is **HSMcloudArchive**:

```
root@mds:~# kmscfg
Agent ID: HSMcloudArchive
```

8. When prompted for an **KMA IP Address**, enter the Internet Protocol address that the OKM administrator provided.

   In the example, this IP address is **192.168.123.123:**

```
root@mds:~# kmscfg
KMA IP Address:: 192.168.123.123
```

9. Display the configuration. Use the command **cryptoadm list -m -v provider='/usr/lib/security/$ISA/pkcs11_kms.so.1'**

```
root@mds:~# cryptoadm list -m -v provider='/usr/lib/security/$ISA/pkcs11_
kms.so.1'
Provider: /usr/lib/security/$ISA/pkcs11_kms.so.1
Number of slots: 1
Slot #1
Description: Oracle Key Management System
Manufacturer: Oracle Corporation
PKCS#11 Version: 2.20
Hardware Version: 0.0
Firmware Version: 0.0
Token Present: True
Slot Flags: CKF_TOKEN_PRESENT
Token Label: KMS
...

Mechanisms: E D S V P E
n e D i V e K a U D C
c c i g e r e i n e
r r g S + r + y r W w r C
```

```
y y e i R i R G G r r i a
H p p s g e f e e e a a v p
Mechanism Name             Minimum    Maximum W t t t n c y c n n p p e s
------------------------ -------- ---------- - - - - - - - - - - - - - -
CKM_AES_KEY_GEN                 32         32 . . . . . . . . X . . . . .
CKM_AES_CBC                     32         32 . X X . . . . . . . X X . .
CKM_AES_CBC_PAD                 32         32 . X X . . . . . . . X X . .
root@mds:~#
```

10. If you are preparing a solution that includes additional Solaris hosts, repeat the tasks specified in "Configuring Oracle Solaris Hosts for Oracle HSM" on page 2-1 until all Solaris hosts have been configured.

11. If you are preparing a solution that includes one or more Linux clients, go to "Configuring Linux Hosts for Oracle HSM Clients".

12. Otherwise, go to "Configuring Storage Hosts and Devices" on page 3-1.

Configure the Oracle HSM Metadata Server for Use with Oracle Key Vault (OKV)

1. If you plan on using Oracle Key Vault (OKV) as your key manager, have an OKV system administrator initiate enrollment of the Oracle HSM metadata server as an OKV endpoint.

   The OKV administer will provide you with a one-time enrollment token via email or some other out-of-band communication method.

2. When you receive the enrollment token, download and install the OKV client software. Follow the instructions the chapter "Using Oracle Key Vault Endpoints" in the Key Vault Administrator's Guide, Oracle Key Vault 12c Release 1 Documentation Library.

   Full documentation for all available versions of Oracle Key Vault can be found at the following URL:

   https://docs.oracle.com/en/database/related-products.html#DatabaseSecurity

3. If you are preparing a solution that includes additional Solaris hosts, repeat the tasks specified in "Configuring Oracle Solaris Hosts for Oracle HSM" on page 2-1 until all Solaris hosts have been configured.

4. If you are preparing a solution that includes one or more Linux clients, go to "Configuring Linux Hosts for Oracle HSM Clients".

5. Otherwise, go to "Configuring Storage Hosts and Devices" on page 3-1.

Create and Configure an Encryption Keystore File

For each cloud library that will manage encryption keys using a local 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. Using a text editor, create the keystore file for the cloud library.

   In the example, we create the keystore file in the directory **/root/**. For clarity, we name the file **cl800.ksf**, where **cl800** is the family set name of corresponding cloud library, and add a comment:

   ```
   root@mds1:~# vi /root/cl800.ksf
   # keystore file for Oracle HSM cloud library cl800
   ```

3. 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 key that cloud drives will use when encrypting volumes stored in the cloud library.

   In the example, the key label is **`Key1`**.

```
root@mds1:~# vi /root/cl800.ksf
# keystore file for Oracle HSM cloud library cl800
key-label = Key1
```

4. In another terminal window, create an 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 AES key **`4e6e2666f...41ba25e3`**:

```
root@mds1:~# dd if=/dev/urandom bs=32 count=1 2>/dev/null | od -t x1 -An | tr
-d '\n \t' ; echo
4e6e2666f053e84ce8f1b67308c77d2d884e2c182bdaf965040c590f41ba25e3
root@mds1:~#
```

5. Using the text editor, add a line to the keystore file of the form **`key-value = AES_key`**, where AES_key is the key that you just generated.

   In the example, the key-value is **`4e6e2666f...41ba25e3`**:

```
root@mds1:~# vi /root/cl800.ksf
# keystore file for Oracle HSM cloud library cl800
key-label = Key1
key-value = 4e6e2666f053e84ce8f1b67308c77d2d884e2c182bdaf965040c590f41ba25e3
```

6. 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 **`Key1`** and **`4e6e2666f...41ba25e3 is 1384cec4...9f522186`**:

```
root@mds1:~# print -n "Key14e6e2666f...41ba25e3" | digest -a sha256
1384cec4e2e81eb80bed983a484b57dcaeaccea0d98ef8d068f00fb29f522186
root@mds1:~#
```

7. 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 **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
```

8. Save the keystore file.

```
root@mds1:~# vi /root/cl800.ksf
# keystore file for Oracle HSM cloud library cl800
key-label = Key1
key-value = 4e6e2666f053e84ce8f1b67308c77d2d884e2c182bdaf965040c590f41ba25e3
key-hash = 1384cec4e2e81eb80bed983a484b57dcaeaccea0d98ef8d068f00fb29f522186
:w
```

9. For each additional key that you intend to use when encrypting volumes in this cloud library, repeat steps 3 through 8.
10. When you have created keystore records for all required keys, close the editor.

    In the example, the finished keystore file holds entries for two AES keys, labelled **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
:q
root@mds1:~#
```

11. Encrypt the keystore file. Use the command encrypt **-a aes -i inputfile -o outputfile**, and enter a strong password 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 a strong 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:~#
```

12. Create a password file to hold the password for the AES-encrypted keystore file. Use the command sam-cloudd -p keyfile_password_file, where keyfile_password_

file is the fully qualified path and file name of the new password file. When prompted for a **Password**, enter the password for the AES-encrypted keystore file.

In the example, we create the password file **cl800.ksf.pwd** in the directory **/root/** and enter the password that we used when encrypting keystore file **cl800.ksf**:

```
root@mds1:~# sam-cloudd -p /root/cl800.ksf.pwd
Enter Password: P^ssw0rd
Reenter Password: P^ssw0rd
root@mds1:~#
```

13. Change the keystore file permissions so that the owner has read and write access and others have none. Use the command **chmod 0600 keystore_file**, where **keystore_file** is the fully qualified path and file name of the keystore file.

    In the example, the keystore file is **/root/cl800.ksf**:

```
root@mds1:~# chmod 0600 /root/cl800.ksf
root@mds1:~#
```

14. If you are preparing a solution that includes additional Solaris hosts, repeat the tasks specified in "Configuring Oracle Solaris Hosts for Oracle HSM" on page 2-1 until all Solaris hosts have been configured.
15. If you are preparing a solution that includes one or more Linux clients, go to "Configuring Linux Hosts for Oracle HSM Clients".
16. Otherwise, go to "Configuring Storage Hosts and Devices" on page 3-1.

Configuring Linux Hosts for Oracle HSM Clients

Before you install the Oracle HSM client software, you must the prepare the Linux operating system as follows:

- Disable incompatible operating system features.
- Install the required kernel-development and utility packages.

Disable Incompatible Operating System Features

1. Log in to the Oracle HSM client host as **root**.

```
[root@linux ~]#
```

2. If SELinux (Secure Linux) is installed, disable it. Open the file **/etc/selinux/config** in a text editor, set the **SELINUX** flag to **disabled**, save the file, close the editor, and reboot.

   Oracle HSM does not support SELinux, which is enabled by default on Oracle Linux and Red Hat Enterprise Linux. In the example, we open the file in the **vi** editor.

```
[root@linux ~]# vi /etc/selinux/config
# This file controls the state of SELinux on the system.
...
#SELINUX=enforcing
#SELINUX=permissive
SELINUX=disabled
SELINUXTYPE=targeted
:wq
[root@linux ~]# reboot
```

3. If AppArmor is installed, disable it using the procedure recommended in the documentation for your Linux distribution.

   AppArmor is sometimes used as an alternative to SELinux. Oracle HSM does not support AppArmor.

4. Next, install the required kernel-development and utility packages.

Install Required Kernel Development and Utility Packages

Prior to installation of the Oracle HSM client software, the Linux kernel development package has to be installed, along with some specified utility packages. To identify and install required packages, use the following procedure:

1. Log in to the Linux client host as **root**.

   In the example, the client is hosted on Oracle Linux:

   ```
   [root@linux ~]#
   ```

2. Identify the kernel version installed on the client. Use the command **uname -r**.

   In the example, the kernel version is **2.6.9-89.0.0.0.1.EL**:

   ```
   [root@linux ~]# uname -r
   2.6.9-89.0.0.0.1.EL
   [root@linux ~]#
   ```

3. Install the kernel development kit, **kernel-devel-***kernel-version*, where *kernel-version* is the version string that you identified in the preceding step.

   The Oracle HSM client installation requires the **Module.symvers** that is part of this package. In the example, we use the Oracle Linux command **yum** with parameters **-y install** (**-y** to insure that all prompts are automatically answered "yes"):

   ```
   [root@linux ~]# yum -y install \ kernel-devel-2.6.9-89.0.0.0.1.EL.i686.rpm
   [root@linux ~]#
   ```

4. See if the Korn shell, **ksh**, is installed. If it is not, install it.

   In the example, we pipe the output of the Oracle Linux command **rpm -qa** to the **grep** command and search for the string **ksh**. The command returns no output, which shows that **ksh** is not installed. So we install it using the command **yum install ksh**:

   ```
   [root@linux ~]# rpm -qa | grep ksh
   [root@linux ~]#
   [root@linux ~]# yum install ksh
   ...
   --> Running transaction check
   ---> Package ksh-20100621-19.e16.x86_64 set to be installed

   ===============================================================================
   Package            Arch          Version                  Repository      Size
   ===============================================================================
   Installing:
    ksh               i686          2.6.9-89.0.0.0.1.EL       updates        506 k
   ...
   Installed:
     ksh-2.6.9-89.0.0.0.1.EL.i686
   Complete!
   [root@linux ~]#
   ```

5. See if the **cpio** utility is installed. If it is not, install it.

   In the example, we pipe the output of the Oracle Linux command **rpm -qa** to the **grep** command and search for the string **cpio**. The command returns version information, so the **cpio** utility is installed:

   ```
   [root@linux ~]#  rpm -qa | grep cpio
   cpio-2.10-10.el6.x86_64
   [root@linux ~]#
   ```

6. See if the **find** utilities are installed. If they are not, install them.

   In the example, we pipe the output of the Oracle Linux command **rpm -qa** to the **grep** command and search for the string **findutils**. The command returns version information, so the **findutils** package is installed:

   ```
   [root@linux ~]#  rpm -qa | grep findutils
   findutils-4.4.2-6.el6.x86_64
   [root@linux ~]#
   ```

7. See if the **gcc** compiler is installed. If it is not, install it.

   In the example, we pipe the output of the Oracle Linux command **rpm -qa** to the **grep** command and search for the string **gcc**. The command returns version information, so the **gcc** compiler is installed:

   ```
   [root@linux ~]#  rpm -qa | grep gcc
   gcc-4.4.7-3.el6.x86_64
   libgcc-4.4.7-3.el6.x86_64
   [root@linux ~]#
   ```

8. See if the **make** utility is installed. If it is not, install it.

   In the example, we pipe the output of the Oracle Linux command **rpm -qa** to the **grep** command and search for the string **make**. The command returns version information, so the **make** utility is installed:

   ```
   [root@linux ~]#  rpm -qa | grep make
   make-4.4.7-3.el6.x86_64
   libmake-3.81.20.el6.x86_64
   [root@linux ~]#
   ```

9. See if the **binutils** package is installed. If it is not, install it.

   If the Oracle HSM installation software needs to build the Linux kernel, it requires the **nm** utility, which is part of this package. In the example, we pipe the output of the Oracle Linux command **rpm -qa** to the **grep** command and search for the string **nm**. The command returns version information, so the **nm** utility is installed:

   ```
   [root@linux ~]#  rpm -qa | grep nm
   binutils-2.20.51.0.2-5.34.el6.x86_64
   [root@linux ~]#
   ```

10. See if the **rpmbuild** package is installed. If it is not, install it.

    In the example, we pipe the output of the Oracle Linux command **rpm -qa** to the **grep** command and search for the string **rpmbuild**. The command returns version information, so the **rpmbuild** package is installed:

    ```
    [root@linux ~]#  rpm -qa | grep rpmbuild
    rpm-build-4.8.0-37.el6.x86_64
    [root@linux ~]#
    ```

11. See if the **rpm** package is installed. If it is not, install it.

    If the Oracle HSM installation software needs to build the Linux kernel, it requires the **rpm2cpio** utility, which is part of this package. In the example, we pipe the output of the Oracle Linux command **rpm -qa** to the **grep** command and search for the string **rpm**. The command returns version information, so the utility is installed:

    ```
    [root@linux ~]#  rpm -qa | grep rpm
    rpm-4.8.0-27.el6.x86_64
    rpm-libs-4.8.0-27.el6.x86_64
    rpm-python-4.8.0-27.el6.x86_64
    [root@linux ~]#
    ```

12. If you are preparing a solution that includes additional Linux clients, repeat the tasks specified in "Configuring Linux Hosts for Oracle HSM Clients" on page 2-15 until all Linux clients have been configured.

13. Otherwise, go to "Configuring Storage Hosts and Devices" on page 3-1.

3 Configuring Storage Hosts and Devices

Carry out the storage configuration tasks outlined in this chapter before proceeding further with Oracle HSM installation and configuration:

- Configure primary storage.
- Then, if you are creating Oracle HSM archiving file systems, configure archival storage.
- Finally, if you are creating a clustered solution, configure storage for high-availability.

Configuring Primary Storage

In an Oracle HSM file system, primary disk or solid-state disk devices store files that are being actively used and modified. Follow the guidelines below when configuring disk or solid-state disk devices for the cache.

Configure Devices for the Primary Cache

1. To estimate a starting capacity for the primary cache, decide how much data each file system will hold when full.
2. Increase this starting capacity by 10% to allow for file-system metadata.
3. If you are preparing for a high-performance **`ma`**-type file system, configure hardware for the **`mm`** metadata devices. One, hardware-controlled, four-disk, RAID 10 (1+0) volume group per **`mm`** metadata device is ideal. Consider using solid-state disk devices for maximum performance.

   The characteristics of striped-mirror, RAID 10 arrays are ideal for storing Oracle HSM metadata. RAID 10 storage hardware is highly redundant, so critical metadata is protected. Throughput is higher and latency is lower than in most other RAID configurations.

   An array that is controlled by dedicated controller hardware generally offers higher performance than an array controlled by software running on a shared, general-purpose processor.

   Solid-state devices are particularly useful for storing metadata that is, by its nature, frequently updated and frequently read.
4. If you are using an external disk array for primary cache storage, configure 3+1 or 4+1 RAID 5 volume groups for each **`md`** or **`mr`** device in the file-system configuration. Configure one logical volume (LUN) on each volume group.

For a given number of disks, smaller, 3+1 and 4+1 RAID 5 volume groups provide greater parallelism and thus higher input/output (I/O) performance than larger volume groups. The individual disk devices in RAID 5 volume groups do not operate independently—from an I/O perspective, each volume group acts much like a single device. So dividing a given number of disks into 3+1 and 4+1 volume groups creates more independent devices, better parallelism, and less I/O contention than otherwise equivalent, larger configurations.

Smaller RAID groups offer less capacity, due to the higher ratio of parity to storage. But, for most users, this is more than offset by the performance gains. In an archiving file system, the small reduction in disk cache capacity is often completely offset by the comparatively unlimited capacity available in the archive.

Configuring multiple logical volumes (LUNs) on a volume group makes I/O to the logically separate volumes contend for a set of resources that can service only one I/O at a time. This increases I/O-related overhead and reduces throughput.

5. Next, start configuring archival storage.

Configuring Archival Storage

Carry out the tasks that your planned archiving configuration requires:

- Zone SAN-attached devices.
- Configure archival disk storage.
- Configure archival tape storage.
- Configure cloud resources.

Zone SAN-attached Devices

Make sure that the storage area network (SAN) is zoned to allow communication between the drive and the host bus adapters on the Oracle HSM host(s). To check zoning, proceed as follows:

1. Make sure that the host can see the devices on the SAN. Enter the Solaris configuration administration command **cfgadm** with the **-al** (*attachment-points list*) and **-o show_SCSI_LUN** options. Examine the output for the World Wide Name (WWN) of the drive port.

   The first column of the output displays the attachment-point ID (**Ap_id**), which consists of the controller number of the host bus adapter and the WWN, separated by colons. The **-o show_SCSI_LUN** option displays all LUNs on the node if the node is the bridged drive controlling a media changer via an ADI interface.

```
root@solaris:~# cfgadm -al -o show_SCSI_LUN
Ap_Id     Type Receptacle Occupant   Condition
c2::500104f000937528   tape connected  configured   unknown
c3::50060160082006e2,0 tape connected  unconfigured unknown
```

2. If the drive's WWN is not listed in the output of **cfgadm -al -o show_SCSI_LUN**, the drive is not visible. Something is wrong with the SAN configuration. So recheck SAN connections and the zoning configuration. Then repeat the preceding step.

3. If the output of the **cfgadm -al** command shows that a drive is unconfigured, run the command again, this time using the **-c** (*configure*) switch.

   The command builds the necessary device files in **/dev/rmt**:

```
root@solaris:~# cfgadm -al
Ap_Id     Type Receptacle Occupant   Condition
c2::500104f000937528   tape connected  configured   unknown
c3::50060160082006e2,0 tape connected  unconfigured unknown
root@solaris:~# cfgadm -c configure 50060160082006e2,0
```

4. Verify the association between the device name and the World Wide Name. Use the command **ls -al /dev/rmt | grep** *WWN*, where *WWN* is the World Wide Name.

```
root@solaris:~# ls -al /dev/rmt | grep 50060160082006e2,0
lrwxrwxrwx 1 root root 94 May 20 05:05 3un -> \
../../devices/pci@1f,700000/SUNW,qlc@2/fp@0,0/st@w50060160082006e2,0:
```

5. If you have the recommended minimum Solaris patch level, configure disk storage now.
6. Otherwise, get the target ID for your device.
7. Edit **/kernel/drv/st.conf**. Add the vendor-specified entry to the **tape-config-list**, specifying the target ID determined above.
8. Force reload the **st** module. Use the command **update_drv -f st**.

```
root@solaris:~# update_drv -f st
root@solaris:~#
```

9. Next, configure disk storage.

Configure Archival Disk Storage

Oracle HSM archiving file systems can archive files to disk as well as to tape media. When a disk file system is configured as a *disk archive*, the software uses the file system more or less as it would a tape cartridge. It addresses the file system by volume serial number (VSN) and stores file copies in tape archive (**tar**) files.

Disk-based archival storage can increase the flexibility and redundancy of an archiving solution. Random-access disk devices do not incur the mounting and positioning overhead associated with sequential-access tape devices. So solutions that archive and retrieve proportionately large numbers of small files may be able do so more rapidly and reliably when they store the first copy of each file on disk. Archiving solutions that must maintain copies on off-site media can often do so simply by writing a copy to NFS-mounted, disk-resident file systems on remote hosts.

You can use ZFS, UFS, NFS, or StorageTek QFS file systems for the volumes in a disk archive. For best archiving and staging performance, configure file systems and underlying storage to maximize the bandwidth available for archiving and staging, while minimizing I/O contention between archiving and staging jobs and between Oracle HSM and other applications. Plan for enough hardware resources to handle the anticipated workload.

Proceed as follows:

1. Estimate the number of archive volumes that you require and the capacity of each.

   Each Oracle HSM process performs I/O to one archive volume at a time. So the number of available volumes is proportional to the number of archiving and staging operations that can be carried out in parallel. In general, a larger number of lower capacity volumes performs better than a smaller number of higher capacity volumes, as long as each volume has enough capacity to store a reasonable number of archive files.

2. Allocate enough direct-attached and/or network-accessible hardware to provide the required number of volumes.

   You can use local disk, SAN-attached RAID storage, and/or NSF-mounted remote resources as archival volumes.

3. Dedicate a set of independent storage devices to each archival volume.

   Volumes should never share a set of physical disks or RAID groups with other archival volumes. If you configure LUNs (logical devices) that reside on a single set of physical devices as volumes, archiving and staging processes will contend for access to a single set of hardware, and switching losses will significantly degrade performance.

4. For each required volume, create one ZFS, UFS, or QFS file system on one independent set of devices, using the procedures specified for the chosen file-system type (for QFS file systems, see "Configuring QFS File Systems" on page 6-1).

   Each file system should function as a single archival volume. If you configure subdirectories of a file system as volumes, archiving and staging processes will contend for access to a single set of hardware, and switching losses will significantly degrade performance.

   For the examples in this section, we create fifteen file systems:

   - **DISKVOL1** is a local QFS file system that we create specifically for use as archival storage.
   - **DISKVOL2** to **DISKVOL15** are ZFS file systems mounted on a remote server named **server**.

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

   ```
   root@solaris:~# mkdir /diskvols
   root@solaris:~#
   ```

6. On the Oracle HSM metadata server host, create a single parent directory to hold the mount points for all archival disk volumes.

   This parent directory is analogous to the physical tape library that holds archival tape volumes. In the example, we create the directory **/diskvols**:

   ```
   root@solaris:~# mkdir /diskvols
   root@solaris:~#
   ```

7. In the parent directory, create a mount-point directory for each archival file system.

   In the example, we create the mount-point directories **DISKVOL1** and **DISKVOL2** to **DISKVOL15**:

   ```
   root@solaris:~# mkdir /diskvols/DISKVOL1
   root@solaris:~# mkdir /diskvols/DISKVOL2
   ...
   root@solaris:~# mkdir /diskvols/DISKVOL15
   root@solaris:~#
   ```

8. Oracle HSM metadata server host, back up the **/etc/vfstab** file.

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

9. Open the **/etc/vfstab** file in an editor, add entries for each archival file system, and add the mount option **nosam** to the entries for any QFS file system. Save the file, and close the editor.

   The **nosam** mount option makes sure that archival copies stored on a QFS file system are not themselves archived.

   In the example, we use the **vi** editor to add entries for **DISKVOL1** and **DISKVOL2** to **DISKVOL15**.

```
root@solaris:~# 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       -
...
DISKVOL1          -        /diskvols/DISKVOL1  samfs   -     yes      nosam
server:/DISKVOL2  -        /diskvols/DISKVOL2  nfs     -     yes
server:/DISKVOL3  -        /diskvols/DISKVOL3  nfs     -     yes
...
server:/DISKVOL15 -        /diskvols/DISKVOL15 nfs     -     yes
:wq
root@solaris:~#
```

10. Mount the archival file system(s) on the Oracle HSM metadata server host.

    In the example, we mount **DISKVOL1** and **DISKVOL2** to **DISKVOL15**:

```
root@solaris:~# mount /diskvols/DISKVOL1
root@solaris:~# mount /diskvols/DISKVOL2
...
root@solaris:~# mount /diskvols/DISKVOL15
```

11. If the storage space for an archive volume will be dynamically allocated from a pool of shared disk devices, make sure that the underlying physical storage is not oversubscribed. Set quotas.

    Quotas help to keep Oracle HSM archiving processes from trying to use more of the aggregate storage than it has available.

12. Plan to use each new file system exclusively as an Oracle HSM archival volume.

    If other applications use an Oracle HSM archival volume as a general-purpose file system, Oracle HSM and application processes will contend for access to a single set of hardware, and switching losses will significantly degrade performance.

13. If file systems have been configured on remote hosts, share them using the Network File System (NFS), and NFS mount them on the Oracle HSM metadata server.

14. If you plan to archive files to the cloud, configure cloud resources now.

15. Otherwise, configure archival tape storage.

Provide Storage Cloud Resources

If you plan to archive files in a public or private storage cloud, provide the required cloud resources. There are two approaches that you can take:

- You can use Oracle Storage Cloud services for a ready-made solution.

- You can use another Solaris 11.3 system with Oracle OpenStack Swift running on top of another instance of Oracle HSM. This will provide another (possibly remote) tier of storage for archiving.

Use the Oracle Storage Cloud

Carry out the following tasks:

- Subscribe to the service.
- Configure the Oracle Storage Cloud REST APIs for use with Oracle HSM.

Subscribe to Oracle Storage Cloud Services

To set up domains and user accounts for an Oracle Storage Cloud solution, proceed as follows:

1. Consult the Oracle Storage Cloud documentation for descriptions of service levels and instructions for subscribing.

   See *Getting Started with Oracle Cloud* at **`http://docs.oracle.com/en/cloud/`**.

2. Provide an unmetered Oracle Storage Cloud account for files that users frequently access following archiving. Follow the instructions in "Buying a Nonmetered Subscription to an Oracle Cloud Service", Chapter 4 of *Getting Started with Oracle Cloud*.

   Users generally request files of this kind within a more or less predictable interval following archiving—often 30 days. So a tiered approach to cloud provisioning makes sense. Size the unmetered account to hold one copy of each newly archived file for the duration of this initial, high-access period. Thereafter, the copy can be released.

   Compared to metered accounts, unmetered Oracle Storage Cloud Standard Service accounts are more expensive for a given capacity and significantly less expensive per I/O operation. So using an unmetered account is ideal for archiving files for the near term, during the period when they are most likely to be read and updated.

3. Provide a metered Oracle Storage Cloud account for files that users access less often. Follow the instructions in "Buying a Metered Subscription to an Oracle Cloud Service", Chapter 5 of *Getting Started with Oracle Cloud*.

   For a given capacity, metered Oracle Storage Cloud accounts are less expensive than unmetered accounts and more expensive per I/O operation.

4. For files that users continue to access following archiving, plan to configure an Oracle HSM cloud library with Oracle Storage Cloud Standard containers.

   Users generally request files within a more or less predictable period following archiving—often 60 to 90 days. For a given capacity, Standard containers are more expensive than Archive containers but less expensive per I/O operation. So archiving sporadically accessed files to a cloud library configured for Standard containers best balances the cost of occasional file I/O against the cost of the storage.

5. For long-term storage of files that users will access rarely, if ever following archiving, plan to configure an Oracle HSM cloud library with Oracle Storage Cloud Archive containers.

   Users seldom request files that have been archived for 90 days or more, and many files are never requested at all. Yet, in most cases, every archived file has to be retained for some specified period. Archive containers are expensive on a per I/O basis. But, for a given capacity, they are the least expensive option. So archiving little used files to a cloud library configured for Archive containers minimizes the overall cost of archiving.

6. Next, gather Oracle Storage Cloud authentication information.

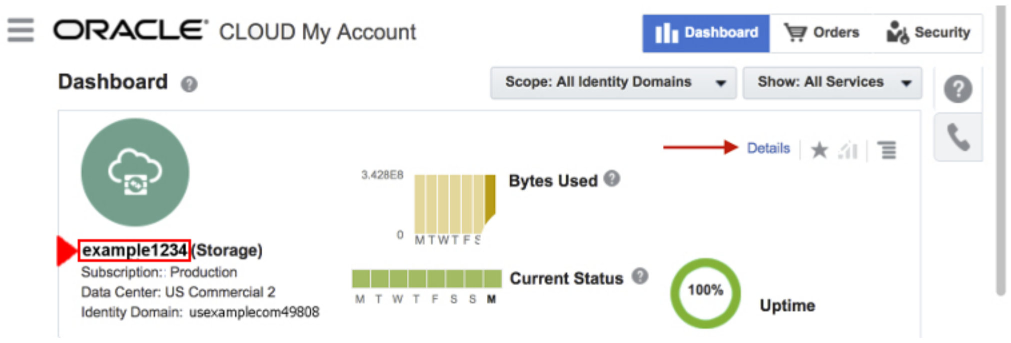
Gather Oracle Storage Cloud Authentication Information

1. Log in to the Oracle Cloud **My Account** page, if you have not already done so.
2. On the **Dashboard**, locate the listing for the Oracle Storage Cloud service that will provide the storage, and make a note of the service name.

   Services are identified by a string of the form ***service_name* (*service_type*)**. In the example, the service name is **example1234** and the service type is **Storage** (displayed in parentheses). Under the service name and type, a **Subscription** field shows that this is a production service, a **Data Center** field shows that it is hosted in the US Commercial 2 datacenter, and an **Identity Domain** field displays the Identity Domain ID.

   We use the **vi** text editor to create a text file called **hsm-ocloud-info.txt** and make a note of the service name:

```
hsmadm@wrkstn:~# vi hsm-ocloud-info.txt
user name:          hsmlibrary800
service name:       example1234
:w
```

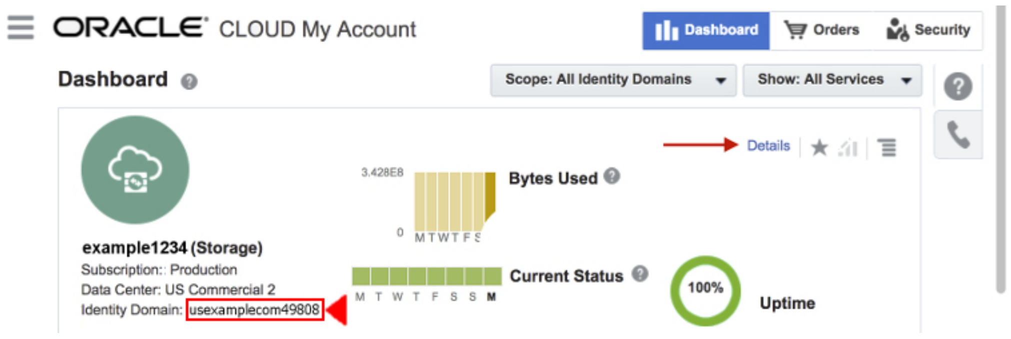


example1234 (Storage)
Subscription:: Production
Data Center: US Commercial 2
Identity Domain: usexamplecom49808

3. Make a note of the Identity Domain ID displayed in the **Identity Domain** field.

   In the example, the **Identity Domain** field displays the Identity Domain ID **usexamplecom49808**. We note this in the **hsm-ocloud-info.txt** text file:

```
hsmadm@wrkstn:~# vi hsm-ocloud-info.txt
service name:       example1234
identity domain id: usexamplecom49808
:w
```



example1234 (Storage)
Subscription:: Production
Data Center: US Commercial 2
Identity Domain: usexamplecom49808

4. Next, create administrative user accounts for Oracle HSM in this identity domain.

Create Oracle Storage Cloud Service User Accounts for Oracle HSM

For each Oracle HSM cloud library that you plan to configure, create an Oracle Storage Cloud storage administrative user account. The account lets Oracle HSM log in to the Identity Domain that governs the service and authorizes creation, use, and deletion of cloud-based storage media. Proceed as follows:

1. Consult Chapter 6 of *Getting Started with Oracle Cloud*, "Adding Users and Assigning Roles".
2. In the same document, review the background information in "Oracle Cloud User Roles and Privileges" and "About Adding Users".
3. Make a note of the first name, last name, and email address of the person in your organization that will be responsible for the Oracle HSM software's use of the Oracle Storage Cloud service.

   The email address can be the named individual's address or, optionally, a valid administrative alias used by the individual, as shown in the examples:

   ```
   John Doe, jdoe@example.com
   John Doe, hsmadms@example.com
   ```

4. Create a service user account for Oracle HSM. Follow the instructions in the "Creating a User and Assigning a Role" section of *Getting Started with Oracle Cloud*, and make a note of the particulars.

   You can use the responsible party's email address as the user name, if desired. But in the example, we create a new user name, **hsmlibrary800**, instead of using **jdoe@example.com** or **hsmadms@example.com**. We make a note of this choice in the text file **hsm-cloud-info.txt**, save the file, and close the editor:

   ```
   hsmadm@wrkstn:~# vi hsm-ocloud-info.txt
   service name:       example1234
   identity domain id: usexamplecom49808
   user name:          hsmlibrary800
   :wq
   hsmadm@wrkstn:~#
   ```

5. If the Oracle Storage Cloud service is metered, assign the Oracle HSM user the *service-name*.**Storage_Administrator** role.
6. If the Oracle Storage Cloud service is not metered, assign the Oracle HSM user the *service-instance-name*.**Storage_Administrator** role.
7. Save the information that you have recorded. You will need it when you configure archival storage in Chapter 6, "Configuring the Basic File System".

   Backing up this information for the long term is also a good idea. So, in the example, we copy the file to a backup Oracle HSM configuration repository that we have configured on a file system that can be mounted from the Oracle HSM hosts (for more information on creating such a repository, see "Create a Backup Location for Your Oracle HSM Configuration" on page 13-1):

   ```
   hsmadm@wrkstn:~# cp hsm-ocloud-info.txt /sam_config/cloud/hsm-cloud-info.txt
   hsmadm@wrkstn:~#
   ```

8. If your configuration includes archival tape storage, configure it now.
9. If you are configuring a high-availability file system, see "Configuring Storage for High-Availability File Systems".

10. Otherwise, go to "Installing Oracle HSM and QFS Software" on page 4-1.

Use Oracle OpenStack Swift with Another Instance of Oracle HSM

Setting up OpenStack Swift on Solaris is outside the scope of this document (for full information consult the documentation published at: **`https://docs.oracle.com/cd/E65465_01/index.html`**). But Oracle HSM will be able to use Solaris OpenStack cloud storage once your administrator configures domains, user accounts, and URLs to suit your requirements.

1. Have your OpenStack Swift administrator set up Oracle OpenStack Swift on another system.
2. Install Oracle HSM on that system and configure file systems for OpenStack to use. See Appendix D, "OpenStack Swift on Oracle HSM File Systems" for more details.
3. Have your OpenStack Swift administrator set up a URL and administrative user account for each cloud library that you plan to configure.
4. Make a note of the domain ID of the assigned domain.

   In the example, the domain ID is **`ohsm`**. We use the vi text editor to make a note of the ID value in a text file called **`hsm-pcloud-info.txt`**:

   ```
   hsmadm@wrkstn:~# vi hsm-pcloud-info.txt
   domain id: ohsm
   :w
   ```

5. Make a note of the administrative user ID assigned to each cloud library.

   In the example, the user ID for the cloud library is **`hsmlibrary810`**:

   ```
   hsmadm@wrkstn:~# vi hsm-pcloud-info.txt
   domainID: ohsm
   userID: hsmlibrary810
   :w
   ```

6. Make a note of the Uniform Resource Locator (URI) where the Oracle HSM user will log in to the domain.

   In the example, the URL is **`https://ohsmcl810.pcloud.example.com`**:

   ```
   hsmadm@wrkstn:~# vi hsm-pcloud-info.txt
   domainID: ohsm
   userID: hsmlibrary810
   url: https://ohsmcl810.cloud.example.com
   :wq
   hsmadm@wrkstn:~#
   ```

7. Save the information that you have recorded. You will need it when you configure archival storage in Chapter 6, "Configuring the Basic File System".

   Backing up this information for the long term is also a good idea. So, in the example, we copy the file to a backup Oracle HSM configuration repository that we have configured on a file system that can be mounted from the Oracle HSM hosts (for more information on creating such a repository, see "Create a Backup Location for Your Oracle HSM Configuration" on page 13-1):

   ```
   hsmadm@wrkstn:~# cp hsm-pcloud-info.txt /sam_config/cloud/hsm-pcloud-info.txt
   hsmadm@wrkstn:~#
   ```

8. If your configuration includes archival tape storage, configure it now.

9. If you are configuring a high-availability file system, see "Configuring Storage for High-Availability File Systems" on page 3-16.
10. Otherwise, go to Chapter 4, "Installing Oracle HSM and QFS Software".

Configure Archival Tape Storage

Carry out the following tasks:

- Determine the order in which drives are installed in the library.
- Configure any direct-attached libraries that are included in the Oracle HSM configuration.

Determine 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 Oracle HSM master configuration file (**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 **/var/adm/messages** file.

For each Oracle HSM metadata server and datamover host, determine the drive order by carrying out the tasks listed below:

- Gather identifying information for the drives from both the library and the Solaris host.
- Then map the drives to Solaris device names, following the procedure appropriate for either a direct- or ACSLS-attached library.

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**.

   ```
   root@solaris:~#
   ```

4. List the Solaris logical device names in **/dev/scsi/changer/**, 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@solaris:~# ls -l /dev/rmt/ > /root/device-mappings.txt
   ```

5. Now, map the Solaris device names to the drives in your direct- 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**.

   ```
   root@solaris:~#
   ```

2. In a text editor, open the device mappings file that you created in the procedure "Gather Drive Information for the Library and the Solaris Host" on page 3-10, and organize it 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 headers and space for library device information:

```
root@solaris:~# vi /root/device-mappings.txt
LIBRARY SOLARIS      SOLARIS
DEVICE  LOGICAL      PHYSICAL
NUMBER  DEVICE       DEVICE
------- ----------   -------------------------------------------
        /dev/rmt/0 -> ../../devices/pci@1f,4000/scsi@2,1/st@2,0:
        /dev/rmt/1 -> ../../devices/pci@1f,4000/scsi@4,1/st@5,0:
        /dev/rmt/2 -> ../../devices/pci@1f,4000/scsi@4,1/st@6,0:
        /dev/rmt/3 -> ../../devices/pci@1f,4000/scsi@4/st@1,0:
lrwxrwxrwx 1 root root 40 Mar 18 2014 /dev/rmt/4 ->
../../devices/pci@1f,4000/scsi@4/st@2,0:
```

3. On the library, make sure that all drives are empty.

4. 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.

5. 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@solaris:~# mt -f /dev/rmt/0 status
/dev/rmt/0: no tape loaded or drive offline
root@solaris:~# 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
```

6. In the device-mappings file, 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, enter **1** in the **LIBRARY DEVICE NUMBER** field of the row for **/dev/rmt/1**:

```
root@solaris:~# vi /root/device-mappings.txt
LIBRARY SOLARIS      SOLARIS
DEVICE  LOGICAL      PHYSICAL
NUMBER  DEVICE       DEVICE
------- ----------   -------------------------------------------
        /dev/rmt/0 -> ../../devices/pci@1f,4000/scsi@2,1/st@2,0:
   1    /dev/rmt/1 -> ../../devices/pci@1f,4000/scsi@4,1/st@5,0:
        /dev/rmt/2 -> ../../devices/pci@1f,4000/scsi@4,1/st@6,0:
        /dev/rmt/3 -> ../../devices/pci@1f,4000/scsi@4/st@1,0:
:w
```

7. Unload the tape.

8. Repeat this procedure until the device-mappings file holds Solaris logical device names for all devices that the library assigns to the Oracle HSM host. Then save the file and close the editor.

```
root@solaris:~# vi /root/device-mappings.txt
LIBRARY SOLARIS      SOLARIS
DEVICE  LOGICAL      PHYSICAL
NUMBER  DEVICE       DEVICE
------- ----------   -------------------------------------------
   2    /dev/rmt/0 -> ../../devices/pci@1f,4000/scsi@2,1/st@2,0:
   1    /dev/rmt/1 -> ../../devices/pci@1f,4000/scsi@4,1/st@5,0:
   3    /dev/rmt/2 -> ../../devices/pci@1f,4000/scsi@4,1/st@6,0:
   4    /dev/rmt/3 -> ../../devices/pci@1f,4000/scsi@4/st@1,0:
:wq
root@solaris:~#
```

9. Keep the mappings file.

   You will need the information when you configure file systems, and you may wish to include it when you back up the finished Oracle HSM configuration.

10. Next, go to "Configure Direct-Attached Libraries" on page 3-14.

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`**.

```
root@solaris:~#
```

2. In a text editor, open the device mappings file that you created in the procedure "Gather Drive Information for the Library and the Solaris Host" on page 3-10, and organize it 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 headers and space for library device information:

```
root@solaris:~# vi /root/device-mappings.txt
LOGICAL DEVICE  DEVICE SERIAL NUMBER  ACSLS DEVICE ADDRESS
--------------  --------------------  ----------------------------------
/dev/rmt/0
/dev/rmt/1
/dev/rmt/2
/dev/rmt/3
```

3. For each logical device name listed in **`/dev/rmt/`**, display the device serial number. Use the command **`luxadm display /dev/rmt/`***`number`*, where *`number`* identifies the drive in **`/dev/rmt/`**.

   In the example, we obtain the serial number **`HU92K00200`** for device **`/dev/rmt/0`**:

```
root@solaris:~# 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@solaris:~#
```

4. Enter the serial number in the corresponding row of the **device-mappings.txt** file.

   In the example, we record the serial number of device **/dev/rmt/0**, **HU92K00200** in the row for logical device **/dev/rmt/0**.

```
root@solaris:~# vi /root/device-mappings.txt
LOGICAL DEVICE  DEVICE SERIAL NUMBER  ACSLS DEVICE ADDRESS
--------------  --------------------  ----------------------------------
/dev/rmt/0      HU92K00200
/dev/rmt/1
/dev/rmt/2
/dev/rmt/3
:wq
root@solaris:~#
```

5. Repeat the two preceding steps until you have identified the device serial numbers for all logical devices listed in **/dev/rmt/** and recorded the results in the **device-mappings.txt** file.

   In the example, there are four logical devices:

```
root@solaris:~# vi /root/device-mappings.txt
LOGICAL DEVICE  DEVICE SERIAL NUMBER  ACSLS DEVICE ADDRESS
--------------  --------------------  ----------------------------------
/dev/rmt/0      HU92K00200
/dev/rmt/1      HU92K00208
/dev/rmt/2      HU92K00339
/dev/rmt/3      HU92K00289
:w
root@solaris:~#
```

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    12    331000049255
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
```

7. Record each ACSLS drive address in the corresponding row of the **device-mappings.txt** file. Save the file, and close the text editor.

```
root@solaris:~# vi /root/device-mappings.txt
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
```

8. Keep the mappings file.

   You will need the information for configuring the file systems (Chapter 6), and you may wish to include it when you back up the Oracle HSM configuration (Chapter 13).

9. You configure Oracle StorageTek ACSLS network-attached libraries when you configure archiving file systems. So, if you are planning a high-availability file system, go to "Configuring Storage for High-Availability File Systems" on page 3-16. Otherwise, go to "Installing Oracle HSM and QFS Software" on page 4-1.

Configure Direct-Attached Libraries

To configure a direct-attached tape library, you must physically connect the hardware and, in some cases, configure the SCSI driver (Oracle HSM controls library robotics via the generic **sgen** driver rather than the **samst** driver used by SAM-QFS prior to release 5.4). Proceed as follows:

1. Physically connect the library and drives to the Oracle HSM server host.

2. If you are installing Oracle HSM for the first time or upgrading an Oracle HSM or SAM-QFS 5.4 configuration on Solaris 11, stop once the hardware has been physically connected.

   Under Solaris 11, **sgen** is the default SCSI driver, so the Oracle HSM installation software can automatically update driver aliases and configuration files.

3. If you are installing Oracle HSM on a Solaris 10 system, see if one of the driver aliases in the list below is assigned to the **sgen** driver. Use the command **grep scs.*,08 /etc/driver_aliases**.

   The **sgen** driver may be assigned any of the following aliases:

   - **scsa,08.bfcp"** and/or **scsa,08.bvhci**
   - **scsiclass,08**

   In the example, Solaris is using the **scsiclass,08** alias for the **sgen** driver:

```
root@solaris:~# grep scs.*,08 /etc/driver_aliases
sgen "scsiclass,08"
root@solaris:~#
```

4. If the **grep** command returns **sgen "*alias*"**, where *alias* is an alias in the list above, the **sgen** driver is installed and correctly assigned to the alias. So, if you are configuring a high-availability file system, see Configuring Storage for High-Availability File Systems. Otherwise go to "Installing Oracle HSM and QFS Software" on page 4-1.

5. If the **grep** command returns *some-driver* **"*alias*"**, where *some-driver* is some driver other than **sgen** and where *alias* is one of the aliases listed above, then the alias is already assigned to the other driver. So create a path-oriented alias for the **sgen** driver.

6. If the command **grep scs.*,08 /etc/driver_aliases** does not return any output, the **sgen** driver is not installed. So install it. Use the command **add_drv -i scsiclass,08 sgen**.

   In the example, the **grep** command does not return anything. So we install the **sgen** driver:

```
root@solaris:~# grep scs.*,08 /etc/driver_aliases
root@solaris:~# add_drv -i scsiclass,08 sgen
```

7. If the command **add_drv -i scsiclass,08 sgen** returns the message **Driver (sgen) is already installed**, the driver is already installed but not attached. So attach it now. Use the command **update_drv -a -i scsiclass,08 sgen**.

   In the example, the **add_drv** command indicates that the driver is already installed. So we attach the driver:

```
root@solaris:~# add_drv -i scsiclass,08 sgen
Driver (sgen) is already installed.
root@solaris:~# update_drv -a -i scsiclass,08 sgen
```

8. If the command **grep scs.*,08 /etc/driver_aliases** shows that the alias **scsiclass,08** is assigned to the **sgen** driver, the driver is properly configured.

```
root@solaris:~# grep scs.*,08 /etc/driver_aliases
sgen "scsiclass,08"
root@solaris:~#
```

9. If you are configuring a high-availability file system, see "Configuring Storage for High-Availability File Systems" on page 3-16.
10. Otherwise, go to "Installing Oracle HSM and QFS Software" on page 4-1.

Create a Path-Oriented Alias for the sgen Driver

If the expected **sgen** alias is already assigned to another driver, you need to create a path-oriented alias that attaches the specified library using **sgen**, without interfering with existing driver assignments. Proceed as follows:

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

```
root@solaris:~#
```

2. Display the system configuration. Use the command **cfgadm -vl**.

   Note that **cfgadm** output is formatted using a two-row header and two rows per record:

```
root@solaris:~# cfgadm -vl
Ap_Id                Receptacle  Occupant     Condition  Information  When
Type        Busy  Phys_Id
c3                   connected   configured   unknown    unavailable
scsi-sas    n     /devices/pci@0/pci@0/pci@2/scsi@0:scsi
c5::500104f0008e6d78 connected   configured   unknown    unavailable
med-changer y     /devices/pci@0/.../SUNW,qlc@0,1/fp@0,0:fc::500104f0008e6d78
...
root@solaris:~#
```

3. In the output of **cfgadm -vl**, find the record for the library. Look for **med-changer** in the **Type** column of the second row of each record.

   In the example, we find the library in the second record:

```
root@solaris:~# cfgadm -vl
Ap_Id                Receptacle  Occupant     Condition  Information  When
Type        Busy  Phys_Id
c3                   connected   configured   unknown    unavailable
scsi-sas    n     /devices/pci@0/pci@0/pci@2/scsi@0:scsi
c5::500104f0008e6d78 connected   configured   unknown    unavailable
med-changer y     /devices/pci@0/.../SUNW,qlc@0,1/fp@0,0:fc::500104f0008e6d78
...
root@solaris:~#
```

4. Get the physical path that will serve as the new path-oriented alias. Remove the substring **/devices** from the entry in the **Phys_Id** column in the output of **cfgadm -vl**.

   In the example, the **Phys_Id** column of the media changer record contains the path **/devices/pci@0/pci@0/pci@9/SUNW,qlc@0,1/fp@0,0:fc::500104f0008e6d78**, so we select the portion of the string following **/devices/** as the alias (note that this physical path has been abbreviated to fit the space available below):

   ```
   root@solaris:~# grep scsiclass,08 /etc/driver_aliases
   sdrv "scsiclass,08"
   root@solaris:~# cfgadm -vl
   Ap_Id                Receptacle  Occupant     Condition Information  When
   Type        Busy  Phys_Id
   c3                   connected   configured   unknown   unavailable
   scsi-sas    n     /devices/pci@0/pci@0/pci@2/scsi@0:scsi
   c5::500104f0008e6d78 connected   configured   unknown   unavailable
   med-changer y     /devices/pci@0/.../SUNW,qlc@0,1/fp@0,0:fc::500104f0008e6d78
   ...
   root@solaris:~#
   ```

5. Create the path-oriented alias and assign it to the **sgen** driver. Use the command **update_drv -d -i '"/***path-to-library***"' sgen**, where *path-to-library* is the path that you identified in the preceding step.

   In the example, we use the library path to create the path-oriented alias **'"/pci@0/pci@0/pci@9/SUNW,qlc@0,1/fp@0,0:fc::500104f0008e6d78"'** (note the single and double quotation marks):

   ```
   root@solaris:~# update_drv -d -i
   '"/pci@0/pci@0/pci@9/SUNW,qlc@0,1/fp@0,0:fc::500104f0008e6d78"' sgen
   root@solaris:~#
   ```

   The library has now been configured using the **sgen** driver

6. If you are configuring a high-availability file system, configure the storage now.

7. Otherwise, go to "Installing Oracle HSM and QFS Software" on page 4-1.

Configuring Storage for High-Availability File Systems

High-availability file systems require redundant hardware and multiple, independent I/O paths, so that single-point hardware failures do not leave the file system unreachable. Carry out the following tasks:

- Configure Solaris Cluster nodes for multipath I/O.
- Configure any Linux clients for multipath I/O.

Configure Solaris Cluster Nodes for Multipath I/O

To configure a high-availability shared file system, you must take care to follow the recommendations in the hardware administration manual for your version of the Solaris Cluster software. Provide both redundant primary storage devices and redundant I/O paths.

Store file system data and metadata on hardware RAID devices or on Solaris Volume Manager (SVM) software RAID volumes, as specified in the documentation for the Solaris Cluster **SUNW.HAStoragePlus** resource type (**SUNW.HAStoragePlus** does *not* support ZFS **zvols**). Place Oracle HSM metadata and configuration files on RAID-10 volume groups or mirrored SVM volumes. Place file-system data on hardware-controlled RAID-10 or RAID-5 volume groups or on mirrored SVM volumes. If you plan on using SVM volumes, you should note that, starting with Solaris 11, Solaris Volume Manager is no longer included with the Solaris operating system and is no longer installed by default. So you must download and install the SVM software packages separately.

Make sure that Storage Area Network connections cannot suffer single-point failures. Install multiple host bus adapters (HBAs) on each cluster node. Configure the Storage Area Network (SAN) with multiple interconnects and redundant switches. Manage path failover with Oracle Solaris I/O multipathing software (for additional details, see the *Oracle Solaris SAN Configuration and Multipathing Guide* in the Oracle Solaris customer documentation library and the **stmsboot** man page).

Configure Linux Clients for Multipath I/O

On Linux clients, configure redundant storage devices for path failover using the Device Mapper Multipath (DMM) software package. The DMM software manages all of the host bus adapters, cables, switches, and controllers that link a host and a storage device as a single, virtual I/O device, the *multipath*.

- Install the Device Mapper Multipath software package.
- Configure the Device Mapper Multipath software.

Install the Device Mapper Multipath Software Package

Follow the instructions below to configure a client that runs Oracle Linux 6.*x* (for other versions of Linux, consult the vendor's documentation).

1. Log in to the Linux host as **root**.

```
[root@linux ~]#
```

2. Change to the **/etc/yum.repos.d** subdirectory and list the directory contents.

```
[root@linux ~]# cd /etc/yum.repos.d
[root@linux ~]# ls -l
total 4
-rw-r--r--. 1 root root 1707 Jun 25  2012 public-yum-ol6.repo
[root@linux ~]#
```

3. If the **/etc/yum.repos.d** subdirectory does not contain a **public-yum-ol6.repo** file, download one from the Oracle YUM repository using the **wget** command.

```
[root@linux ~]# wget http://public-yum.oracle.com/public-yum-ol6.repo
-- 2013-02-25 12:50:32 --   http://public-yum.oracle.com/public-yum-ol6.repo
Resolving public-yum.oracle.com... 14 1.146.44.34
Connecting to public-yum.oracle.com|141.146.44.34|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2411 (2.4K) [text/plain]
Saving to: "public-yum-ol6.repo"
100%[=====================================>] 2,411   -- . - K/s   in 0.001s
2013-02-25 12:50:32 (3.80 MB/s) - "public-yum-ol6.repo" saved
[2411/2411]
[root@linux ~]#
```

4. Using a text editor, open the **public-yum-ol6.repo** file. Make sure that the first entry, **[ol6_latest]**, contains the line **enabled=1**.

In the example, we use the **vi** editor. The required line is present, so we close the file:

```
[root@linux ~]# vi public-yum-ol6.repo
[ol6_latest]
name=Oracle Linux $releasever Latest ($basearch)
baseurl=http://public-yum.oracle.com/repo/OracleLinux/OL6/latest/$basearch/
gpgkey=http://public-yum.oracle.com/RPM-GPG-KEY-oracle-ol6
gpgcheck=1
enabled=1
...
:q
[root@linux ~]#
```

5. Find the device mapper multipath software packages. Use the command **yum search multipath**.

```
[root@linux ~]# yum search multipath
Loaded plugins: refresh-packagekit, security
========================= N/S Matched: multipath ==========================
device-mapper-multipath.x86_64 : Tools to manage multipath devices using
                               : device-mapper
device-mapper-multipath-libs.x86_64 : The device-mapper-multipath modules and
                                    : shared library
  Name and summary matches only, use "search all" for everything.
[root@linux ~]#
```

6. Install the device mapper multipath software. Use the command **yum install device-mapper-multipath**. When prompted, enter **y** (yes) to accept the listed package and its dependencies.

```
[root@linux ~]# yum install device-mapper-multipath
Loaded plugins: refresh-packagekit, security
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package device-mapper-multipath.x86_64 0:0.4.9-56.el6_3.1 will be
installed
--> Processing Dependency: device-mapper-multipath-libs = 0.4.9-56.el6_3.1
for package: device-mapper-multipath-0.4.9-56.el6_3.1.x86_64
--> Processing Dependency: libmultipath.so()(64bit)
for package: device-mapper-multipath-0.4.9-56.el6_3.1.x86_64
--> Running transaction check
---> Package device-mapper-multipath-libs.x86_64 0:0.4.9-56.el6_3.1 will be
installed
--> Finished Dependency Resolution
Dependencies Resolved
============================================================================
 Package                      Arch   Version          Repository   Size
============================================================================
Installing:
 device-mapper-multipath      x86_64 0.4.9-56.el6_3.1  ol6_latest    96 k
Installing for dependencies:
 device-mapper-multipath-libs x86_64 0.4.9-56.el6_3.1  ol6_latest   158 k
Transaction Summary
============================================================================
Install       2 Package(s)
Total download size: 254 k
Installed size: 576 k
Is this ok [y/N]: y
Downloading Packages:
```

```
(1/2): device-mapper-multipath-0.4.9-56.el6_3.1.x86_64.r |  96 kB     00:00
(2/2): device-map
per-multipath-libs-0.4.9-56.el6_3.1.x86 | 158 kB     00:00
----------------------------------------------------------------------------
Total                                           104 kB/s | 254 kB     00:02
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : device-mapper-multipath-libs-0.4.9-56.el6_3.1.x86_64     1/2
  Installing : device-mapper-multipath-0.4.9-56.el6_3.1.x86_64          2/2
  Verifying  : device-mapper-multipath-0.4.9-56.el6_3.1.x86_64          1/2
  Verifying  : device-mapper-multipath-libs-0.4.9-56.el6_3.1.x86_64     2/2
Installed:
  device-mapper-multipath.x86_64 0:0.4.9-56.el6_3.1
Dependency Installed:
  device-mapper-multipath-libs.x86_64 0:0.4.9-56.el6_3.1
Complete!
[root@linux ~]#
```

7. Start the multipath daemon. Use the command **chkconfig multipathd on**.

```
[root@linux ~]# chkconfig multipathd on
[root@linux ~]#
```

8. Next, configure the Device Mapper Multipath software.

Configure the Device Mapper Multipath Software

You configure the Device Mapper Multipath software by editing the **/etc/multipath.conf** file. The file consists of a series of sections, each section contains a set of related attributes, values, and subsections:

- The **default** section configures the multipath software itself. It specifies the level of detail logged, defines failover behavior, and specifies the locations of required operating-system commands and directories.
- The **blacklist** section identifies devices that you need to exclude from the multipath configurations, such as local system disks. You can identify devices by World Wide Name/World Wide Identifier (WWN/WID) or by regular expressions that specify device node names or vendor and product device strings.
- The **blacklist_exceptions** section lets you specifically include devices in the multipath configuration when general rules in the **blacklist** section would otherwise exclude them.
- The **multipaths** section lets you define one or more **multipath** subsections, each of which applies a special configuration to a multipath that you specify by World Wide Name.
- The **devices** section lets you define one or more **device** subsections, each of which applies a special multipath configuration to a device

For detailed descriptions of the individual defaults, see the annotated, sample file **/usr/share/doc/device-mapper-multipath-0.4.9/multipath.conf.annotated**.

4 Installing Oracle HSM and QFS Software

Oracle HSM uses the Image Packaging System (IPS) that became standard with Oracle Solaris 11. IPS is a network-centric package management system that streamlines and coordinates installation, upgrade, and removal of software packages. It greatly simplifies patch management and eases deployment into production environments.

Using the Solaris Package Manager graphical desktop application or IPS terminal commands, administrators access an Oracle Solaris software repository and locate, download, and install the required software packages, while IPS automatically handles dependency checking and package validation. IPS makes changes to a snapshot of the system so that new software can be deployed non-disruptively, during a maintenance window. So changes can be rolled back, if necessary. Installations and updates can thus be applied safely to running, production systems.

- Check Installation Requirements
- Download the HSM Installation Package
- Check and Update the Solaris Operating System
- Install Solaris Cluster Software (High-Availability Configurations Only)
- Upgrade Shared Oracle HSM File Systems
- Install, Upgrade, or Downgrade Oracle HSM on an Oracle Solaris Host
- Install or Update Oracle HSM Client Software on a Linux Host
- Uninstalling Oracle HSM Software

Check Installation Requirements

For the latest information on installation requirements, including the supported versions of the Oracle Solaris and Linux operating systems, Oracle Cluster software, and other required or supported software packages, consult the Oracle HSM release notes, Oracle support services at **`support.oracle.com`**, and the Oracle HSM wiki pages at **`wikis.oracle.com/display/hsmqfs/Home`**.

Download the HSM Installation Package

Download the installation package for your level of HSM. The download method depends on the version.

- To Download Versions 6.1.3 and Below
- To Download Versions 6.1.4 and Above

To Download Versions 6.1.3 and Below

For versions 6.1.3 and below, use the Oracle Software Delivery Cloud.

1. Go to: `edelivery.oracle.com`
2. Sign in, or register.
3. Search for "HSM". Locate the correct version and click **Add to Cart**.
4. Click **Check Out** in the upper right.
5. Select a language/platform. Click **Continue**.
6. Read and accept the terms. Click **Continue**.
7. Select the package and then click **Download**. Save to a directory accessible from all Oracle HSM hosts. For the examples in this chapter, we download the file to the `/hsmqfs` directory on a network file server named `sw_install`
8. While still on the download page, click **View Digest**. Save these values for comparison to the download. For instructions on calculating checksums from a file, see the Solaris `dgst` and `md5` man pages.

To Download Versions 6.1.4 and Above

For versions 6.1.4 and above, use My Oracle Support.

1. Go to: `support.oracle.com`
2. Sign in, or register.
3. Click the **Patches & Updates** tab.
4. Within the Patch Search panel, click the **Search** tab.
5. Click **Product or Family (Advanced)**.
6. In the Product field, enter **HSM**. Select **Oracle Hierarchical Storage Manager (HSM) and StorageTek QFS Software** from the product list.
7. Select the **Include all products in family** checkbox.
8. In the Release field, select either:
   - Oracle Hierarchical Storage Manager (HSM) 6.1.4.0.0
   - StorageTek QFS Software 6.1
9. Click **Search**.
10. Within the Patch Advanced Search Results panel, click the link in the Patch Name column (this will likely be a number, such as 87654321).
11. Click **Download**.
12. Within the File Download pop-up, click the link for the .zip file. Save it to a directory accessible from all Oracle HSM hosts.

Unzip the Download

Once you have downloaded the ZIP file, unzip it in the local directory.

In the example, we unzip the Oracle Hierarchical Storage Manager and StorageTek QFS Software zip file in the `/hsmqfs` subdirectory and then list the contents:

```
root@sw_install:~# cd /hsmqfs
root@sw_install:~# unzip Q12345-02.zip
```

```
root@sw_install:~# ls Q12345-02/
./ COPYRIGHT.txt linux.iso README.txt
../ iso.md5 Oracle-HSM_6.1-04/
root@sw_install:~# ls Oracle-HSM_6.1-04/
total 42
./ COPYRIGHT.txt linux1/ solaris_sparc/
../ README.txt linux2/ solaris_x64/
```

Check and Update the Solaris Operating System

Oracle HSM 6.1.4 requires Solaris 11. If running Solaris 11.4, it must be SRU 9 or later. Solaris 10 and earlier releases are no longer supported. Make sure that all Solaris hosts have been upgraded before proceeding:

1. Log into the host as **root**.

   ```
   root@hsmhost:~#
   ```

2. Display the Solaris version number. Use the command **uname -v**.

   In the example, the host is running Solaris 11.4:

   ```
   root@hsmhost:~# uname -v
   11.4.6.4.0
   root@hsmhost:~#
   ```

3. If the **uname -v** command does not return **11.***x*, upgrade the host operating system using the procedures documented in the *Oracle Solaris 11 Information Library*, which is available from the Oracle Help Center at: https://docs.oracle.com/en/operating-systems/solaris.html.
4. Repeat the three preceding steps until all host have been checked and, if necessary, upgraded.
5. If you are preparing a high-availability file-system, go to "Install Solaris Cluster Software (High-Availability Configurations Only)".
6. If you are upgrading a multi-host, shared file system, go to "Upgrade Shared Oracle HSM File Systems".
7. Otherwise, go directly to "Install, Upgrade, or Downgrade Oracle HSM on an Oracle Solaris Host" on page 4-6.

Install Solaris Cluster Software (High-Availability Configurations Only)

1. On each host, install the Oracle Solaris Cluster and **SUNW.HAStoragePlus** data service software, as described in the installation and data-service administration documents in the online *Information Library* for the Solaris Cluster software.
2. Then go to "Install, Upgrade, or Downgrade Oracle HSM on an Oracle Solaris Host" on page 4-6.

Upgrade Shared Oracle HSM File Systems

If you are upgrading the software for a shared file-system that needs to remain available during the upgrade process, consider a *rolling upgrade*. When one or more potential metadata servers are configured, in addition to the active server, you can update an inactive server, activate the updated server, and then configure and re-activate the primary server before upgrading remaining potential metadata servers and clients. This rolling upgrade process keeps an active Oracle HSM metadata server available at all times, so that file-system data remains accessible to clients.

To perform a rolling upgrade, carry out the following tasks:

- Upgrade any significantly older releases of Oracle HSM.
- Carry out rolling upgrades.

Upgrade Any Significantly Older Releases of Oracle HSM

At any given time, the Oracle HSM software on the metadata server and clients of a shared file system must be, at most, one release apart. If your shared file-system configuration includes hosts that are running Oracle HSM (or SAM-QFS) software that is more than one release behind the targeted upgrade release, you cannot upgrade to the desired release until you carry out corrective action.

1. If any client hosts are not running the same release of the Oracle HSM (or SAM-QFS) software as the metadata server, upgrade them to the release used on the server before proceeding.
2. If the Oracle HSM (or SAM-QFS) software on the active metadata server is more than one release behind the targeted upgrade release and if the file system *needs to remain mounted* during the upgrade, repeatedly carry out the rolling upgrades, one release level at a time, until all hosts are fully up to date.
3. If the Oracle HSM (or SAM-QFS) software on the active metadata server is more than one release behind the targeted upgrade release and if the file system *does not need to remain mounted* during the upgrade, do not attempt a rolling upgrade. Stop the archiving and staging processes, unmount the file system, and upgrade each host individually, as described in "Install, Upgrade, or Downgrade Oracle HSM on an Oracle Solaris Host" on page 4-6 or "Install or Update Oracle HSM Client Software on a Linux Host" on page 4-15.

Carry Out the Rolling Upgrade

1. Make sure that you upgrade any significantly older releases of Oracle HSM before continuing!

   If any host is more than one release behind the targeted upgrade release when you attempt a rolling upgrade, the upgrade will fail, at best leaving the file systems in an inconsistent state.

2. Log in to the currently active (first) metadata server as **`root`**. Then log in to the currently potential (second) metadata server, also 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 inactive, potential metadata server, **`mds2`**:

   ```
   root@mds1:~#

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

3. Upgrade the currently inactive, second metadata server. Install the updated Oracle HSM software using the procedures in "Install, Upgrade, or Downgrade Oracle HSM on an Oracle Solaris Host" on page 4-6 or "Install or Update Oracle HSM Client Software on a Linux Host" on page 4-15.

4. Once the upgrade steps are complete, prepare to activate the second server. If the first, active metadata server mounts an Oracle HSM or SAM-QFS archiving file system, stop any new archiving and staging activity, idle media drives, and wait for current jobs to finish. Then stop the library-control daemon.

   For full description of how to stop archiving activity, see the *Oracle Hierarchical Storage Manager and StorageTek QFS Software Maintenance and Administration Guide*.

   ```
   root@mds1:~# samcmd aridle
   root@mds1:~# samcmd stidle
   root@mds1:~# samcmd 901 idle
   ...
   root@mds1:~# samcmd a
    ...
   Waiting for :arrun
   root@mds1:~# samcmd r
   ...
   ty   eq status      act  use  state vsn
   li  801 ---------p    0   0%  off
           empty
   ...
   root@mds1:~# samd stop
   root@mds1:~#
   ```

5. On the second metadata server, load the Oracle HSM configuration files and start Oracle HSM processes. Use the command **samd config**.

   ```
   root@mds2:~# samd config
   root@mds2:~#
   ```

6. On the second metadata server, mount the Oracle HSM file system.

   ```
   root@mds2:~# mount sharefs1
   root@mds2:~#
   ```

7. Activate the newly updated second metadata server. From the second metadata server, issue the command **samsharefs -s** *server file-system*, where *server* is the hostname of the newly updated 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 **sharefs1**:

   ```
   root@mds2:~# samsharefs -s mds2 sharefs1
   root@mds2:~#
   ```

8. Upgrade the now-inactive first metadata server. Install the updated Oracle HSM software using the procedures in "Install, Upgrade, or Downgrade Oracle HSM on an Oracle Solaris Host" on page 4-6 or "Install or Update Oracle HSM Client Software on a Linux Host" on page 4-15.

9. Once the upgrade steps are complete, prepare to re-activate the first metadata server. If the currently active second metadata server mounts an Oracle HSM archiving file system, stop any new archiving and staging activity, idle media drives, and wait for current jobs to finish. Then stop the library-control daemon.

   ```
   root@mds2:~# samcmd aridle
   root@mds2:~# samcmd stidle
   ...
   root@mds2:~# samd stop
   root@mds2:~#
   ```

10. On the first metadata server, load the Oracle HSM configuration files and start Oracle HSM processes. Use the command **samd config**.

```
root@mds1:~# samd config
root@mds1:~#
```

11. On the first metadata server, mount the Oracle HSM file system.

```
root@mds1:~# mount sharefs1
root@mds1:~#
```

12. Re-activate the first metadata server. From the first metadata server, issue the command **samsharefs -s** *server file-system*, where *server* is the hostname 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 **mds1** and the file system name is **sharefs1**:

```
root@mds1:~# samsharefs -s mds1 sharefs1
root@mds1:~#
```

13. Update the remaining clients. Install the updated Oracle HSM software using the procedures in "Install, Upgrade, or Downgrade Oracle HSM on an Oracle Solaris Host" on page 4-6 or "Install or Update Oracle HSM Client Software on a Linux Host" on page 4-15..

14. Stop here. The upgrade is complete.

Install, Upgrade, or Downgrade Oracle HSM on an Oracle Solaris Host

To install, upgrade, or downgrade Oracle HSM packages on a Solaris host, start by carrying out the following tasks:

- Prepare the host for software changes.
- Locate the packages for your host architecture.

Then carry out the installation task that best fits your situation:

- If you are installing new software and the host operating system is Solaris 11 or later, use the Solaris Image Packaging System (IPS) command **pkg install**.
- If you are upgrading or downgrading software that was installed using the IPS command **pkg install**, use the Image Packaging System (IPS) command **pkg update**.
- If you are installing new software on a Solaris 10 host, use the SVR4 **pkgrm** and **pkgadd** commands.
- If you are upgrading software that was installed using the SVR4 command **pkgadd**, use the SVR4 **pkgrm** and **pkgadd** commands.

Prepare the Host for Software Changes

1. If Oracle HSM software is not currently installed on the host system, go to "Locate the Packages for Your Host Architecture" on page 4-9.

2. Otherwise, log in to the host as **root**.

```
root@solarishosthost:~#
```

3. If Oracle HSM software is currently installed on the host system, 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@solarishosthost:~# samcmd aridle
   root@solarishosthost:~#
   ```

4. 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@solarishosthost:~# samcmd stidle
   root@solarishosthost:~#
   ```

5. 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@solarishosthost:~# samcmd a
   Archiver status samcmd     6.1.4 10:20:34 Feb 20 2015
   samcmd on host
   sam-archiverd:  Waiting for :arrun
   sam-arfind: ...
   Waiting for :arrun
   ```

6. 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@solarishosthost:~# samcmd u
   Staging queue samcmd      6.1.4  10:20:34 Feb 20 2015
   samcmd on solaris.demo.lan
   Staging queue by media type: all
   sam-stagerd:  Waiting for :strun
   root@solarishosthost:~#
   ```

7. Idle all removable media drives before proceeding further. For each drive, use the command **`samcmd idle`** *`equipment-number`*, 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@solarishosthost:~# samcmd idle 801
   root@solarishosthost:~# samcmd idle 802
   root@solarishosthost:~# samcmd idle 803
   root@solarishosthost:~# samcmd idle 804
   root@solarishosthost:~#
   ```

8. 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@solarishosthost:~# samcmd r
   Removable media samcmd     6.1.4 10:37:09 Feb 20 2014
   samcmd on host
   ```

```
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@solarishosthost:~#
```

9. 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@solarishosthost:~# samd stop
root@solarishosthost:~#
```

10. If file systems are shared through NFS or SMB/CIFS, unshare the file systems. On the metadata server, use the command **unshare** *mount-point*, where *mount-point* is the mount point directory of the Oracle HSM file system.

    In the first example, we stop NFS sharing of the Oracle HSM standalone file system **hsmqfs1**.

```
root@solarishosthost:~# unshare /hsmqfs1
root@solarishosthost:~#
```

    In the second example, we stop NFS sharing of the Oracle HSM shared file system **samqfs2**:

```
root@solarishostserver:~# unshare /hsmqfs2
root@solarishostserver:~#
```

11. Unmount all Oracle HSM file systems.

    In the first example, we unmount the unshared, standalone file system **hsmqfs1**:

```
root@solarishosthost:~# umount hsmqfs1
```

    In the second example, we unmount the shared file system **hsmqfs1**, first from the clients and then from the server, allowing **60** seconds for clients to unmount.

```
root@solarishostserver:~# ssh root@samqfs2client1
Password:
[root@solarishostclient1:~# umount /hsmqfs2
[root@solarishostclient1:~# exit
root@solarishostserver:~#

root@solarishostserver:~# ssh root@samqfs2client1
Password:
root@solarishostclient2:~# umount /hsmqfs2
root@solarishostclient2:~# exit
root@solarishostserver:~# umount -o await_clients=60 /sharefs2
```

12. If you currently have SAM-QFS 5.3 or earlier installed, uninstall all packages. Use the command **pkgrm SUNWsamfsu SUNWsamfsr** (**pkgrm SUNWqfsu SUNWqfsr** if only QFS is installed).

    Remove the packages in the order specified, starting with **SUNWsamfsu** and ending with **SUNWsamfsr**. In the example, we pipe the reply **yes** into the command so that all questions are automatically answered:

```
root@solarishosthost:~# yes | pkgrm SUNWsamfsu SUNWsamfsr
```

13. Next, locate the Oracle HSM packages for your host architecture.

Locate the Packages for Your Host Architecture

1. Log in to the Oracle HSM host as **`root`**.

   ```
   root@solarishost:~#
   ```

2. Change to the directory where the Oracle HSM download file was unpacked, and locate the subdirectory where packages for the desired version are stored.

   The initially released packages are stored in the **`Oracle_HSM_`***`release-number`* (or **`STK_QFS_`***`release-number`*) subdirectory, where *`release-number`* is a major and a minor release number, joined by a dot: **`Oracle_HSM_6.1.4+/`**. Patch releases (if any) are located in a similar subdirectory with an additional **`-`***`patch-number`* suffix, where *`patch-number`* is a two-digit patch sequence number: **`Oracle_HSM_6.1.4-01/`**.

   In the example we change to the download directory for the initial release of the software, **`Oracle_HSM_6.1.4/`** and list the contents:

   ```
   root@solarishost:~# cd /net/sw_install/hsmqfs/Oracle_HSM_6.1.4/
   root@solarishost:~# ls -1
   ./
   ../
   linux1/
   linux2/
   Notices/
   README.txt
   solaris_sparc/
   solaris_x64/
   ```

3. Change to the subdirectory that corresponds to your host architecture, either **`solaris_sparc/`** or **`solaris_x64/`**, and list the contents.

   In the example, we change to the **`solaris_sparc/`** subdirectory:

   ```
   root@solarishost:~# cd solaris_sparc/
   root@solarishost:~# ls -1
   ./
   ../
   S11.0/
   S11.4/
   S11.0_ips/
   S11.4_ips/
   ```

   IMPORTANT: Choose the correct directory:

   - S11.0 for Solaris 11.0 - 11.3
   - S11.4 for Solaris 11.4
   - S11.4 for SVr4 packages
   - S11.4_ips for IPS packages

4. When Solaris 11 or later is installed on the host, you can install, upgrade, or downgrade the software using the Image Packaging System. Go to one of the following:

- "Install the Software Using the Image Packaging System (IPS)" on page 4-10.
- "Upgrade or Downgrade the Software Using the Image Packaging System (IPS)" on page 4-11.

5. When Solaris 11 or later is installed on the host, you can also choose to install, upgrade, or downgrade the software using the **pkgadd** method. See "Upgrade or Downgrade the Software Using SVR4 `pkgrm` and `pkgadd` Commands" on page 4-14.
6. When Solaris 10 is installed on the host, you can install, upgrade, or downgrade the software using the **pkgadd** method. Go to "Upgrade or Downgrade the Software Using SVR4 `pkgrm` and `pkgadd` Commands" on page 4-14.

Install the Software Using the Image Packaging System (IPS)

In general, you should use Image Packaging System (IPS) commands to install, upgrade, or downgrade Oracle HSM software on hosts running Solaris 11 or later. For each host, including metadata servers and shared file-system clients (if any), proceed as follows:

1. If you have not already done so, locate the Oracle HSM packages for your host architecture.
2. Change to the repository directory for the Solaris 11 IPS packages, **`repo.samqfs/`**.

   In the example, we change to repository directory for Oracle HSM 6.1.4, **`Oracle_HSM_6.1.4/solaris_sparc/S11_ips/repo.samqfs`**:

```
root@solarishost:~# cd repo.samqfs/
root@solarishost:~#
```

3. To install both the Oracle Hierarchical Storage Manager and StorageTek QFS Software packages, use the command **`pkg install -g . --accept SUNWsamfs SUNWsamqassy`**, where **`.`** is the current directory (the repository) and **`SUNWsamfs`** and **`SUNWsamqassy`** are the Oracle HSM Image Packaging System package names.

```
root@solarishost:~# pkg install -g . --accept SUNWsamfs SUNWsamqassy
Creating plan
...
* The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
           Packages to install:   2
       Create boot environment:  No
Create backup boot environment: Yes
DOWNLOAD                                PKGS         FILES    XFER (MB)   SPEED
Completed                                2/2       520/520    21.4/21.4    0B/s
PHASE                                          ITEMS
Installing new actions                       693/693
Updating package state database                 Done
Updating image state                            Done
Creating fast lookup database                   Done
```

4. To install only the QFS Software packages, use the command **`pkg install -g . --accept SUNWqfs SUNWsamqassy`**, where `.` is the current directory (the repository) and **`SUNWqfs`** and **`SUNWsamqassy`** are the Oracle HSM Image Packaging System package names.

```
root@solarishost:~# pkg install -g . --accept SUNWqfs SUNWsamqassy
Creating plan
```

```
...
* The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
           Packages to install:   2
       Create boot environment:  No
Create backup boot environment: Yes
DOWNLOAD                                PKGS         FILES    XFER (MB)   SPEED
Completed                                2/2       520/520    21.4/21.4    0B/s
PHASE                                          ITEMS
Installing new actions                       693/693
Updating package state database                 Done
Updating image state                            Done
Creating fast lookup database                   Done
```

5. When the packages finish installing, run the post-installation script, **sam-qfs-post-install**. It is located in the **util/** subdirectory of the Oracle HSM installation directory (either **/opt/SUNWsamfs/** or **/opt/SUNWqfs/**).

   In the example, we run **/opt/SUNWsamfs/util/sam-qfs-post-install**:

```
root@solarishost:~# /opt/SUNWsamfs/util/sam-qfs-post-install
SUNWsamfs IPS package installed.

inquiry.conf may have been updated for this release.
...
root@solarishost:~#
```

6. Add the Oracle HSM directories **/opt/SUNWsamfs/bin** and **/opt/SUNWsamfs/sbin** (or **/opt/SUNWqfs/bin** and **/opt/SUNWqfs/sbin**) to the system **PATH** variable, if they are not already in path.
7. Add the Oracle HSM directory **/opt/SUNWsamfs/man** (or **/opt/SUNWqfs/man**) to the system **MANPATH** variable, if it is not already in the man path.
8. If the planned Oracle HSM configuration includes additional Solaris hosts, repeat this procedure from the beginning until the software is installed on all hosts.
9. If the planned Oracle HSM configuration includes Linux hosts as shared file-system clients, go to "Install or Update Oracle HSM Client Software on a Linux Host" on page 4-15.
10. Otherwise, go to "Using the `samsetup` Configuration Wizard" on page 5-1 or "Configuring the Basic File System" on page 6-1.

Upgrade or Downgrade the Software Using the Image Packaging System (IPS)

Use Image Packaging System (IPS) commands to upgrade or downgrade Oracle HSM software that was originally installed using IPS.

For each host, including metadata servers and shared file-system clients (if any), proceed as follows:

1. If you have not yet done so, locate the Oracle HSM packages for your host architecture.
2. To upgrade the Oracle Hierarchical Storage Manager and StorageTek QFS Software packages to the latest versions in the repository, use the command **pkg update -g . --accept SUNWsamfs SUNWsamqassy**, where **.** is the current directory

(the repository) and **SUNWsamfs** and **SUNWsamqassy** are the Oracle HSM Image Packaging System package names.

```
root@solarishost:~# pkg update -g . --accept SUNWsamfs SUNWsamqassy
...
root@solarishost:~#
```

3. To upgrade only the QFS Software packages to the latest versions in the repository, use the command **pkg update -g . --accept SUNWqfs SUNWsamqassy**, where `.` is the current directory (the repository) and **SUNWqfs** and **SUNWsamqassy** are the Oracle HSM Image Packaging System package names.

```
[host1]root@solarishost:~# pkg update -g . --accept SUNWqfs SUNWsamqassy
...
root@solarishost:~#
```

4. To downgrade the Oracle HSM packages or to upgrade them to a specified version, first obtain the fault managed resource identifier (FMRI) for the desired packages. Use the command **pkg info -r -g .** *package-name*, where **.** specifies the current directory and *package-name* is the name of the Oracle HSM package.

   In the example, Oracle HSM version 6.1.4.0 is installed on the host:

```
root@solarishost:~# samcmd l
Usage information samcmd      6.1.4.0 14:06:20 Feb  20 2015 ...
root@solarishost:~#
```

   We need to downgrade to SAM-QFS 5.4.6. So we run the **pkg info** commands for **SUNWsamfs** and **SUNWsamqassy** in the IPS repository for version 5.4.6, **Oracle_HSM_6.1.4/solaris_sparc/S11_ips/repo.samqfs**:

```
root@solarishost:~# pwd
/net/Oracle_HSM_6.1.4/solaris_sparc/S11_ips/repo.samqfs
root@solarishost:~# pkg info -r -g . SUNWsamfs
          Name: SUNWsamfs
       Summary: StorageTek SAM and StorageTek SAM-QFS software
   Description: StorageTek Storage and Archive Manager File System
      Category: System/File System
         State: Not installed
     Publisher: samqfs
       Version: 5.4
 Build Release: 5.11
        Branch: None
Packaging Date: Tue Jul 08 22:56:56 2014
          Size: 88.64 MB
          FMRI: pkg://hsmqfs/SUNWsamfs@5.4,5.11:20140708T225656Z

root@solarishost:~# pkg info -r -g . SUNWsamqassy
          Name: SUNWsamqassy
       Summary: StorageTek QFS and Storage Archive Manager SAM-QFS IPS assembly
services
   Description: SAM-QFS IPS Assembly Services
      Category: System/File System
         State: Installed
     Publisher: samqfs
       Version: 5.4
 Build Release: 5.11
        Branch: None
Packaging Date: Fri Sep 26 17:21:35 2014
          Size: 15.15 kB
          FMRI: pkg://hsmqfs/SUNWsamqassy@5.4,5.11:20140926T172135Z
```

```
root@solarishost:~#
```

5. Then, to downgrade the Oracle HSM packages or to upgrade them to a specified version, run the command **`pkg update -g . `***`fmri`*, where **`.`** specifies the current directory and *`fmri`* specifies the fault managed resource identifier of the desired software version.

   In the example, we specify the FMRIs of the 5.4.6 versions of the **`SUNWsamfs`** and **`SUNWsamqassy`** packages:

```
root@solarishost:~# pkg update -g . SUNWsamfs@5.4,5.11:20140708T225656Z
           Packages to update:   1
       Create boot environment:  No
Create backup boot environment: Yes
DOWNLOAD                                PKGS         FILES    XFER (MB)
 SPEED
Completed                               1/1        160/160    19.2/19.2  3.4M/s
PHASE                                          ITEMS
Updating modified actions                    172/172
Updating package state database                 Done
Updating package cache                           1/1
Updating image state                            Done
Creating fast lookup database                   Done
Updating package cache                           3/3
root@solarishost:~# pkg update -g . SUNWsamqassy@5.4,5.11:20140926T172135Z
...
root@solarishost:~#
```

6. Once the **`pkg update`** command has finished, restart the system. Use the Solaris **`reboot`** command.

```
root@solarishost:~# reboot
```

7. If the planned Oracle HSM configuration includes additional Solaris hosts, repeat this procedure from the beginning until the software has been updated or downgraded on all hosts.

8. If the planned Oracle HSM configuration includes Linux hosts as shared file-system clients, go to "Install or Update Oracle HSM Client Software on a Linux Host" on page 4-15.

Install the Software Using SVR4 `pkgrm` and `pkgadd` Commands

Use SVR4 package commands when you are installing Oracle HSM software on hosts that run Solaris 10 and when you are upgrading software that was originally installed using SVR4 commands.

For each Oracle HSM Solaris host, including metadata servers and shared file-system clients (if any), proceed as follows:

1. If you have not already done so, locate the Oracle HSM packages for your host architecture.

2. To install both the Oracle Hierarchical Storage Manager and StorageTek QFS Software packages, use the command **`pkgadd -d . SUNWsamfsr SUNWsamfsu`** and accept all defaults.

   Note that you must install the **`SUNWsamfsr`** package before installing the **`SUNWsamfsu`** package. In the example, we make sure that we are in the directory for our operating system, **`Oracle_HSM_6.1.4/solaris_sparc/S10`**. Then we pipe the reply **`yes`** into the command so that all questions are automatically answered:

```
root@solarishost:~# pwd
/net/Oracle_HSM_6.1.4/solaris_sparc/s10
root@solarishost:~# yes | pkgadd -d . SUNWsamfsr SUNWsamfsu
```

3. To install only the QFS Software packages, use the command **pkgadd -d . SUNWqfsr SUNWqfsu** and accept all defaults.

   Note that you must install the **SUNWqfsr** package before installing the **SUNWqfsu** package. In the example, we pipe the reply **yes** into the command so that all questions are automatically answered:

```
root@solarishost:~# yes | pkgadd -d . SUNWqfsr SUNWqfsu
```

4. If the planned Oracle HSM configuration includes Linux hosts as shared file-system clients, go to "Install or Update Oracle HSM Client Software on a Linux Host" on page 4-15.
5. Otherwise, go to Chapter 5, "Using the samsetup Configuration Wizard" or Chapter 6, "Configuring the Basic File System".

Upgrade or Downgrade the Software Using SVR4 pkgrm and pkgadd Commands

Use SVR4 package commands when you are upgrading or downgrading Oracle HSM software on hosts that run Solaris 10 and when you are upgrading or downgrading software that was originally installed using SVR4 commands.

For each Oracle HSM Solaris host, including metadata servers and shared file-system clients (if any), proceed as follows:

1. If you are downgrading the Oracle HSM software to SAM-QFS 5.3, start by restoring configuration files to the locations specified by the older software. Use the command **/opt/SUNWsamfs/sbin/backto 5.3**.

   The **backto** command restores files to their previous locations and formats. See the **backto** man page for more information.

   In the example, we convert Oracle HSM 6.1.4 configuration files for use with Oracle SAM 5.3:

```
root@solarishost:~# /opt/SUNWsamfs/sbin/backto 5.3 ...
root@solarishost:~#
```

2. Uninstall all Oracle HSM packages that are currently installed. Use the command **pkgrm SUNWsamfsu SUNWsamfsr** (**pkgrm SUNWqfsu SUNWqfsr** if only QFS is installed).

   Remove the packages in the order specified, starting with **SUNWsamfsu** and ending with **SUNWsamfsr**. In the example, we pipe the reply **yes** into the command so that all questions are automatically answered:

```
root@solarishost:~# yes | pkgrm SUNWsamfsu SUNWsamfsr
```

3. If you have not already done so, locate the Oracle HSM packages for your host architecture.
4. To install both the Oracle Hierarchical Storage Manager and StorageTek QFS Software packages, use the command **pkgadd -d . SUNWsamfsr SUNWsamfsu** and accept all defaults.

   Note that you must install the **SUNWsamfsr** package before installing the **SUNWsamfsu** package. In the example, we make sure that we are in the correct directory for our operating system, **Oracle_HSM_6.1.4/solaris_sparc/S10**. Then we pipe the reply **yes** into the command so that all questions are automatically answered:

```
root@solarishost:~# pwd
/net/Oracle_HSM_6.1.4/solaris_sparc/s10
root@solarishost:~# yes | pkgadd -d . SUNWsamfsr SUNWsamfsu
```

5. To install only the QFS Software packages, use the command **pkgadd -d . SUNWqfsr SUNWqfsu** and accept all defaults.

   Note that you must install the **SUNWqfsr** package before installing the **SUNWqfsu** package. In the example, we pipe the reply **yes** into the command so that all questions are automatically answered:

```
root@solarishost:~# pwd
/net/Oracle_HSM_6.1.4/solaris_sparc/s10
root@solarishost:~# yes | pkgadd -d . SUNWqfsr SUNWqfsu
```

6. If the planned Oracle HSM configuration includes Linux hosts as shared file-system clients, go to "Install or Update Oracle HSM Client Software on a Linux Host".
7. Otherwise, go to Chapter 5, "Using the samsetup Configuration Wizard" or Chapter 6, "Configuring the Basic File System".

Install or Update Oracle HSM Client Software on a Linux Host

For each Linux client of a Oracle HSM shared file system, proceed as follows:

1. Log in to the Linux client as **root**.

```
[root@linux ~]#
```

2. Unmount all mounted Oracle HSM file systems.
3. Uninstall old Oracle HSM packages. Run the script **/var/opt/SUNWsamfs/Uninstall**:

```
[root@linux ~]# /var/opt/SUNWsamfs/Uninstall
```

4. Locate the Linux client ISO image. The ISO image is in the directory where you downloaded the Oracle HSM installation software (see "Download the HSM Installation Package" on page 4-1).

   In the example, we use **ssh** to login to the repository host **sw_install** (IP address **192.168.0.2**). We locate the software in the directory **/hsmqfs**:

```
[root@linux ~]# ssh root@sw_install
Password:
root@sw_install:~# ls -1 /hsmqfs
./       COPYRIGHT.txt       linux.iso            README.txt
../      iso.md5             Oracle-HSM_6.0/
```

5. On the Linux host, create a temporary directory.

   In the example, we create the directory **/hsmtemp**:

```
[root@linux ~]# mkdir /hsmtemp
[root@linux ~]#
```

6. Make the **linux.iso** image available to the Linux host. NFS mount the remote directory that holds the image on the temporary directory that you just created.

Use the command **mount -t nfs** *repository-host-IP***:***hsm-repository-dir* *temp-dir*, where:

- **-t nfs** identifies the type of file system being mounted.
- *repository-host-IP* is the IP address of the server that hosts your installation software.
- *hsm-repository-dir* is the directory that holds the Oracle HSM installation software.
- *temp-dir* is the temporary directory that you created on the Linux host.

In the example, we NFS mount directory **/hsmqfs** of host **sw_install** (**192.168.0.2**) on mount-point directory **/hsmtemp**:

```
[root@linux ~]# mount -t nfs 192.168.0.2:/hsmqfs /hsmtemp
[root@linux ~]#
```

7. Mount the **linux.iso** image on the Linux host. Use the command **mount -o ro,loop -t iso9660** *temp-dir***/linux.iso /mnt**, where:

   - **-o** specifies a list of mount options.
   - **ro** mounts the image read-only.
   - **loop** mounts the image as a loop device.
   - **-t iso9660** identifies the type of file system being mounted.
   - *temp-dir* is the temporary directory where the remote image repository directory is mounted.
   - **/mnt** is the standard, temporary mount point directory on Linux systems.

   In the example, the ISO image is in **/hsmtemp**:

```
[root@linux ~]# mount -o ro,loop -t iso9660 /hsmtemp/linux.iso /mnt
[root@linux ~]#
```

8. Run the installer. Use the command **/mnt/linux1/Install**.

```
[root@linux ~]# /mnt/linux1/Install
```

9. If the installation program does not recognize the installed version of the Linux kernel, it will prompt you to create a custom kernel. Enter **yes**.

```
[root@linux ~]# ./Install
...
A direct match for your kernel wasn't found. Attempt creating a custom rpm for
your kernel (yes/no)? yes
```

   Many variations of the Linux kernel exist. The Oracle HSM installation program compiles custom kernel modules so that it can support the largest possible number of variations.

10. Follow the on-screen instructions.

11. If you are installing a SuSE Linux client, configure the system to recognize the man pages. Open the **/etc/manpath.config** file in a text editor, and add **1m** to the value of the **SECTION** parameter.

    In the example, we use the **vi** editor:

```
[root@linux ~]# vi /etc/manpath.config
...
#-------------------------------------------------------
```

```
# Section names. Manual sections will be searched in the order listed here;
# the default is 1, n, l, 8, 3, 2, 5, 4, 9, 6, 7. Multiple SECTION
# directives may be given for clarity, and will be concatenated together in
# the expected way.
# If a particular extension is not in this list (say, 1mh), it will be
# displayed with the rest of the section it belongs to. The effect of this
# is that you only need to explicitly list extensions if you want to force a
# particular order. Sections with extensions should usually be adjacent to
# their main section (e.g. "1 1mh 8 ...").
SECTION 1 1m n l 8 3 2 3posix 3pm 3perl 5 4 9 6 7
```

12. If the planned Oracle HSM configuration includes additional Linux client hosts, repeat this procedure from the beginning until the client software is installed on all hosts.
13. Otherwise, go to "Using the `samsetup` Configuration Wizard" on page 5-1 or "Configuring the Basic File System" on page 6-1.

Uninstalling Oracle HSM Software

This section outlines following procedures:

- Uninstall Oracle HSM on a Solaris Host
- Uninstall the Oracle HSM Client on a Linux Host.

Caution: Do not uninstall software if you intend to upgrade or reinstall Oracle HSM using an existing configuration! Uninstalling removes all configuration files. Instead, use one of the upgrade methods outlined in "Install, Upgrade, or Downgrade Oracle HSM on an Oracle Solaris Host" on page 4-6.

Uninstall Oracle HSM on a Solaris Host

To completely uninstall software and remove the configuration files, proceed as follows.

1. Log in to the host as **`root`**.

```
root@solarishost:~#
```

2. If the software was installed on Solaris 11 or later using the Solaris Image Packaging System, uninstall the software using the command **`pkg uninstall SUNWsamfs SUNWsamqassy`** (or **`pkg uninstall SUNWqfs SUNWsamqassy`** if only the QFS software is installed).

```
root@solarishost:~#  pkg uninstall SUNWsamfs SUNWsamqassy
```

3. If the software was installed on Solaris 10 or on Solaris 11 using the SVR4 **`pkginstall`** method, uninstall the software using the command **`pkgrm SUNWsamfsu SUNWsamfsr`** (**`pkgrm SUNWqfsu SUNWqfsr`** if only the QFS software is installed).

   Remove the packages in the order specified, starting with **`SUNWsamfsu`** and ending with **`SUNWsamfsr`**. In the example, we pipe the reply **`yes`** into the command so that all questions are automatically answered:

```
root@solarishost:~# yes | pkgrm SUNWsamfsu SUNWsamfsr
```

4. If the software was installed on Solaris 10 or on Solaris 11 using the SVR4 **pkginstall** method, delete configuration and log files that are no longer required.

```
root@solarishost:~# rm -R /var/opt/SUNWsamfs/
root@solarishost:~# rm -R /etc/opt/SUNWsamfs/
root@solarishost:~# rm -R /var/adm/sam-log/
root@solarishost:~#
```

5. Reboot the host.

```
root@solarishost:~#  reboot
```

6. Stop here.

Uninstall the Oracle HSM Client on a Linux Host

To uninstall and completely remove the Linux client software, proceed as follows.

1. Log in to the Linux client host as **root**.

```
[root@linux ~]#
```

2. Run the Oracle HSM script **/var/opt/SUNWsamfs/Uninstall** (**/var/opt/SUNWqfs/Uninstall** if only QFS is installed).

   Do not use any other method! Other methods, such as **rpm -e**, can cause unexpected results and problems with uninstalling or reinstalling the software. So always use the script:

```
[root@linux ~]#  /var/opt/SUNWsamfs/Uninstall
```

5
Using the **`samsetup`** Configuration Wizard

The **`samsetup`** wizard is a simple, text-based, menu-driven utility that lets you quickly create and configure Oracle HSM file systems to meet most commonly encountered requirements. The wizard can guide you through all of the following basic tasks:

- creating QFS standalone file systems mounted on a single host
- creating QFS shared file systems mounted on multiple hosts
- configuring Oracle HSM archiving for QFS file systems
- configuring storage hardware, including primary (cache) disk storage, archival disk storage, and removable-media libraries, drives, and media.

The wizard's output—valid Oracle HSM configuration scripts—can also make a useful starting point when you are creating more specialized solutions.

The **`samsetup`** wizard is essentially self-documenting—menus and prompts guide you through the process, and context-sensitive online help is immediately available. So this chapter does not duplicate the information provided by the tool itself.

You should, however, review the remaining sections of this book before using the wizard, particularly if you are new to Oracle HSM:

- "Configuring the Basic File System" on page 6-1, provides essential information about the way Oracle HSM works and explains the configuration files and processes that create file systems. You will need this information to fully understand the options that the wizard offers you, even if you never feel the need to create and edit configuration files yourself.
- If you require an Oracle HSM archiving file system, you will need the information in "Configure File System Protection" on page 6-50. The **`samsetup`** wizard does not configure scheduled backups of critical file-system metadata and logs.
- If you require an Oracle HSM shared file system, review "Accessing File Systems from Multiple Hosts" on page 8-1 as well. The sections "Accessing File Systems from Multiple Hosts Using Oracle HSM Software" on page 8-1 and "Configuring an Oracle HSM Shared File System" on page 8-7 will be particularly relevant.
- If you need to use the additional features of the Oracle HSM, you should consult the relevant sections of this book for additional configuration instructions. See, for example, "Configure Archival Media Validation" on page 6-56, "Enabling Support for Write Once Read Many (WORM) Files" on page 6-65, "Enabling Support for the Linear Tape File System (LTFS)" on page 6-67, "Preparing High-Availability Solutions" on page 9-1, "Configuring SAM-Remote" on page 7-1, and "Configuring the Reporting Database" on page 10-1.

- Finally, whether you configure file systems using **`samsetup`** or using the command line or Oracle HSM Manager user interface, you should protect your work as described in "Backing Up the Oracle HSM Configuration" on page 13-1.

6

Configuring the Basic File System

QFS file systems are the basic building blocks of all Oracle HSM solutions. Used on their own, they offer high performance, effectively unlimited capacity, and support for extremely large files. When used with Oracle Hierarchical Storage Manager and suitably configured archival storage, they become Oracle HSM archiving file systems. Both archiving and non-archiving QFS file systems can then form the basis of more complex, multiple-host and high-availability configurations. So this chapter outlines the basic tasks involved when creating and configuring them.

Configuring QFS File Systems

Creating and configuring a basic QFS file system is straightforward. You save the file system information in a Master Configuration File (**`mcf`**) and create the corresponding file system using the **`/opt/SUNWsamfs/sbin/sammkfs`** command. Then create a mount point, add the file system's mount parameters to the host's virtual file system configuration, and mount the new file system. The process can be performed using either the graphical Oracle HSM Manager interface or a text editor and commandline terminal. But in the examples, we use the editor-and-commandline method to make the underlying process explicit and thus easier to understand.

When creating a QFS file system, proceed as follows:

- Select the type of QFS file system that best meets your needs.

  In most cases, a general-purpose file system that stores data and metadata on the same devices is the best choice. When file systems are large and seek time must be minimized, a high-performance file system that stores metadata and data on separate, dedicated devices may be preferable.

- Configure the required file system.

  See "Configure a General-Purpose `ms` File System" on page 6-1 or "Configure a High-Performance `ma` File System" on page 6-5.

- Mount the file system.

  See "Create a Mount Point for the New QFS File System" on page 6-7 and "Add the New QFS File System to the Solaris Virtual File System Table" on page 6-8.

Configure a General-Purpose `ms` File System

1. Log in to the file-system host as **`root`**. Log in to the global zone if the host is configured with zones.

```
root@qms1mds:~#
```

2. Create the file **/etc/opt/SUNWsamfs/mcf**.

   The **mcf** (*master configuration file*) is a table of six columns separated by white space, each representing one of the parameters that define a QFS file system: **Equipment Identifier**, **Equipment Ordinal**, **Equipment Type**. **Family Set**, **Device State**, and **Additional Parameters**. The rows in the table represent file-system equipment, which includes both storage devices and groups of devices (*family sets*).

   You can create the **mcf** file by selecting options in the Oracle HSM Manager graphical user interface or by using a text editor. In the example below, we use the **vi** text editor:

```
root@qms1mds:~# vi /etc/opt/SUNWsamfs/mcf
~
~
"/etc/opt/SUNWsamfs/mcf"  [New File]
```

3. For the sake of clarity, enter column headings as comments.

   Comment rows start with a hash sign (**#**):

```
# Equipment          Equipment  Equipment  Family     Device   Additional
# Identifier         Ordinal    Type       Set        State    Parameters
#-----------------   ---------  ---------  ---------  ------   ----------------
```

4. In the **Equipment Identifier** field (the first column) of the first row, enter the name of the new file system.

   The equipment identifier can contain up to 31 characters, must start with an alphabetic character, and can contain only letters, numbers, and/or underscore (_) characters. In this example, the file system is named **qms1**:

```
# Equipment          Equipment  Equipment  Family     Device   Additional
# Identifier         Ordinal    Type       Set        State    Parameters
#-----------------   ---------  ---------  ---------  ------   ----------------
qms1
```

5. In the **Equipment Ordinal** field (the second column), enter a number that will uniquely identify the file system.

   The equipment ordinal number uniquely identifies all equipment controlled by Oracle HSM. In this example, we use **100** for the **qms1** file system:

```
# Equipment          Equipment  Equipment  Family     Device   Additional
# Identifier         Ordinal    Type       Set        State    Parameters
#-----------------   ---------  ---------  ---------  ------   ----------------
qms1                 100
```

6. In the **Equipment Type** field (the third column), enter the equipment type for a general-purpose QFS file system, **ms**:

```
# Equipment          Equipment  Equipment  Family     Device   Additional
# Identifier         Ordinal    Type       Set        State    Parameters
#-----------------   ---------  ---------  ---------  ------   ----------------
qms1                 100        ms
```

7. In the **Family Set** field (the fourth column), enter the name of the file system.

   The **Family Set** parameter defines a group of equipment that are configured together to form a unit, such as a robotic tape library and its resident tape drives or a file system and its component disk devices.

The family set name must have the same value as the equipment identifier. So, in the example, we name the family set **qms1**:

```
# Equipment          Equipment  Equipment  Family     Device   Additional
# Identifier         Ordinal    Type       Set        State    Parameters
#-----------------   ---------  ---------  ---------  ------   ----------------
qms1                 100        ms         qms1
```

8. Enter **on** in the **Device State** column, and leave the **Additional Parameters** column blank.

   This row is complete:

```
# Equipment          Equipment  Equipment  Family     Device   Additional
# Identifier         Ordinal    Type       Set        State    Parameters
#-----------------   ---------  ---------  ---------  ------   ----------------
qms1                 100        ms         qms1       on
```

9. Start a new row. Enter the identifier for one of the disk devices that you selected in the **Equipment Identifier** field (the first column), and enter a unique number in the **Equipment Ordinal** field (the second column).

   In the example, we indent the device line to emphasize the fact that the device is part of the **qms1** file-system family set and increment the equipment number of the family set to create the device number, in this case **101**:

```
# Equipment          Equipment  Equipment  Family     Device   Additional
# Identifier         Ordinal    Type       Set        State    Parameters
#-----------------   ---------  ---------  ---------  ------   ----------------
qms1                 100        ms         qms1       on
/dev/dsk/c1t3d0s3    101
```

10. In the **Equipment Type** field of the disk device row (the third column), enter the equipment type for a disk device, **md**.

    For more information on device identifiers, see the "Glossary of Equipment Types" on page A-1.

```
# Equipment          Equipment  Equipment  Family     Device   Additional
# Identifier         Ordinal    Type       Set        State    Parameters
#-----------------   ---------  ---------  ---------  ------   ----------------
qms1                 100        ms         qms1       on
/dev/dsk/c1t3d0s3    101        md
```

11. Enter the family set name in the **Family Set** field of the disk device row (the fourth column), enter **on** in the **Device State** field (the fifth column), and leave the **Additional Parameters** field (the sixth column) blank.

    The family set name **qms1** identifies the disk equipment as part of the hardware for the **qms1** file system.

```
# Equipment          Equipment  Equipment  Family     Device   Additional
# Identifier         Ordinal    Type       Set        State    Parameters
#-----------------   ---------  ---------  ---------  ------   ----------------
qms1                 100        ms         qms1       on
/dev/dsk/c1t3d0s3    101        md         qms1       on
```

12. Now add entries for any remaining disk devices, save the file, and quit the editor.

```
# Equipment          Equipment  Equipment  Family     Device   Additional
# Identifier         Ordinal    Type       Set        State    Parameters
#-----------------   ---------  ---------  ---------  ------   ----------------
qms1                 100        ms         qms1       on
```

```
/dev/dsk/c1t3d0s3    101        md         qms1       on
/dev/dsk/c1t4d0s5    102        md         qms1       on
:wq
root@qms1mds:~#
```

13. Check the **mcf** file for errors by running the **sam-fsd** command.

    The **sam-fsd** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error:

    ```
    root@qms1mds:~# sam-fsd
    ```

14. 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@qms1mds:~# sam-fsd
    Problem in mcf file /etc/opt/SUNWsamfs/mcf for filesystem qms1
    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 slice number part of the equipment name for device **102**, the second **md** device:

    ```
    qms1                 100        ms         qms1       on
    /dev/dsk/c0t0d0s0    101        md         qms1       on
    /dev/dsk/c0t3d0so    102        md         qms1       on
    ```

15. 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@qms1mds:~# sam-fsd
    Trace file controls:
    sam-amld      /var/opt/SUNWsamfs/trace/sam-amld
                  cust err fatal ipc misc proc date
                  size    10M  age 0
    sam-archiverd /var/opt/SUNWsamfs/trace/sam-archiverd
                  cust err fatal ipc misc proc date module
                  size    10M  age 0
    sam-catserverd /var/opt/SUNWsamfs/trace/sam-catserverd
                  cust err fatal ipc misc proc date module
                  size    10M  age 0
    ...
    Would start sam-archiverd()
    Would start sam-stagealld()
    Would start sam-stagerd()
    Would start sam-amld()
    ```

16. Tell the Oracle HSM software to reread the **mcf** file and reconfigure itself accordingly. Use the command **samd config**.

    ```
    root@qms1mds:~# samd config
    Configuring SAM-FS
    root@qms1mds:~#
    ```

17. If the command **samd config** fails with the message **You need to run /opt/SUNWsamfs/util/SAM-QFS-post-install**, you forgot to run the post-installation script when you installed the software. Run it now.

    ```
    root@qms1mds:~# /opt/SUNWsamfs/util/SAM-QFS-post-install
    ```

```
- The administrator commands will be executable by root only (group bin).
If this is the desired value, enter "y".  If you want to change
the specified value enter "c".
...
root@qms1mds:~#
```

18. Create the file system using the **/opt/SUNWsamfs/sbin/sammkfs** command and the family set name of the file system.

    The Oracle HSM software uses dual-allocation and default Disk Allocation Unit (DAU) sizes for **md** devices. This is a good choice for a general-purpose file system, because it can accommodate both large and small files and I/O requests. In the example, we accept the defaults:

    ```
    root@qms1mds:~# sammkfs qms1
    Building 'qms1' will destroy the contents of devices:
      /dev/dsk/c1t3d0s3
      /dev/dsk/c1t4d0s5
    Do you wish to continue? [y/N]yes
    total data kilobytes       = ...
    ```

    If we needed to specify a non-default DAU size that better met our I/O requirements, we could use the **-a** option to specify the number of 1024-byte blocks in the desired DAU (for additional information, see the **sammkfs** (1m) man page):

    ```
    root@qms1mds:~# sammkfs -a 16 qms1
    ```

19. If you are using Oracle Hierarchical Storage Manager to set up an archiving file system, go to "Configuring Oracle HSM Archiving File Systems" on page 6-11 now.

20. Otherwise, create a mount point for the new file system.

Configure a High-Performance ma File System

Once the Oracle HSM software is installed on the file-system host, you configure an **ma** file system as described below.

1. Log in to the file system host as **root**. Log in to the global zone if the host is configured with zones.

   ```
   root@qma1mds:~#
   ```

2. Select the disk devices that will hold the metadata.

3. Select the disk devices that will hold the data.

4. Create the **mcf** file.

   You can create the **mcf** file by selecting options in the Oracle HSM Manager graphical user interface or by using a text editor. In the example below, we use the **vi** text editor:

   ```
   root@qma1mds:~# vi /etc/opt/SUNWsamfs/mcf
   ~
   "/etc/opt/SUNWsamfs/mcf"  [New File]
   ```

5. For the sake of clarity, enter column headings as comments.

   Comment rows start with a hash sign (**#**):

   ```
   # Equipment           Equipment  Equipment  Family  Device   Additional
   # Identifier          Ordinal    Type       Set     State    Parameters
   ```

```
#------------------   ---------  ---------  ------  ------   -----------------
```

6. Create an entry for the file-system family set.

   In this example, we identify the file system as **qma1**, increment the equipment ordinal to **200**, set the equipment type to **ma**, set the family set name to **qma1**, and set the device state **on**:

```
# Equipment           Equipment  Equipment  Family  Device   Additional
# Identifier          Ordinal    Type       Set     State    Parameters
#------------------   ---------  ---------  ------  ------   -----------------
qma1                  200        ma         qma1    on
```

7. Add an entry for each metadata device. Enter the identifier for the disk device you selected in the equipment identifier column, set the equipment ordinal, and set the equipment type to **mm**.

   Add enough metadata devices to hold the metadata required for the size of the file system. In the example, we add a single metadata device:

```
# Equipment           Equipment  Equipment  Family  Device   Additional
# Identifier          Ordinal    Type       Set     State    Parameters
#------------------   ---------  ---------  ------  ------   -----------------
qma1                  200        ma         qma1    on
/dev/dsk/c0t0d0s0     201        mm         qma1    on
```

8. Now add entries for the data devices, save the file, and quit the editor.

   These can be either **md**, **mr**, or striped-group (**g***XXX*) devices. For this example, we will specify **md** devices:

```
# Equipment          Equipment  Equipment  Family  Device   Additional
# Identifier         Ordinal    Type       Set     State    Parameters
#------------------  ---------  ---------  ------  ------   -----------------
qma1                 200        ma         qma1    on
/dev/dsk/c0t0d0s0    201        mm         qma1    on
/dev/dsk/c0t3d0s0    202        md         qma1    on
/dev/dsk/c0t3d0s1    203        md         qma1    on
:wq
root@qma1mds:~#
```

9. Check the **mcf** file for errors by running the **sam-fsd** command.

   The **sam-fsd** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error:

```
root@qma1mds:~# sam-fsd
```

10. 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@qma1mds:~# sam-fsd
Problem in mcf file /etc/opt/SUNWsamfs/mcf for filesystem qma1
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 an exclamation point (**!**) instead of a 1 in the slice number part of the equipment name equipment name for device **202**, the first **md** device:

```
sharefs1             200        ma         qma1    on
```

```
/dev/dsk/c0t0d0s0    201        mm         qma1    on
/dev/dsk/c0t0d0s!    202        md         qma1    on
/dev/dsk/c0t3d0s0    203        md         qma1    on
```

11. 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@qma1mds:~# sam-fsd
    Trace file controls:
    sam-amld      /var/opt/SUNWsamfs/trace/sam-amld
                  cust err fatal ipc misc proc date
                  size    10M  age 0
    sam-archiverd /var/opt/SUNWsamfs/trace/sam-archiverd
                  cust err fatal ipc misc proc date module
                  size    10M  age 0
    sam-catserverd /var/opt/SUNWsamfs/trace/sam-catserverd
                  cust err fatal ipc misc proc date module
                  size    10M  age 0
    ...
    Would start sam-archiverd()
    Would start sam-stagealld()
    Would start sam-stagerd()
    Would start sam-amld()
    ```

12. Create the file system using the **/opt/SUNWsamfs/sbin/sammkfs** command and the family set name of the file system.

    In the example, we create the file system using the default Disk Allocation Unit (DAU) size for **ma** file systems with **md** devices, **64** kilobytes:

    ```
    root@qma1mds:~# sammkfs qma1
    Building 'qma1' will destroy the contents of devices:
      /dev/dsk/c0t0d0s0
      /dev/dsk/c0t3d0s0
      /dev/dsk/c0t3d0s1
    Do you wish to continue? [y/N]yes
    total data kilobytes       = ...
    ```

    The default is a good, general-purpose choice. But if the file system were to primarily support smaller files or applications that read and write smaller amounts of data, we could also specify a DAU size of **16** or **32** kilobytes. To specify a 16-kilobytes DAU, we would use the **sammkfs** command with **-a** option:

    ```
    root@qma1mds:~# sammkfs -a 16 qma1
    ```

    The DAU for **mr** devices and **gXXX** striped groups is fully adjustable within the range **8-65528** kilobytes, in increments of 8 kilobytes. The default is **64** kilobytes for **mr** devices and **256** kilobytes for **gXXX** striped groups. See the **sammkfs** man page for additional details.

13. If you are using Oracle Hierarchical Storage Manager to set up an archiving file system, go to "Configuring Oracle HSM Archiving File Systems" on page 6-11 now.

14. Otherwise, create a mount point for the new file system.

Create a Mount Point for the New QFS File System

1. If you are not currently logged in, log in to the file system host as **root**. Log in to the global zone if the host is configured with zones.

```
root@qmx1mds:~#
```

2. Create a mount-point directory for the new file system.

```
root@qmx1mds:~# mkdir /hsm/qmx1
root@qmx1mds:~#
```

3. 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 **/qm*x*1** mount-point directory and set permissions to **755** (**-rwxr-xr-x**):

```
root@qmx1mds:~# mkdir /hsm/qmx1
root@qmx1mds:~# chmod 755 /hsm/qmx1
root@mds1:~#
```

4. Add the file system to the **/etc/vfstab** file.

Add the New QFS File System to the Solaris Virtual File System Table

1. If you are not currently logged in, log in to the file system host as **root**. Log in to the global zone if the host is configured with zones.

```
root@qmx1mds:~#
```

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

```
root@qmx1mds:~# cp /etc/vfstab /etc/vfstab.backup
```

3. . Open the **/etc/vfstab** file in a text editor, start a new line, and enter the name of the file system in the first column, under the heading **File Device to Mount**.

```
root@qmx1mds:~# 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       -
...
/qmx1
```

4. In the second column, under the heading **Device to fsck**, enter a hyphen (**-**).

```
root@qmx1mds:~# 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       -
...
/qmx1      -
```

5. In the third column, under the heading **Mount Point**, enter the path to the mount point that you created for the file system.

```
root@qmx1mds:~# 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       -
```

```
...
/qmx1      -       /hsm/qmx1
```

6. In the fourth column, under the heading **System Type**, enter the file system type, **samfs**.

```
root@qmx1mds:~# 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       -
...
/qmx1      -        /hsm/qmx1 samfs
```

7. In the fifth column, under the heading **fsck Pass**, enter a hyphen (**-**).

```
root@qmx1mds:~# 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       -
...
/qmx1      -        /hsm/qmx1 samfs   -
```

8. In the sixth column, under the heading **Mount at Boot**, enter **no**.

```
root@qmx1mds:~# 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       -
...
qmx1       -        /hsm/qmx1 samfs   -     no
```

9. To specify round-robin allocation, add the **stripe=0** mount option.

   Setting mount options in **/etc/vfstab** is usually simplest during initial file system configuration. But you can also set most options in an optional **/etc/opt/SUNWsamfs/samfs.cmd** file or from the command line. See the **samfs.cmd** (4) and **mount_samfs** (1m) man pages for details.

```
#File
#Device    Device   Mount     System  fsck  Mount    Mount
#to Mount  to fsck  Point     Type    Pass  at Boot  Options
#--------  -------  --------  ------  ----  -------  -------------------------
/devices   -        /devices  devfs   -     no       -
...
qmx1       -        /hsm/qmx1 samfs   -     no       stripe=0
```

10. To specify striped allocation, add the **stripe=***stripe-width* mount option, where *stripe-width* is an integer in the range **[1-255]** that represents the number of Disk Allocation Units (DAUs) that should be written to each disk in the stripe.

    When striped allocation is specified, data is written to devices in parallel. So, for best performance, choose a stripe width that fully utilizes the bandwidth available with your storage hardware. Note that the volume of data transferred for a given stripe width depends on how hardware is configured. For **md** devices implemented on single disk volumes, a stripe width of **1** writes one 64-kilobyte DAU to each of two disks for a total of 128 kilobytes. For **md** devices implemented on 3+1 RAID 5 volume groups, the same stripe width transfers one 64-kilobyte DAU to each of the three data disks on each of two devices, for a total of six DAUs or 384 kilobytes per transfer. In our example, we set the stripe width to one DAU:

```
#File
#Device    Device   Mount     System  fsck  Mount    Mount
#to Mount  to fsck  Point     Type    Pass  at Boot  Options
#--------  -------  --------  ------  ----  -------  -------------------------
/devices   -        /devices  devfs   -     no       -
...
qmx1       -        /hsm/qmx1 samfs   -     no       stripe=1
```

11. You can try adjusting the stripe width to make better use of the available hardware. In the **Mount Options** field for the file system, set the **stripe=***n* mount option, where *n* is a multiple of the DAU size specified for the file system. Test the I/O performance of the file system and readjust the setting as needed.

    When you set **stripe=0**, Oracle HSM writes files to devices using round-robin allocation. Each file is completely allocated on one device until that device is full. Round-robin is preferred for shared file systems and multistream environments.

    In the example, we have determined that the bandwidth of our RAID-5 volume groups are under-utilized with a stripe width of one, so we try **stripe=2**:

```
#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       -
...
qmx1       -        /hsm/qmx1 samfs   -     no       ...,stripe=2
```

12. Otherwise, save the **vfstab** file.

```
...
qmx1       -        /hsm/qmx1 samfs   -     no       stripe=1
:wq
root@qmx1mds:~#
```

13. Mount the new file system using the Solaris **mount** command.

```
root@qmx1mds:~# mount /qfxms
```

    The basic file system is now complete and ready to use.

14. If you are using Oracle Hierarchical Storage Manager to set up an archiving file system, see "Configuring Oracle HSM Archiving File Systems" on page 6-11.
15. If you need to enable WORM (Write Once Read Many) capability on the file system, see "Enabling Support for Write Once Read Many (WORM) Files" on page 6-65.
16. If you need to interwork with systems that use LTFS or if you need to transfer large quantities of data between remote sites, see "Enabling Support for the Linear Tape File System (LTFS)" on page 6-67.
17. If you have additional requirements, such as multiple-host file-system access or high-availability configurations, see "Beyond the Basics" on page 6-70.
18. Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

Configuring Oracle HSM Archiving File Systems

Archiving file systems combine one or more QFS **ma**- or **ms**-type file systems with archival storage and Oracle Hierarchical Storage Manager software. When users and applications create or modify files stored in the primary file-system disk cache, the Oracle HSM software automatically archives a specified number of file copies on specified archival storage media. Archival media can include magnetic or solid-state disk devices, removable magnetic tapes or optical disks, and/or Oracle Storage Cloud containers.

The Oracle HSM software integrates archival storage into basic file-system operations. So little-used, primary cache-resident files can be released to free primary disk space without deleting any data from the file system as a whole. Files can be maintained in multiple copies on varied media for maximum redundancy and continuous data protection.

To configure an archiving file system, carry out the tasks below:

- Configure archival storage devices and media.
- Configure the archiving file system.
- Mount the archiving file system.
- Configure the archiving process.
- If you are using a network-attached tape library, catalog the media.
- Configure file system protection.
- If required, configure archival media validation.
- If required, enable write-once, read-many (WORM) support.

Configure Archival Storage Devices and Media

Oracle HSM archiving file systems can copy files from the primary file-system disk cache to any of three types of archival media: solid-state or magnetic disk, magnetic tape stored in a robotic library, or cloud storage. To configure Oracle HSM any of these types of storage, perform the corresponding task listed below:

- Add disk volumes to Oracle HSM host configurations.
- Add cloud libraries to Oracle HSM host configurations.
- Configure removable media libraries and drives.

Configure Oracle HSM Hosts to Use Disk Volumes

1. If you have not already done so, allocate the devices and configure the file systems for the archival disk volumes. See "Configure Archival Disk Storage" on page 3-3.
2. Create the **/etc/opt/SUNWsamfs/diskvols.conf** file in a text editor, and assign a volume serial number (VSN) to each file system. For each file system, start a new line consisting of the desired volume serial number, white space, and the path to the file-system mount point. Then save the file.

   In the example, we have configured the host to use fifteen NFS file systems as disk-based archival volumes, **DISKVOL1** to **DISKVOL15**. All are mounted on the **/diskvols/** directory:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/diskvols.conf
   # Volume
   # Serial     Resource
   ```

```
# Number     Path
# ------     ---------------------
DISKVOL1     /diskvols/DISKVOL1
DISKVOL2     /diskvols/DISKVOL2
...
DISKVOL15    /diskvols/DISKVOL15
```

3. If you plan use Oracle Storage Cloud services as Oracle HSM logical media, configure Oracle HSM for the services now.
4. Otherwise, configure network-attached robotic libraries, removable media, and drives.

Configure Oracle HSM Hosts to Use Storage Cloud Libraries

Cloud libraries (equipment type cr) and cloud media volumes (media type cl) are the Oracle HSM interface to public and private storage clouds. Storage clouds are abstract, network services that provide an agreed level of service rather than a set of defined physical resources. Oracle HSM cloud libraries and media let you use cloud resources in the same way that you use a removable media library.

For each cloud library that you plan to use with Oracle HSM, carry out the following tasks:

- If you have not done so, carry out the required preliminary configuration steps, as described in "Enable Oracle HSM Cloud Libraries" on page 2-3.
- Create a password file for the storage cloud account.
- Create a parameters file for the storage cloud account.

Create a Storage Cloud Account Password File

When you initially configure cloud storage accounts and, periodically, thereafter, you must supply the software with current authentication passwords. The software stores them securely in encrypted form. For each account that you intend to use with Oracle HSM, proceed as follows:

1. Before proceeding further, make sure that you have the password for the cloud user account that you have provided for the Oracle HSM software.

   If you are using Oracle Storage Cloud, this will be the Storage_Administrator account password that you assigned. See "Create Oracle Storage Cloud Service User Accounts for Oracle HSM" on page 3-8. Otherwise, your OpenStack Swift administrator should have provided this password.

2. Log in to the Oracle HSM metadata server host as the user that manages the cloud storage.

   The cloud storage management user can be **`root`** or a less privileged account configured on the Oracle HSM metadata server host. In the example, the host name is **`mds1`**. We log in as user **`root`**:

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

3. Create a password file to hold the password for the cloud user account. Use the command **`sam-cloudd -p`** *`path`*/*`filename`*, where:

   - *`path`* is the absolute path to the directory where you intend to store the password.
   - *`filename`* is the name of the file that is to hold the password.

The command prompts you for the password that the file will store.

In the example, we create the file **/etc/opt/SUNWsamfs/ocld1auth**:

```
root@mds1:~# sam-cloudd -p /etc/opt/SUNWsamfs/ocld1auth
Password:
```

4. At the prompt, enter the password for the Oracle Storage Cloud **Storage_Administrator** user account, and confirm it when prompted.

   The **sam-cloudd -p** *path***/***filename* command encrypts the password and stores the result in the specified file. In the example, the string **P^ssw0rd** represents the account password:

```
root@mds1:~# sam-cloudd -p /etc/opt/SUNWsamfs/ocld1auth
Password: P^ssw0rd
Password: P^ssw0rd
root@mds1:~#
```

5. If you are warned that the specified file already exists, confirm that the file can be overwritten.
6. Now create a cloud library parameters file.

Create a Cloud Library Parameter File

The cloud library parameters file configures storage cloud resources as a virtual, network-attached, automated library of removable media. For each cloud library that you need to configure, 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. Make sure that you have the user name for Oracle HSM's administrative user account and the domain ID of the domain to which it belongs.

   You made a note of this information when carrying out the instructions in "Provide Storage Cloud Resources" on page 3-5.
3. Choose a family set name for the cloud library.

   The family set name of a cloud library is also the name of the parameters file that defines the library, the value of the name parameter in the parameters file, the prefix for the volume serial numbers (VSNs) assigned to cloud media volumes, and the equipment identifier for the library in the master configuration file (**mcf**).

   In the examples, the family set name simply combines the abbreviation cl (*cloud library*) with the **mcf** equipment number that we plan to use for the library: **cl800**.
4. Create the parameters file for the cloud library. In a text editor, open the file **/etc/opt/SUNWsamfs/***file-name*, where *file-name* is the family set name that you are assigning to the cloud resource.

   In the example, we use the **vi** editor to create the file **/etc/opt/SUNWsamfs/cl800** and add a descriptive comment:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
# Oracle Storage Cloud parameters file for library cl800
```

5. Specify the type of cloud container that this cloud library will use. Enter a line of the form **type =** *container_type*, where *container_type* is one of the following:

- **oracle-archive** specifies Oracle Storage Cloud archive-storage containers
- **oracle-object** specifies Oracle Storage Cloud object-storage containers
- **swift-object** specifies standard OpenStack Swift object-storage containers

See "Create Oracle Storage Cloud Service User Accounts for Oracle HSM" on page 3-8 for advice on selecting Oracle Storage Cloud containers.

In the first example, we specify **oracle-archive** containers:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
# Oracle Storage Cloud parameters file for library cl800
type = oracle-archive
```

In the second example, we specify swift-object containers:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/cl810
# Oracle Storage Cloud parameters file for library cl810
type = swift-object
```

6. Start a new line for the storage cloud Uniform Resource Locator (URL). Enter **url=https://cloud-service-URL**, where **cloud-service-URL** is the URL assigned by your cloud service (Oracle Storage Cloud or an OpenStack private cloud).

   If you are using Oracle Storage Cloud, **cloud-service-URL** takes the form **service_name-identity_domain_id.storage.oraclecloud.com**, where:

   - *service_name* is the name of the service that provides the storage for this cloud library.
   - *identity_domain_id* is the Oracle-assigned identifier for the identity domain, the authorization and authentication domain for your part of the multiple-tenancy cloud environment.

   In the first example, we are using Oracle Storage Cloud. We consult our notes in the **hsm-ocloud-info.txt** file for this information. The service name is **example1234** and the identity domain ID is **usexamplecom49808**:

```
root@mds1:~# cat hsm-cloud-info.txt
service name:       example1234
identity domain id: usexamplecom49808
user name:          hsmlibrary800
```

   In the **/etc/opt/SUNWsamfs/cl800** file, we enter the URL shown:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
# Oracle Storage Cloud parameters file for library cl800
type = oracle-archive
url = https://example1234-usexamplecom49808.storage.oraclecloud.com
```

   In the second example, we are using an OpenStack private cloud. We enter the supplied by the Swift administrator, **https://ohsmcl810.cloud.example.com**:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/cl810
# Oracle Storage Cloud parameters file for library cl810
type = swift-object
url = https://ohsmcl810.cloud.example.com
```

7. Add a new line of the form **domain_id = domain_id**, where **domain_id** is the administrative domain that contains the Oracle HSM user and resources.

   If you are using Oracle Storage Cloud, **domain_id** takes the form **service_name-identity_domain_id**, where:

- **service_name** is the name of the service that provides the storage for this cloud library
- **identity_domain_id** is the Oracle-assigned identifier for the authorization and authentication domain for your part of the multiple-tenancy cloud environment

In the first example, we are using Oracle Storage Cloud. So the **domain_id** is **example1234-usexamplecom49808**:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
# Oracle Storage Cloud parameters file for library cl800
type = oracle-archive
url = https://example1234-usexamplecom49808.storage.oraclecloud.com
domain_id = example1234-usexamplecom49808
```

In the second example, we are using an OpenStack private cloud. So the **domain_id** is the string supplied by the administrator, **ohsm**:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/cl810
# Oracle Storage Cloud parameters file for library cl810
type = swift-object
url = https://ohsmcl810.cloud.example.comm
domain_id = ohsm
```

8. Supply the user name for the Oracle Storage Cloud **Storage_Administrator** account that you created for the use of the Oracle HSM software. Enter a line of the form **username =** *name*, where *name* is the user name.

   In the first example, we are using Oracle Storage Cloud. So the username is the name that we configured under the **Storage_Administrator** role, **hsmlibrary800** (see "Create Oracle Storage Cloud Service User Accounts for Oracle HSM" on page 3-8):

```
root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
# Oracle Storage Cloud parameters file for library cl800
type = oracle-archive
url = https://example1234-usexamplecom49808.storage.oraclecloud.com
domain_id = example1234-usexamplecom49808
username = hsmlibrary800
```

   In the second example, we are using an OpenStack private cloud. So the **domain_id** is the value supplied by the administrator, **ohsm**:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/cl810
# Oracle Storage Cloud parameters file for library cl810
type = swift-object
url = https://ohsmcl810.cloud.example.comm
username = ohsmcl810
```

9. On a new line, enter the path to the password file that will authenticate Oracle HSM as the authorized user of the cloud storage account. Enter a line of the form **password_file =** *pathfile*, where:

   - *path* is the absolute path to the password file that you created with the **sam-cloudd -p** command.
   - *file* is the name of the file that contains the encrypted account password.

   In the example, the password file is **/etc/opt/SUNsamfs/ocld1auth**:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
```

```
# Oracle Storage Cloud parameters file for library cl800
type = oracle-archive
url = https://example1234-usexamplecom49808.storage.oraclecloud.com
domain_id = example1234-usexamplecom49808
username = hsmlibrary800
password_file = /etc/opt/SUNsamfs/ocld1auth
```

10. Specify a volume serial number (VSN) prefix that will uniquely identify virtual volumes created by this cloud service configuration. Enter a line of the form **`name =`** *`string`*, where *`string`* is the prefix that you wish to use.

    Prefixes consist of 4 to 20 letters and numerals. In the example, we use the family set name of the library, **`cl800`**, as the prefix::

```
root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
# Oracle Storage Cloud parameters file for library cl800
type = oracle-archive
url = https://example1234-usexamplecom49808.storage.oraclecloud.com
domain_id = example1234-usexamplecom49808
username = hsmlibrary800
password_file = /etc/opt/SUNsamfs/ocld1auth
name = cl800
```

11. Specify the number of logical drives that the cloud service will make available to Oracle HSM. Enter a line of the form **`drives =`** *`number`*, where *`number`* is an integer in the range [**`1-4`**].

    The **`drives`** parameter sets the number of concurrent archiver/stager requests that the cloud service supports. The default is **`4`**. In the example, we do not need the full, default bandwidth, so we specify **`2`** to reduce memory usage:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
# Oracle Storage Cloud parameters file for library cl800
type = oracle-archive
url = https://example1234-usexamplecom49808.storage.oraclecloud.com
domain_id = example1234-usexamplecom49808
username = hsmlibrary800
name = cl800
drives = 2
```

12. Save the file, and close the editor.

```
root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
# Oracle Storage Cloud parameters file for library cl800
type = oracle-archive
url = https://example1234-usexamplecom49808.storage.oraclecloud.com
domain_id = example1234-usexamplecom49808
username = hsmlibrary800
name = cl800
drives = 2
:wq
root@mds1:~#
```

13. If you need to encrypt data before sending it to the cloud, configure cloud encryption now.

14. Otherwise, configure network-attached robotic libraries, removable media, and drives.

Configuring Cloud Storage Encryption

If you need to encrypt data before sending it to the cloud, carry out the following tasks:

- Generate a cloud encryption password file to hold the password to the encryption keystore.
- Add encryption parameters to the cloud library parameter file.

Create a Cloud Encryption Password File

1. If you use Oracle Key Manager (OKM) or Oracle Key Vault (OKV) to manage and secure your keystore, make sure that you have the password for accessing the keystore.
2. Log in to the file system metadata server host as the user that manages the cloud storage.

   The cloud storage management user can be root or a less privileged account configured on the Oracle HSM metadata server host. In the example, the host name is **`mds1`**. We log in as user **`root`**:

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

3. Enter the name and path of a 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 intend to store the password
   - **`filename`** is the name of the file that is to hold the password

   The command prompts you for the account password.

   In the example, we create the file **`/etc/opt/SUNWsamfs/keystore_auth`**:

   ```
   root@mds1:~# sam-cloudd -p /etc/opt/SUNWsamfs/keystore_auth
   Password:
   ```

4. At the prompt, enter the password for the Oracle Storage Cloud **`Storage_Administrator`** user account. If 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.

   ```
   root@mds1:~# sam-cloudd -p /etc/opt/SUNWsamfs/keystore_auth
   Password: ********
   root@mds1:~#
   ```

5. Now add encryption parameters to the cloud library parameters file.

Add Encryption Parameters to the Cloud Library Parameter File

In the cloud library parameter file, you configure the Oracle HSM cloud encryption feature by defining the encryption keystore where cryptographic keys and certificates are to be kept and the type of key label that identifies keys within the keystore. The keystore may be either a key management application, such as Oracle Key Manager (OKM) or Oracle Key Vault (OKV), or it can be an encrypted keystore files (KSF), as described in the **`keystore-file`** (7) man page. Key labels let Oracle HSM request encryption or decryption of a specified cloud media volume without directly—and perhaps insecurely—accessing the actual cryptographic keys.

To add the required keystore parameters to the cloud library parameters file, proceed as follows:

1. If you have not already done so, log in to the file system metadata server host as **root**.

   In the example, the host name is **mds1**:

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

2. In a text editor, open the file **vi /etc/opt/SUNWsamfs/cloud_library**, where **cloud_library** is the family set name that you assigned to the cloud resource when you created the cloud library parameters file.

   In the example, we have given the cloud library the family set name **cl800**:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
   # Oracle Storage Cloud parameters file for library cl800
   type = oracle-archive
   url = https://example1234-usexamplecom49808.storage.oraclecloud.com
   domain_id = example1234-usexamplecom49808
   username = hsmlibrary800
   name = cl800
   drives = 2
   ```

3. In the cloud library parameters file, identify the type of keystore that you intend to use. Enter a line of the form **keystore_type = type**, where **type** is one of the following:

   - **pkcs11** specifies a Public Key Cryptography Standards #11 (Cryptoki) keystore.
   - **file** specifies an encrypted key store file that holds a label, value, and hash value for each key stored.

   In the example, we use **pkcs11**.

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
   # Oracle Storage Cloud parameters file for library cl800
   type = oracle-archive
   url = https://example1234-usexamplecom49808.storage.oraclecloud.com
   domain_id = example1234-usexamplecom49808
   username = hsmlibrary800
   name = cl800
   drives = 2
   keystore_type = pkcs11
   ```

4. Next, name the keystore implementation that you intend to use. Enter a line of the form **keystore_name = name**, where **name** is the PKCS #11 token name of a supported encryption provider. It can have any one of the following values:

   - **KMS** specifies a keystore managed by Oracle Key Manager (OKM) using the Solaris PKCS #11 Key Management Service and a private protocol.
   - **OKV** specifies a keystore managed by Oracle KeyVault using the Key Management Interoperability Protocol (KMIP).
   - **path/file** identifies a keystore file.

   In the example, we use KMS.

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
   ...
   name = cl800
   ```

```
drives = 2
keystore_type = pkcs11
keystore_name = KMS
```

5. Next, identify the file that stores the password for the encrypted keystore file. Enter a line of the form **`keystore_password_file = path/file`**, where **`path/file`** is the path and file name for the file that you generated to hold the keystore password.

   In the example, the file is **`/etc/opt/SUNWsamfs/keystore_auth`**:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
...
name = cl800
drives = 2
keystore_type = pkcs11
keystore_name = KMS
keystore_password_file = /etc/opt/SUNWsamfs/keystore_auth
```

6. Next, define the way in which the library uses encryption keys and generates key labels. Enter a line of the form **`keylabel_type = dynamic|static`**, where **`dynamic`** or **`static`** specifies the required keying and key-labeling behavior:

   - When **`dynamic`** keying and key-labeling is specified, the cloud library automatically generates a unique encryption key and key label when it first encrypts each volume. It bases the key label on the Volume Serial Number (VSN) that identifies the newly encrypted volume.
   - When **`static`** keying and key-labeling is specified, the cloud library uses a single encryption key and key label to encrypt all volumes. The key label must specified by the optional keylabel_name parameter of the cloud library parameter file.

   In the example, we specify dynamic keying and key-labeling for cloud library **`cl800`**:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
...
name = cl800
drives = 2
keystore_type = pkcs11
keystore_name = KMS
keystore_password_file = /etc/opt/SUNWsamfs/keystore_auth
keylabel_type = dynamic
```

7. If you specified static keying and key-labeling behavior (**`keylabel_type = static`**), supply the text that the cloud library should use for the key label. Enter a line of the form **`keylabel_name = label_text`**, where **`label_text`** is the desired text.

   In the example, we have specified static keying and key-labeling for cloud library **`cl801`**, so we set **`keylabel_name`** to our chosen **`label_text`**, **`HSMcl801Key`**:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/cl801
...
name = cl801
drives = 2
keystore_type = pkcs11
keystore_name = KMS
keystore_password_file = /etc/opt/SUNWsamfs/keystore_auth
keylabel_type = static
keylabel_name = HSMcl801Key
```

8. Save the cloud library parameters file, and close the editor.

```
root@mds1:~# vi /etc/opt/SUNWsamfs/cl800
...
name = cl800
drives = 2
keystore_type = pkcs11
keystore_name = KMS
keystore_password_file = /etc/opt/SUNWsamfs/keystore_auth
keylabel_type = dynamic
:wq
root@mds1:~#
```

9. Next, configure network-attached robotic libraries, removable media, and drives.

Configure Network-Attached Robotic Libraries, Removable Media, and Drives

Carry out the tasks listed below:

- If your archiving solution includes an Oracle StorageTek ACSLS network-attached robotic library, configure it now.

  If you have not yet configured a SCSI- or SAN-attached library, go to "Configure Direct-Attached Libraries" on page 3-14.
- Configure labeling behavior for barcoded removable media.
- Check and, if necessary, adjust default drive timing values.

Configure an Oracle StorageTek ACSLS Network-Attached Automated Library

You can configure an Oracle StorageTek ACSLS network-attached library as follows or you can use the Oracle HSM Manager graphical user interface to automatically discover and configure the library (for instructions, see the Oracle HSM Manager online help).

Proceed as follows:

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

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

2. Change to the **`/etc/opt/SUNWsamfs`** directory.

   ```
   root@mds1:~# cd /etc/opt/SUNWsamfs
   ```

3. In a text editor, start a new file with a name that corresponds to the type of network-attached library that you are configuring.

   In the example, we start a parameters file for an Oracle StorageTek ACSLS network-attached library:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/acsls1params
   # Configuration File for an ACSLS Network-Attached Tape Library 1
   ```

4. Enter the parameters and values that the Oracle HSM software will use when communicating with the ACSLS-attached library.

   The Oracle HSM software uses the following Oracle StorageTek Automated Cartridge System Application Programming Interface (ACSAPI) parameters to control ACSLS-managed libraries (for more information, see the **`stk`** man page):

- **access=***user-id* specifies an optional user identification value for access control. By default, there is no user identification-based access control.
- **hostname=***hostname* specifies the hostname of the server that runs the StorageTek ACSLS interface.
- **portnum=***portname* specifies the port number that is used for communication between ACSLS and Oracle HSM software.
- **ssihost=***hostname* specifies the hostname that identifies a multihomed Oracle HSM server to the network that connects to the ACSLS host. The default is the name of the local host.
- **ssi_inet_port=***ssi-inet-port* specifies the fixed firewall port that the ACSLS Server System Interface must use for incoming ACSLS responses. Specify either **0** or a value in the range [**1024-65535**]. The default, **0**, allows dynamic port allocation.
- **csi_hostport=***csi-port* specifies the Client System Interface port number on the ACSLS server to which the Oracle HSM sends its ACSLS requests. Specify either **0** or a value in the range [**1024-65535**]. The default, **0**, causes the system to query the port mapper on the ACSLS server for a port.
- **capid=(acs=***acsnum***, lsm=***lsmnum***, cap=***capnum***)** specifies the ACSLS address of a cartridge access port (CAP), where *acsnum* is the Automated Cartridge System (ACS) number for the library, *lsmnum* is the Library Storage Module (LSM) number for the module that holds the CAP, and *capnum* is the identifying number for the desired CAP. The complete address is enclosed in parentheses.
- **capacity=(***index-value-list***)** specifies the capacities of removable media cartridges, where *index-value-list* is a comma-delimited list of *index***=***value* pairs. Each *index* in the list is the index of an ACSLS-defined media type and each *value* is the corresponding volume capacity in units of 1024 bytes.

  The file **/export/home/ACSSS/data/internal/mixed_media/media_types.dat** defines the media-type indices. In general, you only need to supply a capacity entry for new cartridge types or when you need to override the supported capacity.
- *device-path-name***=(acs=***ACSnumber***, lsm=***LSMnumber***, panel=***Panelnumber***, drive=***Drivenumber***)[shared]** specifies the ACSLS address of a drive that is attached to the client, where *device-path-name* identifies the device on the Oracle HSM server, *acsnum* is the Automated Cartridge System (ACS) number for the library, *lsmnum* is the Library Storage Module (LSM) number for the module that controls the drive, *Panelnumber* is the identifying number for the panel where the drive is installed, and *Drivenumber* is the identifying number of the drive. The complete address is enclosed in parentheses.

  Adding the optional **shared** keyword after the ACSLS address lets two or more Oracle HSM servers share the drive as long as each retains exclusive control over its own media. By default, a cartridge in a shared drive can be idle for 60 seconds before being unloaded.

In the example, we identify **acslserver1** as the ACSLS host, limit access to **sam_user**, specify dynamic port allocation, and map a cartridge access port and two drives:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/acsls1params
# Configuration File for an ACSLS Network-Attached Tape Library 1
hostname = acslserver1
portnum = 50014
```

```
access = sam_user
ssi_inet_port = 0
csi_hostport = 0
capid = (acs=0, lsm=1, cap=0)
/dev/rmt/0cbn = (acs=0, lsm=1, panel=0, drive=1)
/dev/rmt/1cbn = (acs=0, lsm=1, panel=0, drive=2)
```

5. Save the file and close the editor.

```
root@mds1:~# vi /etc/opt/SUNWsamfs/acsls1params
# /etc/opt/SUNWsamfs/acslibrary1
# Configuration File for an ACSLS Network-Attached Tape Library
...
/dev/rmt/0cbn = (acs=0, lsm=1, panel=0, drive=1)
/dev/rmt/1cbn = (acs=0, lsm=1, panel=0, drive=2)
:wq
root@mds1:~#
```

6. If the library or the application software uses non-standard labels for barcoded removable media, configure labeling behavior now.
7. If drives or application software are known to be incompatible with Oracle HSM defaults, set drive timing values now.
8. Otherwise, go to "Configure the Archiving File System" on page 6-25.

Configure Labeling Behavior for Barcoded Removable Media

Oracle HSM identifies tape volumes using six-character, ANSI-standard labels written on the tape media itself. If the library holds a barcode reader and barcoded tape cartridges, Oracle HSM can automatically label media using the first or last six characters of the corresponding barcode. Barcodes of up to 31 characters are supported. If tape media are already labeled, Oracle HSM can use the existing labels.

By default, Oracle HSM automatically labels the media in the library with the first six characters of the cartridge barcode. To configure alternative behavior or restore the default, proceed as follows:

1. Log in to the Oracle HSM host as **root**.

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

2. If you require a non-default behavior or if you have previously overridden the default and need to reset it, open the file **/etc/opt/SUNWsamfs/defaults.conf** in a text editor.

   See the **defaults.conf** (4) man page for additional information on this file.

   In the example, we open the file in the **vi** editor:

```
root@mds1:~# vi /opt/SUNWsamfs/examples/defaults.conf
...
```

3. Locate the line **labels =**, if present, or add it if it is not present.

   In the example, we add the directive:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
# These are the defaults.
...
labels =
```

4. To re-enable the default, automatic labeling based on the first six characters of the barcode, set the value of the **labels** directive to **barcodes**. Save the file, and close the editor.

   The Oracle HSM software now automatically relabels an unlabeled tape media with the first six characters of the cartridge's barcode:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
   ...
   labels = barcodes
   :wq
   root@mds1:~#
   ```

5. To enable automatic labeling based on the last six characters of the barcode, set the value of the **labels** directive to **barcodes_low**. Save the file, and close the editor.

   When the **labels** directive is set to **barcodes_low**, the Oracle HSM software automatically labels media using the last six characters of the cartridge's barcode:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
   ...
   labels = barcodes_low
   :wq
   root@mds1:~#
   ```

6. To configure Oracle HSM to read existing labels from the tape media, set the value of the **labels** directive to **read**. Save the file, and close the editor.

   When the **labels** directive is set to **read**, the Oracle HSM software ignores the barcodes and uses the existing labels. It will not automatically relabel tapes.

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
   ...
   labels = read
   idle_unload = 0
   ...
   :wq
   root@mds1:~#
   ```

7. If drives or application software are known to be incompatible with Oracle HSM defaults, set drive timing values now.
8. Otherwise, go to "Configure the Archiving File System" on page 6-25.

Set Drive Timing Values

By default, the Oracle HSM software sets drive timing parameters as follows:

- The minimum time that must elapse before a specified device type can dismount media is **60** seconds.
- The amount of time that Oracle HSM software waits before issuing new commands to a library that is responding to a SCSI **unload** command is **15** seconds.
- The amount of time that Oracle HSM software waits before unloading an idle drive is **600** seconds (10 minutes).
- The amount of time that Oracle HSM software waits before unloading an idle drive that is shared by two or more Oracle HSM servers is **600** seconds (10 minutes).

To change the default timing values, proceed as follows:

1. If you are not logged in, log in to the Oracle HSM host as **root**.

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

2. Open the **/etc/opt/SUNWsamfs/defaults.conf** file in a text editor.

   In the example, we use the **vi** editor:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
   # These are the defaults.  To change the default behavior, uncomment the
   # appropriate line (remove the '#' character and change the value.
   ...
   ```

3. If required, specify the minimum time that must elapse before a specified device type can dismount media. In the **defaults.conf** file, add a directive of the form *equipment-type*_**delay =** *number-of-seconds*, where *equipment-type* is the two-character, Oracle HSM code that identifies the drive type that you are configuring and *number-of-seconds* is an integer representing the default number of seconds for this device type.

   See Appendix A, "Glossary of Equipment Types" for listings of equipment type codes and corresponding equipment. In the example, we change the unload delay for LTO drives (equipment type **li**) from the default value (**60** seconds) to **90** seconds):

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
   # These are the defaults.  To change the default behavior, uncomment the
   # appropriate line (remove the '#' character and change the value.
   ...
   li_delay = 90
   ```

4. If required, specify the amount of time that Oracle HSM software waits before issuing new commands to a library that is responding to a SCSI **unload** command. In the **defaults.conf** file, add a directive of the form *equipment-type*_**unload =** *number-of-seconds*, where *equipment-type* is the two-character, Oracle HSM code that identifies the drive type that you are configuring and *number-of-seconds* is an integer representing the number of seconds for this device type.

   See Appendix A, "Glossary of Equipment Types" for listings of equipment type codes and corresponding equipment. Set the longest time that the library might need when responding to the **unload** command in the worst-case. In the example, we change the unload delay for LTO drives (equipment type **li**) from the default value (**15** seconds) to **35** seconds:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
   # These are the defaults.  To change the default behavior, uncomment the
   # appropriate line (remove the '#' character and change the value.
   ...
   li_delay = 90
   li_unload = 35
   ```

5. If required, specify the amount of time that Oracle HSM software waits before unloading an idle drive. In the **defaults.conf** file, add a directive of the form **idle_unload =** *number-of-seconds*, where *number-of-seconds* is an integer representing the specified number of seconds.

   Specify **0** to disable this feature. In the example, In the example, we disable this feature by changing the default value (**600** seconds) to **0**:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
   # These are the defaults.  To change the default behavior, uncomment the
   ```

```
# appropriate line (remove the '#' character and change the value.
...
li_delay = 90
li_unload = 35
idle_unload = 0
```

6. If required, specify the amount of time that Oracle HSM software waits before unloading a shared idle drive. In the **defaults.conf** file, add a directive of the form **shared_unload =** *number-of-seconds*, where *number-of-seconds* is an integer representing the specified number of seconds.

   You can configure Oracle HSM servers to share removable-media drives. This directive frees drives for use by other servers when the server that owns the loaded media is not actually using the drive. Specify **0** to disable this feature. In the example, we disable this feature by changing the default value (**600** seconds) to **0**:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
# These are the defaults.  To change the default behavior, uncomment the
# appropriate line (remove the '#' character and change the value.
...
idle_unload = 600
li_delay = 90
li_unload = 35
idle_unload = 0
shared_unload = 0
```

7. Save the file, and close the editor.

```
root@mds1:~# vi /etc/opt/SUNWsamfs/defaults.conf
# These are the defaults.  To change the default behavior, uncomment the
# appropriate line (remove the '#' character and change the value.
...
idle_unload = 600
li_delay = 90
li_unload = 35
idle_unload = 0
shared_unload = 0
:wq
root@mds1:~#
```

8. Now, configure the archiving file system.

Configure the Archiving File System

The procedure for creating an archiving file system is identical to creating a non-archiving file system, except that we add devices for storing additional copies of the data files:

1. Start by configuring a QFS file system. You can configure either a general-purpose **ms** or high-performance **ma** file system.

   While you can use the Oracle HSM Manager graphical user interface to create file systems, for the examples in this section, we use the **vi** editor. Here, we create a general purpose, **ms** file system with the family set name **hqfs1** and the equipment ordinal number **100**:

```
root@mds1:~# 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/c1t3d0s4        102       md        hqfs1   on
```

2. To add archival tape storage, start by adding an entry for the library. In the equipment identifier field, enter the device ID for the library and assign an equipment ordinal number.

   In this example, the library equipment identifier is **/dev/scsi/changer/c1t0d5**. We set the equipment ordinal number to **700**, the range following the range chosen for the QFS file system that we are using as a disk volume:

```
root@mds1:~# 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/c1t3d0s4        102       md        hqfs1   on
/dev/scsi/changer/c1t0d5 700
```

3. Set the equipment type to **rb**, a generic SCSI-attached tape library, provide a name for the tape library family set, and set the device state **on**.

   In this example, we are using the library **lib1**:

```
root@mds1:~# 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/c1t3d0s4        102       md        hqfs1   on
/dev/scsi/changer/c1t0d5 700       rb        lib1    on
```

4. Optionally, in the **Additional Parameters** column, enter the path where the library catalog will be stored.

   If you do not opt to supply a catalog path, the software will set a default path for you.

   Note that, due to document layout limitations, the example abbreviates the long path to the library catalog **var/opt/SUNWsamfs/catalog/lib1cat**:

```
root@mds1:~# 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/c1t3d0s4        102       md        hqfs1   on
/dev/scsi/changer/c1t0d5 700       rb        lib1    on     ...catalog/lib1cat
```

5. Next, add an entry for each tape drive that is part of the library family set. Add each drive in the order in which it is physically installed in the library.

   Follow the drive order listed in the drive-mapping file that you created in "Determine the Order in Which Drives are Installed in the Library" on page 3-10.

   In the example, the drives attached to Solaris at **/dev/rmt/1**, **/dev/rmt/0**, **/dev/rmt/2**, and **/dev/rmt/3** are, respectively, drives **1**, **2**, **3**, and **4** in the library. So

/dev/rmt/1 is listed first in the **mcf** file, as device **701**. The **tp** equipment type specifies a generic SCSI-attached tape drive:

```
root@mds1:~# 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/c1t3d0s4        102       md        hqfs1  on
/dev/scsi/changer/c1t0d5 700       rb        lib1   on     ...catalog/lib1cat
/dev/rmt/1cbn            701       tp        lib1   on
/dev/rmt/0cbn            702       tp        lib1   on
/dev/rmt/2cbn            703       tp        lib1   on
/dev/rmt/3cbn            704       tp        lib1   on
```

6. To add a cloud storage library, enter the path to the parameters file that defines the equipment in the **Equipment Identifier** field.

   In the example, we enter the path and name of the parameters file that we created above, **/etc/opt/SUNWsamfs/cl800**:

```
root@mds1:~# 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/c1t3d0s4        102       md        hqfs1    on
/dev/scsi/changer/c1t0d5 700       rb        lib1     on     ...catalog/lib1cat
/dev/rmt/0cbn            701       tp        lib1     on
/dev/rmt/1cbn            702       tp        lib1     on
/dev/rmt/2cbn            703       tp        lib1     on
/dev/rmt/3cbn            704       tp        lib1     on
/etc/opt/SUNWsamfs/cl800
```

7. For each cloud storage library, enter an equipment number in the **Equipment Ordinal** field.

   In the example, we assign the equipment ordinal **800**:

```
root@mds1:~# 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/c1t3d0s4        102       md        hqfs1    on
/dev/scsi/changer/c1t0d5 700       rb        lib1     on     ...catalog/lib1cat
/dev/rmt/0cbn            701       tp        lib1     on
/dev/rmt/1cbn            702       tp        lib1     on
/dev/rmt/2cbn            703       tp        lib1     on
/dev/rmt/3cbn            704       tp        lib1     on
/etc/opt/SUNWsamfs/cl800 800
```

8. For each cloud storage library, enter **cr** (*cloud robot*) in the **Equipment Type** field.

```
root@mds1:~# 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/c1t3d0s4        102       md        hqfs1    on
/dev/scsi/changer/c1t0d5 700       rb        lib1     on     ...catalog/lib1cat
/dev/rmt/0cbn            701       tp        lib1     on
/dev/rmt/1cbn            702       tp        lib1     on
/dev/rmt/2cbn            703       tp        lib1     on
/dev/rmt/3cbn            704       tp        lib1     on
/etc/opt/SUNWsamfs/cl800 800       cr
```

9. For each cloud storage library, enter the family set name that you chose when configuring the parameters file in the **Family Set** field, and enter a hyphen (**-**) in both the **Device State** and **Additional Parameters** fields.

   In the example, we use the family set name **cl800**:

```
root@mds1:~# 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/c1t3d0s4        102       md        hqfs1    on
/dev/scsi/changer/c1t0d5 700       rb        lib1     on     ...catalog/lib1cat
/dev/rmt/0cbn            701       tp        lib1     on
/dev/rmt/1cbn            702       tp        lib1     on
/dev/rmt/2cbn            703       tp        lib1     on
/dev/rmt/3cbn            704       tp        lib1     on
/etc/opt/SUNWsamfs/cl800 800       cr        cl800    -      -
```

10. Finally, if you wish to configure a Oracle HSM historian yourself, add an entry using the equipment type **hy**. Enter a hyphen in the family-set and device-state columns and enter the path to the historian's catalog in additional-parameters column.

    The historian is a virtual library that catalogs volumes that have been exported from the archive. If you do not configure a historian, the software creates one automatically using the highest specified equipment ordinal number plus one.

    Note that the example abbreviates the long path to the historian catalog for page-layout reasons. The full path is **/var/opt/SUNWsamfs/catalog/historian_cat**:

```
root@mds1:~# 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/c1t3d0s4        102       md        hqfs1    on
/dev/scsi/changer/c1t0d5 700       rb        lib1     on     ...catalog/lib1cat
/dev/rmt/0cbn            701       tp        lib1     on
/dev/rmt/1cbn            702       tp        lib1     on
/dev/rmt/2cbn            703       tp        lib1     on
/dev/rmt/3cbn            704       tp        lib1     on
/etc/opt/SUNWsamfs/cl800 800       cr        cl800    -      -
historian                999       hy        -        -      .../historian_cat
```

11. Save the **mcf** file, and close the editor.

```
...
/dev/rmt/3cbn            704       tp        lib1     on
/etc/opt/SUNWsamfs/cl800 800       cr        cl800    -      -
```

```
historian              999      hy         -      -      .../historian_cat
:wq
root@mds1:~#
```

12. Check the **mcf** file for errors by running the **sam-fsd** command. Correct any errors found.

    The **sam-fsd** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error:

```
root@mds1:~# sam-fsd
Trace file controls:
...
Would start sam-archiverd()
Would start sam-stagealld()
Would start sam-stagerd()
Would start sam-amld()
root@mds1:~#
```

13. Tell the Oracle HSM software to reread the **mcf** file and reconfigure itself accordingly. Correct any errors reported and repeat as necessary

```
root@mds1:~# /opt/SUNWsamfs/sbin/samd config
Configuring SAM-FS
root@mds1:~#
```

14. Next, mount the archiving file system.

Mount the Archiving File System

1. Log in to the file system host as **root**. Log in to the global zone if the host is configured with zones.

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

2. Create a mount-point directory for the new file system.

```
root@mds1:~# mkdir /hsm/hqfs1
root@mds1:~#
```

3. 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 **/hqfs1** mount-point directory and set permissions to **755** (**-rwxr-xr-x**):

```
root@mds1:~# mkdir /hsm/hqfs1
root@mds1:~# chmod 755 /hsm/hqfs1
root@mds1:~#
```

4. Back up the Solaris **/etc/vfstab** file, and open it in a text editor.

   In the example, we use the **vi** editor.

```
root@mds1:~# cp /etc/vfstab /etc/vfstab.backup
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       -
...
hqfs1      -       /hsm/hqfs1 samfs   -     yes      -
```

5. Set the *high-water mark*, the percentage disk cache utilization that causes Oracle HSM to release previously archived files from disk. In the last column of the Oracle HSM file-system entry, enter the mount option **high=***percentage*, where *percentage* is a number in the range [**0-100**].

   Set this value based on disk storage capacity, average file size, and an estimate of the number of files that are accessed at any given time. You want to make sure that there is always enough cache space for both new files that users create and archived files that users are currently using. But you also want to retain as many files in the cache as possible, so that you can do as little staging as possible. Handling file requests from the disk cache avoids the overhead associated with mounting removable media volumes or recalling files from an Oracle Storage Cloud service.

   If the primary cache is implemented using the latest high-speed disk or solid-state devices or if you are archiving files to the Oracle Storage Cloud, set the high-water mark value at 95%. Otherwise use 80-85%. In the example, we set the high-water mark to 95%:

```
root@mds1:~# cp /etc/vfstab /etc/vfstab.backup
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       -
...
hqfs1      -       /hsm/hqfs1 samfs   -     yes      high=95
```

6. Set the *low-water mark*, the percentage disk cache utilization that causes Oracle HSM to stop releasing previously archived files from disk. In the last column of the Oracle HSM file-system entry, enter the mount option **low=***percentage*, where *percentage* is a number in the range [**0-100**].

   Set this value based on disk storage capacity, average file size, and an estimate of the number of files that are accessed at any given time. You want to keep as many recently active files in cache as you can, particularly when files are frequently requested and modified and when archive copies are stored in an Oracle Storage Cloud account. This keeps staging-related overhead to a minimum. But you do not want previously cached files to consume space needed for new files and for files that have to be staged to disk.

   If the primary cache is implemented using the latest high-speed disk or solid-state devices or if you are archiving files to the Oracle Storage Cloud, set the low-water mark value at 90%. Otherwise use 70-75%. In the example, we set the high-water mark to 90%:

```
root@mds1:~# cp /etc/vfstab /etc/vfstab.backup
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       -
...
hqfs1      -       /hsm/hqfs1 samfs   -     yes      high=95,low=90
```

7. If your users need to retain some file data in the disk cache when previously archived files are released from disk, enter partial releasing mount options in the last column of the Oracle HSM file-system entry.

   Partial releasing lets Oracle HSM leave the first part of a designated file in the disk cache when it releases archived files to recover disk space. This approach gives applications immediate access to the data at the start of the file while the remainder stages from archival media, such as tape. The following mount options govern partial releasing:

   - **maxpartial=***value* sets the maximum amount of file data that can remain in disk cache when a file is partially released to *value*, where *value* is a number of kilobytes in the range **0-2097152** (**0** disables partial releasing). The default is **16**.
   - **partial=***value* sets the default amount of file data that remains in disk cache after a a file is partially released to *value*, where *value* is a number of kilobytes in the range [**0-***maxpartial*]. The default is **16**. But note that the retained portion of a file always uses a kilobytes equal to at least one Disk Allocation Unit (DAU).
   - **partial_stage=***value* sets the minimum amount of file data that must be read before an entire partially released file is staged to *value*, where *value* is a number of kilobytes in the range [**0-***maxpartial*]. The default is the value specified by **-o partial**, if set, or **16**.
   - **stage_n_window=***value* sets the maximum amount of data that can be read at any one time from a file that is read directly from tape media, without automatic staging. The specified *value* is a number of kilobytes in the range [**64-2048000**]. The default is **256**.

     For more information on files that are read directly from tape media, see **OPTIONS** section of the **stage** man page under **-n**.

   In the example, we set **maxpartial** to **128** and **partial** to **64**, based on the characteristics of our application, and otherwise accept default values:

```
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       -
...
hqfs1       -       /hsm/hqfs1 samfs   -     yes  ...maxpartial=128,partial=64
```

8. If you need to exclude QFS file systems from archiving, add the **nosam** mount option to the **/etc/vfstab** entry for each.

   In the example, the **nosam** option is set for the **DISKVOL1** file system, which is a disk archive. Here, the **nosam** mount option makes sure that archival copies are not themselves archived:

```
#File
#Device           Device   Mount               System  fsck  Mount    Mount
#to Mount         to fsck  Point               Type    Pass  at Boot  Options
#--------         -------  ------------------  ------  ----  -------  -------
/devices          -        /devices            devfs   -     no       -
...
hqfs1             -        /hsm/hqfs1          samfs   -     yes      ...=64
DISKVOL1          -        /diskvols/DISKVOL1  samfs   -     yes      nosam
server:/DISKVOL2  -        /diskvols/DISKVOL2  nfs     -     yes
```

```
...
server:/DISKVOL15 -        /diskvols/DISKVOL15 nfs     -     yes
```

9. Save the **/etc/vfstab** file, and close the editor.

```
...
server:/DISKVOL15 -        /diskvols/DISKVOL15 nfs     -     yes
:wq
root@mds1:~#
```

10. Mount the Oracle HSM archiving file system.

```
root@mds1:~# mount /hsm/hqfs1
root@mds1:~#
```

11. Next, configure the archiving process.

Configure the Archiving Process

Once archiving file systems have been created and mounted, you can generally address all or most of your archiving requirements with little additional configuration. In most cases, you need do little more than create a text file, **archiver.cmd**, that identifies the file systems, specifies the number of archive copies of each of your, and assigns media volumes to each copy.

While the Oracle HSM archiving process does have a number of tunable parameters, you should generally accept the default settings in the absence of well-defined, special requirements. The defaults have been carefully chosen to minimize the number of media mounts, maximize utilization of media, and optimize end-to-end archiving performance in the widest possible range of circumstances. So if you do need to make adjustments, be particularly careful about any changes that unnecessarily restrict the archiver's freedom to schedule work and select media. If you try to micromanaging storage operations, you can reduce performance and overall efficiency, sometimes drastically.

You should, however, enable archive logging in almost all situations. Archive logging is not enabled by default, because the log files can reach excessive sizes if not properly managed (management is covered in the *Oracle Hierarchical Storage Manager and StorageTek QFS Software Maintenance and Administration Guide*). But, if a file system is ever damaged or lost, the archive log file lets you recover files that cannot otherwise be easily restored. When you configure protection for a file system, the file-system metadata in a recovery point file lets you rapidly rebuild a file system from the data stored in archive copies. But a few files are inevitably archived *before* the file system is damaged or lost but *after* the last recovery point is generated. In this situation, the archival media holds valid copies, but, in the absence of file-system metadata, the copies cannot be automatically located. Since the file system's archive log records the volume serial numbers of the media that holds each archive copy and the position of the corresponding **tar** file(s) within each volume, you can use **tar** utilities to recover these files and fully restore the file system.

To create the **archiver.cmd** file and configure the archiving process, proceed as follows:

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

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

2. Open a new **/etc/opt/SUNWsamfs/archiver.cmd** file in a text editor.

Each line in an **archiver.cmd** consists of one or more fields separated by white space (leading white space is ignored).

In the example, we use the **vi** editor to open the file and enter a comment:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/archiver.cmd
# Configuration file for archiving file systems
```

3. At the beginning of the **archiver.cmd** file, enter any general archiving directives that you need.

   General directives contain the equals (**=**) character in the second field or have no additional fields. In most cases, you can use the default values instead of setting general directives (see the **GENERAL DIRECTIVES SECTION** of the **archiver.cmd** man page for details).

   While we could leave this section empty, in the example, we have entered the default values for two general directives to illustrate their form:

   - The **archivemeta = off** directive tells the archiving process that it should not archive metadata.
   - The **examine = noscan** directive tells the archiving process to check for files that need archiving whenever the file system reports that files have changed (the default).

     Older versions of Oracle HSM scanned the whole file system periodically. In general, you should not change this directive unless you must do so for compatibility with legacy Oracle HSM configurations.

```
# Configuration file for archiving file systems
#-----------------------------------------------------------------------
# General Directives
archivemeta = off                                               # default
examine = noscan                                                # default
```

4. Once you have entered all required general archiving directives, start assigning files to archive sets. On a new line, enter the assignment directive **fs =** *filesystem-name*, where *filesystem-name* is the family set name for a file system defined in the **/etc/opt/SUNWsamfs/mcf** file.

   The assignment directive maps a set of files in the specified file system to a set of copies on archival media. A set of files can be as large as all file systems or as small as a few files. But, for best performance and efficiency, you should not over-specify. Do not create more archive sets than you need to, as this can cause excessive media mounts, needless repositioning of media, and poor overall media utilization. In most cases, assign one archive set per file system.

   In the example, we start the archive-set assignment directive for the archiving file system **hqfs1**:

```
# Configuration file for archiving file systems
#-----------------------------------------------------------------------
# General Directives
archivemeta = off                                               # default
examine = noscan                                                # default
#-----------------------------------------------------------------------
# Archive Set Assignments
fs = hqfs1
```

5. On the next line, enable archive logging. Enter the **logfile =** *path/filename* directive, where *path/filename* specifies the location and file name.

As noted above, archives log data are essential for a complete recovery following loss of a file system. So configure Oracle HSM to write the archiver log to a non-Oracle HSM directory, such as **/var/adm/**, and save copies regularly. While you can create a global **archiver.log** that records archiver activity for all file systems together, configuring a log for each file system makes it easier to search the log during file recovery. So, in the example, we specify **/var/adm/hqfs1.archiver.log** here, with the file-system assignment directives:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/archiver.cmd
...
#-----------------------------------------------------------------------
# Archive Set Assignments
fs = hqfs1
logfile = /var/adm/hqfs1.archiver.log
```

6. Next, assign any files that should never be archived to the special **no_archive** set.

   Use a directory path and, optionally, archive set assignment parameters to identify the files that should be included in the **no_archive** set. See the **archiver.cmd** (4) man page for details.

   In the example, the **no_archive** set includes all files in the **hqfs1** file system that reside on a path matching the regular expression specified by the **-name** parameter. The regular expression matches temporary and backup files found under the path **/hsm/hqfs1/data/**:

```
...
#-----------------------------------------------------------------------
# Archive Set Assignments
fs = hqfs1
logfile = /var/adm/hqfs1.archiver.log
no_archive . -name \/hsm\/hqfs1\/data\/((tmp|bak)\/.*)|([.A-Za-z0-9_-]+\.tmp)
```

7. On the next line, organize the files that will be archived into archive sets. For each archive set that you need to create, enter the directive *archiveset-name starting-directory expression*, where:

   - *archiveset-name* is the name that you choose for new the archive set.
   - *starting-directory* is the path to the directory where Oracle HSM starts to search for the files that belong in the set (relative to the file-system mount point).
   - *expression* is one of the Boolean expressions defined by the Solaris **find** command.

   You should keep archive set definitions as inclusive and simple as possible in most cases. But note that, when circumstances dictate, you can limit archive set membership by specifying additional, more restrictive qualifiers, such as user or group file ownership, file size, file date/time stamps, and file names (using regular expressions). See the **archiver.cmd** man page for full information.

   In the first example, we put all files found in the **hqfs1** file system in a single archive set named **allhqfs1**. We specify the path using a dot (**.**) to start the search in the mount point directory itself (**/hqfs1**).

```
...
#-----------------------------------------------------------------------
# Archive Set Assignments
fs = hqfs1
logfile = /var/adm/hqfs1.archiver.log
no_archive . -name \/hsm\/hqfs1\/data\/((tmp|bak)\/.*)|([.A-Za-z0-9_-]+\.tmp)
```

```
allhqfs1 .
```

In the second example, we define an archive set named **inactive** before we define the **allhqfs1** archive set. The **inactive** archive set includes all files (**.**) in the **hqfs1** file system that have not been accessed for at least one year (**-access 1y**). The Oracle HSM archiver processes directives in the order in which the archive sets are defined. So the **inactive** set definition lets us restrict the **allhqfs1** set to actively used files. We can then define archiving policies that store dormant files on low-cost, long term storage media, such as the Oracle Storage Cloud, and active files on disk archives and tape:

```
...
#-----------------------------------------------------------------------
# Archive Set Assignments
fs = hqfs1
logfile = /var/adm/hqfs1.archiver.log
no_archive . -name \/hsm\/hqfs1\/data\/((tmp|bak)\/.*)|([.A-Za-z0-9_-]+\.tmp)
inactive . -access 1y
allhqfs1 .
```

8. Next, add copy directives for each archive set. For each copy, start the line with one or more spaces, and enter the directive *copy-number* **-release -norelease** *archive-age unarchive-age*, where:

   - *copy-number* is an integer.
   - **-release** and **-norelease** are optional parameters that control how disk cache space is managed once copies have been made. On its own, **-release** causes the disk space to be automatically released as soon as the corresponding copy is made. On its own, **-norelease** prevents release of disk space until **all** copies that have **-norelease** set have been made **and** the releaser process has been run. Together, **-release** and **-norelease** automatically release disk cache space as soon as all copies that have **-norelease** set have been made.
   - *archive-age* is the time that must elapse from the time when the file was last modified before it is archived. Express time as any combination of integers and the identifiers **s** (seconds), **m** (minutes), **h** (hours), **d** (days), **w** (weeks) and **y** (years). The default is **4m**.

     If the first copy of the archive set will be archived to a cloud service, set the archive age to 30 minutes (**30m**) or more. The greater archive age allows time for changes to accumulate prior to archiving. If files are archived every time changes are made, the cloud resources accumulate large numbers of old, stale copies, increasing costs.
   - *unarchive-age* is the time that must elapse from time when the file was last modified before it can be unarchived. The default is to never unarchive copies.

   For full redundancy, always *specify at least two copies of each archive set* (the maximum is four).

   In the example, we specify one copy for archive set **inactive** and set the archive age to one year. We specify three copies for archive set **allhqfs1** and set a different archive age for each. Copy **allhqfs1.1** will be made when files are 15 minutes old. Copy **allhqfs1.2** will be made files are 24 hours old. Copy **allhqfs1.3** will be made to tape media when files are 48 hours old.

```
...
#-----------------------------------------------------------------------
# Archive Set Assignments
```

```
fs = hqfs1
logfile = /var/adm/hqfs1.archiver.log
no_archive . -name \/hsm\/hqfs1\/data\/((tmp|bak)\/.*)|([.A-Za-z0-9_-]+\.tmp)
inactive . -access 1y
    1 1y
allhqfs1 .
    1 15m
    2 24h
    3 48h
```

9. Define archive sets for any remaining file systems.

   In the example, we have configured an additional QFS file system, **DISKVOL1**, for use as archival disk media only. We do not want to be making archival copies of archival copies. So we start an entry for **fs = DISKVOL1** and include all files in the **no_archive** set:

```
...
#-----------------------------------------------------------------------
# Archive Set Assignments
fs = hqfs1
logfile = /var/adm/hqfs1.archiver.log
no_archive . -name \/hsm\/hqfs1\/data\/((tmp|bak)\/.*)|([.A-Za-z0-9_-]+\.tmp)
inactive . -access 1y
    1 1y
allhqfs1 .
    1 15m
    2 45m
    3 24h
fs = DISKVOL1                       # QFS File System (Archival Media)
no_archive .
```

10. Next we enter the directives that govern how copies are created. On a new line, start the copy parameters section by entering the key word **params**.

```
...
fs = DISKVOL1                         # QFS File System (Archival Media)
no_archive .
#-----------------------------------------------------------------------
# Copy Parameter Directives
params
```

11. To set common copy parameters that apply to all copies of all archive sets, enter a line of the form **allsets** *parameter-list* where:

    - **allsets** is the special archive set that represents all configured archive sets.
    - *parameter-list* is a sequence of parameter/value pairs separated by spaces. Each pair takes the form **-***parameter-name value*.

      See the see the **ARCHIVE SET COPY PARAMETERS SECTION** of the **archiver.cmd** (4) man page for a full list of parameter names and possible values.

    The directive in the example is optimal for most file systems. The special **allsets** archive set insures that all archive sets are handled uniformly, for optimal performance and ease of management. The **-sort path** parameter insures that the tape archive (**tar**) files for all copies of all archive sets are sorted by path, so that files in the same directories remain together on the archive media. The **-offline_copy stageahead** parameter can improve performance when archiving offline files. The **-reserve set** parameter insures that files are always copied to media dedicated for the use of each archive set:

```
...
#-----------------------------------------------------------------------
# Copy Parameter Directives
params
allsets -sort path -offline_copy stageahead -reserve set
```

12. To set copy parameters specific to each copy of all archive sets, enter a line of the form **`allsets.`***`copy-number parameter-list`*, where:

    - **`allsets`** is the special archive set that represents all configured archive sets.
    - *`copy-number`* is the number of the copy to which the parameters apply
    - *`parameter-list`* is a sequence of parameter/value pairs separated by spaces. Each pair takes the form **`-`***`parameter-name value`*.

    See the see the **`ARCHIVE SET COPY PARAMETERS SECTION`** of the **`archiver.cmd`** (4) man page and the "Copy Parameters" section of Appendix C for full information on copy parameters and values.

    The examples combine tailor copy jobs to archiving requirements with just a few, commonly used parameters:

    - The **`-startage`** and **`-startsize`** control the start of archiving. Archiving starts when an amount of time specified by **`-startage`** has elapsed since the earliest modification date for a file in the archive set and/or when the aggregate size of the files in the archive set exceeds the size specified by **`-startsize`**.
    - The **`-drives`** and **`-archmax`** parameters control drive utilization. The archiver can use no more than the number of tape devices specified by **`-drives`** and can create archive (**`.tar`**) files no larger than the size specified by **`-archmax`**.

    In the first example, the copy parameters optimize copy **`allsets.1`** for promptly backing up small, frequently modified user files to disk volumes. Archiving starts when the first file selected for archiving has been waiting for 15 minutes or when the total size of all waiting files is at least 500 megabytes. A maximum of 10 drives can be used to make the copy and each **`tar`** file in the copy need be no larger than one gigabyte.

    The copy parameters optimize the first tape copy, **`allsets.2`**, for staging files that users request after the disk-cached copy has been released. Since these files are being actively used,**`-startage`** and **`-startsize`** insure that modified files are always archived within 24 hours or whenever 20 gigabytes of modified files have accumulated. The **`-drives`** parameter insures that archiving does not monopolize the available devices at the expense of staging, while **`-archmax`** limits the archive files to a size that is large enough to be efficiently written to tape but small enough to efficiently stage requested files.

    The copy parameters optimize **`allsets.3`** for data migration and disaster recovery. Oracle HSM starts archiving to the second tape copy when the first file selected for archiving has been waiting for 48 hours or when the total size of all waiting files is at least 50 gigabytes. Each **`tar`** file in the copy can be no larger than 55 gigabytes. The larger maximum file size and greater start age increase the efficiency of writes to and reads from tape by eliminating the overhead associated with writing a larger numbers of smaller files.

```
...
#-----------------------------------------------------------------------
# Copy Parameter Directives
params
allsets -sort path -offline_copy stageahead -reserve set
allsets.1 -startage 15m -startsize 500M -drives 10 -archmax 1G
allsets.2 -startage 24h -startsize 20G -drives 2 -archmax 24G
allsets.3 -startage 48h -startsize 50G -drives 2 -archmax 55G
```

In the second example, copy parameters optimize **`allsets.1`** for fast transfer to tape, using multiple drives in parallel. The **`-drives`** parameter makes more devices available to the archiver. But one archive (**`.tar`**) file can be written to exactly one drive. So **`-archmax`** specifies a smaller maximum file size to insure that the copy creates enough archive files to utilize the additional drives:

```
allsets.1 -startage 8h -startsize 8G -drives 6 -archmax 10G
```

13. To set copy parameters for individually specified archive sets, enter a line of the form *`archive-set`*`.`*`copy-number parameter-list`*, where:

    - *`archive-set`* is the name of the archive set as specified in the File System Directives section of the file.
    - *`copy-number`* is the number of the copy to which the parameters apply
    - *`parameter-list`* is a sequence of parameter/value pairs separated by spaces. Each pair takes the form `-`*`parameter-name value`*.

      See the see the **`ARCHIVE SET COPY PARAMETERS SECTION`** of the **`archiver.cmd`** (4) man page for a full list of parameter names and possible values.

    In the example, the directive **`inactive.1`** specifies one copy of the dormant files in the **`inactive`** archive set. The copy is made once the oldest unarchived in the set has not been modified for one year or once the total size of the files exceeds 250 megabytes. The dormant files will be stored remotely, in the Oracle Storage Cloud, so the parameter **`-drives 2`** specifies the number of streams that will be sent to the cloud. The first copy directive for the **`allhqfs1`** archive set, **`allhqfs1.1`**, insures that frequent changes to data files are backed up promptly to archival disk. The second directive, **`allhqfs1.2`**, optimizes the archived files for staging from tape to the disk cache. The third directive, **`allhqfs1.3`**, optimizes the archived files for disaster recovery or migration to new media:

```
...
#-----------------------------------------------------------------------
# Copy Parameter Directives
params
allsets -sort path -offline_copy stageahead -reserve set
inactive.1 -startage 1y -startsize 250M -drives 2 -archmax 1G
allhqfs1.1 -startage 10m -startsize 500M -drives 10 -archmax 1G
allhqfs1.2 -startage 24h -startsize 20G -drives 2 -archmax 24G
allhqfs1.3 -startage 48h -startsize 50G -drives 2 -archmax 55G
```

14. When you have set all required copy parameters, close the copy parameters list by entering the **`endparams`** keyword on a new line:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/archiver.cmd
...
#-----------------------------------------------------------------------
# Copy Parameter Directives
params
inactive.1 -startage 1y -startsize 250m -drives 2 -archmax 1G
allsets -sort path -offline_copy stageahead -reserve set
allhqfs1.1 -startage 10m -startsize 500M -drives 10 -archmax 1G
allhqfs1.2 -startage 24h -startsize 20G -drives 2 -archmax 24G
allhqfs1.3 -startage 48h -startsize 50G -drives 2 -archmax 48G
endparams
```

15. Optionally, you can define media pools by entering the **vsnpools** keyword, one or more directives of the form *pool-name media-type volumes*, where:

    - *pool-name* is the name that you have assigned to the pool.
    - *media-type* is one of the media type codes defined in Appendix A, "Glossary of Equipment Types".
    - *volumes* is a list of volume serial numbers (VSNs) or a regular expression that matches one or more volume serial numbers.

    You can substitute the name of the media pool for a range of VSNs when assigning media for a copy. If you define media pools, avoid excessively restricting the media available to the archiving process.

```
root@mds1:~# vi /etc/opt/SUNWsamfs/archiver.cmd
...
#-----------------------------------------------------------------------
# VSN Pool Definitions
vsnpools
pool1 ti TP9[0-2][0-9][0-9] TP9[5-6][0-9][0-9]
pool2 ti TP9[3-4][0-9][0-9] TP9[7-8][0-9][0-9]
```

16. Close the VSN Pool Definitions list with the **endvsnpools** keyword.

```
root@mds1:~# vi /etc/opt/SUNWsamfs/archiver.cmd
...
#-----------------------------------------------------------------------
# VSN Pool Definitions
vsnpools
pool1 li VOL90[0-4]
pool2 li VOL90[5-9]
endvsnpools
```

17. Next, start identifying the archival media that your archive set copies should use. On a new line, enter the keyword **vsns**:

```
...
#-----------------------------------------------------------------------
# VSN Directives
vsns
```

18. Specify media for each archive-set copy by entering a line of the form *archive-set-name.copy-number media-type volumes*, where:

    - *archive-set-name.copy-number* specifies the archive set and copy to which the directive applies.
    - *media-type* is one of the media type codes defined in Appendix A, "Glossary of Equipment Types"
    - *volumes* is a regular expression that matches one or more volume serial numbers (VSNs).

    For full redundancy, always assign each archive set copy to a different range of media, so that both copies never reside on the same physical volume. If possible, always assign at least one copy to removable media, such as tape.

    In the example, we archive files in the **inactive** archive set to Oracle Storage Cloud volumes (type **cl**) that have volume serial numbers starting with **cl800**, the **name** prefix that we assigned when setting up the cloud-storage parameters file. We send the first copy of the files in the **allhqfs1** archive set to archival disk volumes (type **dk**) that have serial numbers in the range **DISKVOL1** to **DISKVOL15**.

We send the second copy of the files in **allhqfs1** to tape volumes (type **tp**) that have volume serial numbers in the range **VOL000** to **VOL399**. We send the third copy to tape volumes that have volume serial numbers in the range **VOL400** to **VOL899**:

```
...
#-----------------------------------------------------------------------
# Copy Parameter Directives
params
allsets -sort path -offline_copy stageahead -reserve set
allhqfs1.1 -startage 10m -startsize 500M -drives 10 -archmax 1G
allhqfs1.2 -startage 24h -startsize 20G -drives 2 -archmax 24G
allhqfs1.3 -startage 48h -startsize 50G -drives 2 -archmax 55G
endparams
#-----------------------------------------------------------------------
# VSN Directives
vsns
inactive.1 cl cl800.*
allhqfs1.1 dk DISKVOL[1-15]
allhqfs1.2 tp VOL[0-3][0-9][0-9]
allhqfs1.3 tp VOL[4-8][0-9][0-9]
```

19. When you have specified media for all archive-set copies, close the **vsns** directives list by entering the **endvsns** keyword on a new line. Save the file and close the editor.

```
...
#-----------------------------------------------------------------------
# Copy Parameter Directives
params
allsets -sort path -offline_copy stageahead -reserve set
allfiles.1 -startage 10m -startsize 500M -drives 10 -archmax 1G
allfiles.2 -startage 24h -startsize 20G -drives 2 -archmax 24G
allfiles.3 -startage 48h -startsize 50G -drives 2 -archmax 55G
endparams
#-----------------------------------------------------------------------
# VSN Directives
vsns
inactive.1 cl cl800.*
allhqfs1.1 dk DISKVOL[1-15]
allhqfs1.2 tp VOL[0-3][0-9][0-9]
allhqfs1.3 tp VOL[4-8][0-9][0-9]
endvsns
:wq
root@mds1:~#
```

20. 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. In the example, we have an error:

```
root@mds1:~# archiver -lv
Reading '/etc/opt/SUNWsamfs/archiver.cmd'.
...
13: #File System Directives
14: #
15: fs = hqfs1
16: logfile = /var/adm/hqfs1.archiver.log
17: all .
18:     1 -norelease 15m
19:     2 -norelease 15m
```

```
20: fs=DISKVOL1                          # QFS File System (Archival Media)
21:
...
42: endvsns
DISKVOL1.1 has no volumes defined
1 archive set has no volumes defined
root@mds1:~#
```

21. If errors were found in the **archiver.cmd** file, correct them, and then re-check the file.

    In the example above, we forgot to enter the **no_archive** directive to the file-system directives **DISKVOL1**, the QFS file system that we configured as a disk archive. When we correct the omission, **archiver -lv** runs without errors:

```
root@mds1:~# archiver -lv
Reading '/etc/opt/SUNWsamfs/archiver.cmd'.
...
20: fs=DISKVOL1                          # QFS File System (Archival Media)
21: no_archive .
...
42: endvsns
Notify file: /etc/opt/SUNWsamfs/scripts/archiver.sh
...
allfiles.1
    startage: 10m startsize: 500M drives 10: archmax: 1G
 Volumes:
  DISKVOL1 (/diskvols/DISKVOL15)
  ...
  DISKVOL15 (/diskvols/DISKVOL3)
 Total space available:   150T
allfiles.2
    startage: 24h startsize: 20G drives: 2 archmax: 24G reserve: set
 Volumes:
   VOL000
...
   VOL199
 Total space available:  300T
allfiles.3
    startage: 48h startsize: 20G drives: 2 archmax: 24G reserve: set
 Volumes:
   VOL200
...
   VOL399
 Total space available:  300T
root@mds1:~#
```

22. Tell the Oracle HSM software to reread the **archiver.cmd** file and reconfigure itself accordingly. Use the **samd config** command.

```
root@mds1:~# /opt/SUNWsamfs/sbin/samd config
Configuring SAM-FS
root@mds1:~#
```

23. Open the **/etc/opt/SUNWsamfs/releaser.cmd** file in a text editor, add the line **list_size = 300000**, save the file, and close the editor.

    The **list_size** directive sets the number of files that can be released from a file system at one time to an integer in the range [**10-2147483648**]. If there is enough space in the **.inodes** file for one million inodes (allowing 512- bytes per inode), the default value is **100000**. Otherwise the default is **30000**. Increasing this number to

300000 better suits typical file systems that contain significant numbers of small files.

In the example, we use the **vi** editor:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/releaser.cmd
#releaser.cmd
logfile = /var/opt/SUNWsamfs/releaser.log
list_size = 300000
:wq
root@mds1:~#
```

24. Open the **/etc/opt/SUNWsamfs/stager.cmd** file in a text editor, and add the line **maxactive =** *stage-requests*, where *stage-requests* is **500000** on hosts that have 8 gigabytes of RAM or more and **100000** on hosts that have less than 8 gigabytes. Save the file, and close the editor.

    The **maxactive** directive sets the maximum number of stage requests that can be active at one time to an integer in the range [**1-500000**]. The default is to allow 5000 stage requests per gigabyte of host memory.

    In the example, we use the **vi** editor:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/stager.cmd
#stager.cmd
logfile = /var/opt/SUNWsamfs/stager.log
maxactive = 300000
:wq
root@mds1:~#
```

25. Recycling is not enabled by default. So, if you require recycling of removable media volumes, go to "Configuring the Recycling Process" on page 6-42.
26. If the **mcf** file for the archiving Oracle HSM file system includes a network-attached tape library in the archiving equipment section, go to "Catalog Archival Media Stored in a Network-Attached Tape Library" on page 6-48.
27. If you need to be able to verify the data integrity of archival tape volumes, go to "Configure Archival Media Validation" on page 6-56.
28. Otherwise, "Configure File System Protection" on page 6-50.

Configuring the Recycling Process

When removable media volumes contain fewer than a user-specified number of valid archive sets, the recycler consolidates the valid data on other volumes so that the original volumes can be exported for long-term storage or relabeled for reuse. You can configure recycling in either of two ways:

- You can configure recycling by archive set.

  When you recycle media by archive set, you add recycling directives to the **archiver.cmd** file. You can specify exactly how media in each archive set copy is recycled. Recycling criteria are more narrowly applied, since only members of the archive set are considered.

  Where possible, recycle media by archive sets. In an Oracle HSM archiving file system, recycling is logically part of file-system operation rather than library management. Recycling complements archiving, releasing, and staging. So it makes sense to configure it as part of the archiving process.

Note that you must configure recycling by archive sets if your configuration includes disk-archive volumes and/or SAM-Remote.

- You can configure recycling by library.

  Recycling by library makes most sense when the logistical aspects of accessing the storage make managing the library as a unit desirable. Recycling cloud volumes is a good example.

  When you recycle media by library, you add recycling directives to a **`recycler.cmd`** file. You can thus set common recycling parameters for all media contained in a specified library. Recycling directives apply to all volumes in the library, so they are inherently less granular than archive set-specific directives. You can explicitly exclude specified volume serial numbers (VSNs) from examination. But otherwise, the recycling process simply looks for volumes that contain anything that it does not recognize as a currently valid archive file.

  As a result, recycling by library can destroy files that are not part of the file system that is being recycled. If a recycling directive does not explicitly exclude them, useful data, such as back up copies of archive logs and library catalogs or archival media from other file systems, may be at risk.

  For this reason, you cannot recycle by library if you are using SAM-Remote. Volumes in a library controlled by a SAM-Remote server contain foreign archive files that are owned by clients rather than by the server.

Configure Recycling by Archive Set

1. Log in to the Oracle HSM file-system host as **`root`**.

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

2. Open the **`/etc/opt/SUNWsamfs/archiver.cmd`** file in a text editor, and scroll down to the copy **`params`** section.

   In the example, we use the **`vi`** editor.

```
root@mds1:~# vi /etc/opt/SUNWsamfs/archiver.cmd
...
#-----------------------------------------------------------------------
# Copy Parameter Directives
params
allsets -sort path -offline_copy stageahead
allfiles.1 -startage 6h  -startsize 6G  -startcount 500000
allfiles.2 -startage 24h -startsize 20G -startcount 500000 -drives 5
```

3. In the **`params`** section of the **`archiver.cmd`** file, enter your recycler directives by archive set, in the form *`archive-set directive-list`*, where archive-set is one of the archive sets and *`directive-list`* is a space-delimited list of directive name/value pairs (for a list of recycling directives, see the **`archiver.cmd`** man page). Then save the file and close the editor.

   In the example, we add recycling directives for archive sets **`allfiles.1`** and **`allfiles.2`**. The **`-recycle_mingain 30`** and **`-recycle_mingain 90`** directives do not recycle volumes unless, respectively, at least 30 percent and 90 percent of the volume's capacity can be recovered. The **`-recycle_hwm 60`** directive starts recycling when 60 percent of the removable media capacity has been used.

```
root@mds1:~# vi /etc/opt/SUNWsamfs/archiver.cmd
...
#-----------------------------------------------------------------------
# Copy Parameters Directives
```

```
params
allsets -sort path -offline_copy stageahead
allfiles.1 -startage 6h  -startsize 6G -startcount 500000
allfiles.1 -recycle_mingain 30 -recycle_hwm 60
allfiles.2 -startage 6h -startsize 6G -startcount 500000
allfiles.2 -recycle_mingain 90 -recycle_hwm 60
endparams
#-----------------------------------------------------------------------
# VSN Directives
vsns
allfiles.1 dk DISKVOL1
allfiles.2 tp VOL0[0-1][0-9]
endvsns
:wq
[root@mds1:~#
```

4. Check the **archiver.cmd** file for errors. Use the command **archiver -lv**.

   The command **archiver -lv** reads the **archiver.cmd** and generates a configuration report if no errors are found. Otherwise, it notes any errors and stops. In the example, the file does not contain any errors:

   ```
   root@mds1:~# archiver -lv
   Reading '/etc/opt/SUNWsamfs/archiver.cmd'.
   ...
      VOL399
    Total space available:  300T
   root@mds1:~#
   ```

5. If errors were found in the **archiver.cmd** file, correct them, and then re-check the file.

6. Create the **recycler.cmd** file in a text editor. Specify a path and file name for the recycler log. Then save the file and close the editor.

   Configure Oracle HSM to write logs to a non-Oracle HSM directory, such as **/var/adm/**. In the example, we use the **vi** editor, and specify **/var/adm/recycler.log**:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/recycler.cmd
   logfile = /var/adm/recycler.log
   :wq
   root@mds1:~#
   ```

7. Next, customize the **/etc/opt/SUNWsamfs/scripts/recycler.sh** script to correctly handle recycled volumes.

Configure Recycling by Library

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

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

2. Create the **/etc/opt/SUNWsamfs/recycler.cmd** file in a text editor.

   In the example, we use the **vi** editor.

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/recycler.cmd
   # Configuration file for archiving file systems
   #-----------------------------------------------------------------------
   ```

3. Specify a path and file name for the recycler log using the **logfile** directive.

Configure Oracle HSM to write logs to a non-Oracle HSM directory, such as **/var/adm/**. In the example, we specify **/var/adm/recycler.log**:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/recycler.cmd
# Configuration file for archiving file systems
#-----------------------------------------------------------------------
logfile = /var/adm/recycler.log
```

4. If there are any volumes in the archival media library that must not be recycled, enter the directive **no_recycle** *media-type volumes*, where *media-type* is one of the media type codes defined in Appendix A, "Glossary of Equipment Types" and *volumes* is a regular expression that matches one or more volume serial numbers (VSNs).

   In the example, we disable recycling for volumes in the range [**VOL020-VOL999**]:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/recycler.cmd
# Configuration file for archiving file systems
#-----------------------------------------------------------------------
logfile = /var/adm/recycler.log
no_recycle tp VOL[0-9][2-9][0-9]
```

5. On a new line, enter the directive *library parameters*, where *library* is the family set name that the **/etc/opt/SUNWsamfs/mcf** file assigns to a removable media library and where *parameters* is a space-delimited list of parameter/value pairs drawn from the following list:

   - **-dataquantity** *size* sets the maximum amount of data that can be scheduled for rearchiving at one time to *size*, where *size* is a number of bytes. The default is 1 gigabyte.
   - **-hwm** *percent* sets the library's *high-water mark*, the percentage of the total media capacity that, when used, triggers recycling. The high-water mark is specified as *percent*, a number in the range [**0-100**]. The default is **95**.
   - **-ignore** prevents recycling for this library, so that you can test the **recycler.cmd** file non-destructively.
   - **-mail** *address* sends recycling messages to *address*, where *address* is a valid email address. By default, no messages are sent.
   - **-mingain** *percent* limits recycling to volumes that can increase their available free space by at least a minimum amount, expressed as a percentage of total capacity. This minimum gain is specified as *percent*, a number in the range [**0-100**]. The defaults are **60** for volumes with a total capacity under 200 gigabytes and **90** for capacities of 200 gigabytes or more.
   - **-vsncount** *count* sets the maximum number of volumes that can be scheduled for rearchiving at one time to *count*. The default is **1**.

   In the example, we set the high-water mark for library **library1** to 95% and require a minimum capacity gain per cartridge of 60%:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/recycler.cmd
# Configuration file for archiving file systems
#-----------------------------------------------------------------------
logfile = /var/adm/recycler.log
no_recycle tp VOL[0-9][2-9][0-9]
library1 -hwm 95 -mingain 60
```

6. Repeat the preceding step for any other libraries that are part of the Oracle HSM configuration.

In the example, we configure recycling of volumes in the cloud-storage library **cl800**. When recycling cloud-resident volumes, we want to avoid the additional overhead and increased cost of moving still-active files from recycling candidates to other volumes. We want to recycle volumes that can be immediately relabeled and reused, without rearchiving any files. So the directive recycles the first 100 VSNs in the library (**-vsncount 100**) that hold no active data. The **-dataquantity** parameter recycles volumes that hold no more than one byte of data, and the **-hwm** and **-mingain** parameters let recycling proceed regardless of the percentage of capacity currently used or to be gained. The recycler log lists cartridges that meet these criteria as **no-data VSN** volumes.

```
root@mds1:~# vi /etc/opt/SUNWsamfs/recycler.cmd
# Configuration file for archiving file systems
#-----------------------------------------------------------------------
logfile = /var/adm/recycler.log
no_recycle tp VOL[0-9][2-9][0-9]
library1 -hwm 95 -mingain 60
cl800 -vsncount 100 -dataquantity 1b -hwm 1 -mingain 1
root@mds1:~#
```

7. Then save the **recycler.cmd** file, and close the editor.

```
root@mds1:~# vi /etc/opt/SUNWsamfs/recycler.cmd
# Configuration file for archiving file systems
#-----------------------------------------------------------------------
logfile = /var/adm/recycler.log
no_recycle tp VOL[0-9][2-9][0-9]
library1 -hwm 95 -mingain 60
cl800 -vsncount 100 -dataquantity 1b -hwm 1 -mingain 1
:wq
root@mds1:~#
```

8. Next, customize the **/etc/opt/SUNWsamfs/scripts/recycler.sh** script to correctly handle the recycled volumes.

Customize the `recycler.sh` Script to Handle Recycled Media Per Requirements

When the recycling process identifies a removable media volume that has been drained of valid archive copies, it calls the **recycler.sh** file, a C-shell script designed to handle disposition of recycled media. You edit this script to perform the tasks that you need, from notifying administrators that volumes are ready for recycling to relabeling the volumes for reuse or exporting them from the library.

By default, the script reminds the **root** user to set up the script.

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

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

2. Open the file **/etc/opt/SUNWsamfs/scripts/recycler.sh** in a text editor.

   In the example, we use the **vi** editor:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/scripts/recycler.sh
#!/bin/csh -f
#    SAM-QFS_notice_begin
...
#    SAM-QFS_notice_end
```

3. Read and abide by the terms set in the license text at the head of the file.

4. Enable logging by uncommenting the line indicated in the script comments. If necessary, specify an alternate path for the file.

```
root@mds1:~# vi /etc/opt/SUNWsamfs/scripts/recycler.sh
...
#   It is a good idea to log the calls to this script
echo `date` $* >>  /var/opt/SUNWsamfs/recycler.sh.log
#   As an example, if uncommented, the following lines will relabel the VSN,
#   if it exists in a physical library.  If the VSN is in the historian
```

5. To relabel recycled volumes that are resident in the library and notify root of any recycled, off-site volumes, uncomment the lines indicated in the script comments. Make changes as needed to suit local requirements.

```
#   As an example, if uncommented, the following lines will relabel the VSN,
#   if it exists in a physical library.  If the VSN is in the historian
#   catalog (e.g., it's been exported from a physical library and moved
#   to off-site storage), then email is sent to "root" informing that the
#   medium is ready to be returned to the site and reused.
#
set stat=0
if ( $6 != hy ) then
    /opt/SUNWsamfs/sbin/chmed -R $5.$2
    /opt/SUNWsamfs/sbin/chmed -W $5.$2
    if ( $1 != "od" ) then
        /opt/SUNWsamfs/sbin/${1}label -w -vsn $2 -old $2 $4\:$3
          if ( $status != 0 ) then
              set stat = 1
          endif
    else
        /opt/SUNWsamfs/sbin/${1}label -w -vsn $2 -old $2 $4\:$3\:$7
          if ( $status != 0 ) then
              set stat = 1
          endif
    endif
else
    mail root <</eof
VSN $2 of type $5 is devoid of active archive images. It is currently
in the historian catalog, which indicates that it has been exported
...
/eof
endif
echo `date` $* done >>  /var/opt/SUNWsamfs/recycler.sh.log
if ( $stat != 0 ) then
     exit 1
else
     exit 0
endif
# These lines would inform "root" that the VSN should be removed
```

6. If you wish to export volumes that contain no current data files for long-term off-site storage, uncomment the lines indicated in the script comments. Make changes as needed to suit local requirements.

```
# These lines would inform "root" that the VSN should be removed
# from the robotic library:
mail root <</eof
VSN $2 in library $4 is ready to be shelved off-site.
/eof
echo `date` $* done >>  /var/opt/SUNWsamfs/recycler.sh.log
exit 0
```

```
# The default action is to mail a message reminding you to set up this
```

7. Once you have edited the script to handle recycled cartridges, comment out the lines that generate the default reminder message.

```
# The default action is to mail a message reminding you to set up this
# file.  You should comment out these lines (through and including the /eof
# below) after you've set up this file.
#/bin/ppriv -s I=basic -e /usr/bin/mailx -s "Robot $6 ... recycle." root <</eof
#The /etc/opt/SUNWsamfs/scripts/recycler.sh script was called by the Oracle HSM
#recycler with the following arguments:
#
#      Media type: $5($1)  VSN: $2  Slot: $3  Eq: $4
#      Library: $6
#
#/etc/opt/SUNWsamfs/scripts/recycler.sh is a script which is called when the
#recycler determines that a VSN has been drained of all known active archive
...
#/eof
##echo `date` $* done >>  /var/opt/SUNWsamfs/recycler.sh.log
exit 0
```

8. When you are finished editing the file, save your changes and close the editor.

```
#/eof
##echo `date` $* done >>  /var/opt/SUNWsamfs/recycler.sh.log
exit 0
:wq
root@mds1:~#
```

9. If the **mcf** file for the archiving Oracle HSM file system includes a network-attached tape library in the archiving equipment section, catalog the archival media in the library.

10. Otherwise, backup your configuration and configure file-system protection.

Catalog Archival Media Stored in a Network-Attached Tape Library

After you mount a file system, the Oracle HSM software creates catalogs for each automated library that is configured in the **mcf** file. However, if you have a network-attached library, you have to take some additional steps to populate the its catalog.

Proceed as follows:

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

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

2. If the archiving file system uses an Oracle StorageTek ACSLS-attached tape library, draw the required Oracle HSM archival media from the library's scratch pool and generate the catalog automatically. Use the command **samimport -c** *volumes* **-s** *pool*, where:

   - *volumes* is the number of volumes needed.
   - *pool* is the name of the scratch media pool defined for the library

   In the example, we request **20** tape volumes drawn from the pool called **scratch**:

```
root@mds1:~# samimport -c 20 -s scratch
```

3. Once **samimport** has cataloged the media in an Oracle StorageTek ACSLS-attached tape library, you are ready to configure file system protection.

4. If the archiving file system uses an IBM 3494 library configured as a single, unshared logical library, place the required tape volumes in the library mail slot, and let the library catalog them automatically.

   If the **Additional Parameters** field of the **mcf** file specifies **access=private**, is configured as a single logical library.

5. Once an IBM 3494 library has automatically cataloged the media, you are ready to configure file system protection.

6. Otherwise, if the archiving file system uses either of the following, create a catalog input file using a text editor:

   - an IBM 3494 library configured as a shared library

     If the **Additional Parameters** field of the **mcf** file specifies **access=shared**, the IBM 3494 library is divided into multiple logical libraries.

   - any other network-attached library.

   In the example, we use the **vi** editor to create a catalog input file, **input3494cat**, for a shared IBM 3494 library:

```
root@mds1:~# vi input3494cat
~
"~/input3494cat" [New File]
```

7. Start a record by entering the record **index**. Always enter **0** (zero) for the first record, then increment the index for each succeeding record. Enter a space to indicate the end of the field.

   Rows define records and spaces delimit fields in **build_cat** input files. The value of the first field, the **index**, is simply a consecutive integer starting from **0** that identifies the record within the Oracle HSM catalog. In the example, this is the first record, so we enter **0**:

```
0
~
"~/input3494cat" [New File]
```

8. In the second field of the record, enter the volume serial number (VSN) of the tape volume or, if there is no VSN, a single **?** (question mark). Then enter a space to indicate the end of the field.

   Enclose values that contain white-space characters (if any) in double quotation marks: **"VOL 0A"**. In this example, the VSN of the first volume does not contain spaces:

```
0 VOL001
~
"~/input3494" [New File]
```

9. In the third field, enter the barcode of the volume (if different from the volume serial number), the volume serial number, or, if there is no volume serial number, the string **NO_BAR_CODE**. Then enter a space to indicate the end of the field.

   In the example, the barcode of the first volume has the same value as the VSN:

```
0 VOL001 VOL001
~
"~/input3494cat" [New File]
```

10. Finally, in the fourth field, enter the media type of the volume. Then enter a space to indicate the end of the field.

    The media type is a two-letter code, such as **`li`** for LTO media (see Appendix A, "Glossary of Equipment Types", for a comprehensive listing of media equipment types). In the example, we are using an IBM 3494 network-attached tape library with LTO tape drives, so we enter **`li `** (including the terminating space):

    ```
    0 VOL001 VOL001 li
    ~
    "~/input3494cat" [New File]
    ```

11. Repeat steps 3-6 to create additional records for each of the volumes that you intend to use with Oracle HSM. Then save the file.

    ```
    0 VOL001 VOL001 li
    1 VOL002 VOL002 li
    ...
    13 VOL014 VOL014 li
    :wq
    root@mds1:~#
    ```

12. Create the catalog with the **`build_cat`** *`input-file catalog-file`* command, where *`input-file`* is the name of your input file and *`catalog-file`* is the full path to the library catalog.

    If you have specified a catalog name in the **`Additional Parameters`** field of the **`mcf`** file, use that name. Otherwise, if you do not create catalogs, the Oracle HSM software creates default catalogs in the **`/var/opt/SUNWsamfs/catalog/`** directory using the file name *`family-set-name`*, where *`family-set-name`* is equipment name that you use for the library in **`mcf`** file. In the example, we use the family set **`i3494`**:

    ```
    root@mds1:~# build_cat input_vsns /var/opt/SUNWsamfs/catalog/i3494
    ```

13. If the archiving file system is shared, repeat the preceding step on each potential metadata server.

    The archiving file system is now complete and ready for use.

14. Next, configure file system protection.

Configure File System Protection

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 as well. When you back up the metadata, you back up directory paths, inodes, access controls, symbolic links, and the pointers that tie files to archival copies stored on removable media. So, to recover the file system, you simply have to restore the metadata. When a user subsequently requests a path and file, the file system will use the metadata to find the archive copy and automatically stage the corresponding data to disk.

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 inactive. 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.

File system protection thus requires some planning. On the one hand, you want to create recovery points and log-file copies frequently enough and retain them long enough to give you the best chance of recovering lost or damaged files and file systems. On the other hand, you do not want to create recovery points and log-file copies while data files are actively changing and you need to be cognizant of the disk space that they consume (recovery point files and logs can be large). Accordingly, this section recommends a broadly applicable configuration that can be used with many file system configurations without modification. When changes are necessary, the recommended configuration illustrates the issues and serves as a good starting point.

To create and manage recovery points, carry out the following tasks:

- Create locations for storing recovery point files and copies of the archiver log.
- Automatically create recovery points and save archiver logs.

Create Locations for Storing Recovery Point Files and Copies of the Archiver Log

For each archiving file system that you have configured, proceed as follows:

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

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

2. Select a storage location for the recovery point files. Select an independent file system that can be mounted on the file system host.

3. Make sure that the selected file system has enough space to store both new recovery point files and the number of recovery point files that you plan to retain at any given time.

   Recovery point files can be large and you will have to store a number of them, depending on how often you create them and how long you retain them.

4. Make sure that the selected file system does not share any physical devices with the archiving file system.

   Do not store recovery point files in the file system that they are meant to protect. Do not store recovery point files on logical devices, such as partitions or LUNs, that reside on physical devices that also host the archiving file-system.

5. In the selected file system, create a directory to hold recovery point files. Use the command **`mkdir`** *`mount-point`***`/`***`path`*, where *`mount-point`* is the mount point for the selected independent file system and *`path`* is the path and name of the chosen directory.

   Do not store recovery point files for several archiving file systems in a single, catch-all directory. Create a separate directory for each, so that recovery point files are organized and easily located when needed.

   In the example, we are configuring recovery points for the archiving file system **`/hqfs1`**. So we have created the directory **`/zfs1/hqfs1_recovery`** on the independent file system **`/zfs1`**:

   ```
   root@mds1:~# mkdir /zfs1/hqfs1_recovery
   ```

6. If a file system does not share any physical devices with the archiving file system, create a subdirectory for storing point-in-time copies of the archiver log(s) for your file system(s).

   In the example, we choose to store log copies in the **`/var`** directory of the host's root file system. We are configuring file system protection for the archiving file system **`/hqfs1`**. So we create the directory **`/var/hqfs1_archlogs`**:

   ```
   root@mds1:~# mkdir /var/hqfs1_archlogs
   ```

7. Next, automate creation of recovery points and saving of archiver logs.

Automatically Create Recovery Points and Save Archiver Logs

While you can create metadata recovery point files automatically, either by creating entries in the **`crontab`** file or by using the scheduling feature of the Oracle HSM Manager graphical user interface, the latter method does not automatically save archiver log data. So this section focuses on the **`crontab`** approach. If you wish to use the graphical user interface to schedule recovery points, refer to the Manager online help.

The procedure below creates two **`crontab`** entries that run daily: one that deletes out-of-date recovery point files and then creates a new recovery point and one that saves the archiver log. For each archiving file system that you have configured, proceed as follows:

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

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

2. Open the **`root`** user's **`crontab`** file for editing. Use the command **`crontab -e`**.

The **crontab** command opens an editable copy of the **root** user's **crontab** file in the text editor specified by the **EDITOR** environment variable (for full details, see the Solaris **crontab** man page). In the example, we use the **vi** editor:

```
root@mds1:~# crontab -e
...
# The root crontab should be used to perform accounting data collection.
10 3 * * * /usr/sbin/logadm
15 3 * * 0 [ -x /usr/lib/fs/nfs/nfsfind ] && /usr/lib/fs/nfs/nfsfind
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh
```

3. First, create the entry that deletes out-of-date recovery point files and creates a new recovery point. On a new line, specify the time of day when the work will be done. Enter *minutes hour* *** * ***, where:

- *minutes* is an integer in the range [**0-59**] that species the minute when the job starts.
- *hour* is an integer in the range [**0-23**] that species the hour when the job starts.
- ***** (asterisk) specifies unused values.

  For a task that runs daily, the values for day of the month [**1-31**], month [**1-12**], and day of the week [**0-6**] are unused.
- Spaces separate the fields in the time specification.
- *minutes hour* specify a time when files are not being created or modified.

  Creating a recovery point file when file-system activity is minimal insures that the file reflects the state of the archive as accurately and completely as possible. Ideally, all new and altered files will have been archived before the time you specify.

In the example, we schedule work to begin at 2:10 AM every day:

```
...
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh
10 2 * * *
```

4. Continuing on the same line, enter the shell commands that clean up the old recovery point files. Enter the text **(find** *directory* **-type f -mtime +***retention* **-print | xargs -l1 rm -f;**, where:

- **(** (opening parenthesis) marks the start of the command sequence that the **crontab** entry will execute.
- *directory* is the path and directory name of the directory where recovery point files are stored and thus the point where we want the Solaris **find** command to start its search.
- **-type f** is the **find** command option that specifies plain files (as opposed to block special files, character special files, directories, pipes, etc).
- **-mtime +***retention* is the **find** command option that specifies files that have not been modified for more than *retention*, an integer representing the number of hours that recovery point files are retained.
- **-print** is the **find** command option that lists all files found to standard output.

- **|xargs -l1 rm -f** pipes the output from **-print** to the Solaris command **xargs -l1**, which sends one line at a time as arguments to the Solaris command **rm -f**, which in turn deletes each file found.
- **;** (semicolon) marks the end of the command line.

In the example, the **crontab** entry searches the directory **/zfs1/hqfs1_recovery** for any files that have not been modified for 72 hours (3 days) or more and deletes any it finds. Note that the **crontab** entry continues on the same line but wraps around the display area:

```
# The root crontab should be used to perform accounting data collection.
10 3 * * * /usr/sbin/logadm
15 3 * * 0 [ -x /usr/lib/fs/nfs/nfsfind ] && /usr/lib/fs/nfs/nfsfind
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh
10 2 * * * ( find /zfs1/hqfs1_recovery -type f -mtime +72 -print | xargs -l1 rm
-f;
```

5. Continuing on the same line, enter the shell command that changes to the directory where the recovery point is to be created. Enter the text **cd** *mount-point;*, where *mount-point* is the root directory of the archiving file system and the semicolon (;) marks the end of the command line.

   The command that creates recovery point files, **samfsdump**, backs up the metadata for all files in the current directory and in all subdirectories. In the example, we change to the **/hqfs1** directory, the mount point for the file system that we are protecting. Note that the **crontab** entry continues on the same line but wraps around the display area:

```
# The root crontab should be used to perform accounting data collection.
10 3 * * * /usr/sbin/logadm
15 3 * * 0 [ -x /usr/lib/fs/nfs/nfsfind ] && /usr/lib/fs/nfs/nfsfind
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh
10 2 * * * ( find /zfs1/hqfs1_recovery -type f -mtime +72 -print | xargs -l1 rm
-f; cd /hqfs1;
```

6. Continuing on the same line, enter the shell commands that create the new daily recovery point. Enter the text **/opt/SUNWsamfs/sbin/samfsdump -f** *directory*/**'date +\%y\%m\%d')**, where:

   - **/opt/SUNWsamfs/sbin/samfsdump** is the command that creates recovery points (see the man page for full details).
   - **-f** is the **samfsdump** command option that specifies the location where the recovery point file will be saved.
   - *directory* is the directory that we created to hold recovery points for this file system.
   - **'date +\%y\%m\%d'** is the Solaris **date** command plus a formatting template that creates a name for the recovery point file: *YYMMDD*, where *YYMMDD* is the last two digits of the current year, the two-digit number of the current month, and the two-digit day of the month (for example, **150122**, January 22, 2015).
   - ; (semicolon) marks the end of the command line.
   - **)** (closing parenthesis) marks the end of the command sequence that the **crontab** entry will execute.

In the example, we specify the recovery-point directory that we created above, **/zfs1/hqfs1_recovery**. Note that the **crontab** entry continues on the same line but wraps around the display area:

```
# The root crontab should be used to perform accounting data collection.
10 3 * * * /usr/sbin/logadm
15 3 * * 0 [ -x /usr/lib/fs/nfs/nfsfind ] && /usr/lib/fs/nfs/nfsfind
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh
10 2 * * * ( find /zfs1/hqfs1_recovery -type f -mtime +72 -print | xargs -l1 rm
-f; cd /hqfs1 ; /opt/SUNWsamfs/sbin/samfsdump -f /zfs1/hqfs1_recovery/'date
+\%y\%m\%d')
```

7. Now create the entry that saves the archiver log. On a new line, specify the time of day when the work will be done by entering *minutes hour* *** * ***, where:

   - *minutes* is an integer in the range [**0-59**] that species the minute when the job starts.
   - *hour* is an integer in the range [**0-23**] that species the hour when the job starts.
   - ***** (asterisk) specifies unused values.

     For a task that runs daily, the values for day of the month [**1-31**], month [**1-12**], and day of the week [**0-6**] are unused.
   - Spaces separate the fields in the time specification.
   - *minutes hour* specify a time when files are not being created or modified.

   In the example, we schedule work to begin at 3:15 AM every Sunday:

```
...
30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh
10 2 * * * ( find /zfs1/hqfs1_recovery -type f -mtime +72 -print | \
xargs -l1 rm -f; cd /hqfs1 ; /opt/SUNWsamfs/sbin/samfsdump \
-f /zfs1/hqfs1_recovery/'date +\%y\%m\%d')
15 3 * * 0
```

8. Continuing on the same line, enter a shell command that moves the current archiver log to a backup location and gives it a unique name. Enter the text **(mv /var/adm/hqfs1.archive.log /var/hqfs1_archlogs/"date +%y%m%d";**.

   This step saves log entries that would be overwritten if left in the active log file. In the example, we move the archiver log for the **hqfs1** file system to our chosen location, **/var/hqfs1_archlogs/**, and rename it *YYMMDD*, where *YYMMDD* is the last two digits of the current year, the two-digit number of the current month, and the two-digit day of the month (for example, **150122**, January 22, 2015):

```
...
30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh
10 2 * * * ( find /hqfs1_recovery/dumps -type f -mtime +72 -print | xargs -l1
rm -f; \ cd /hqfs1 ; /opt/SUNWsamfs/sbin/samfsdump -f /zfs1/hqfs1_
recovery/'date +\%y\%m\%d')
15 3 * * 0 ( mv /var/adm/hqfs1.archiver.log /var/hqfs1_archlogs/"date +%y%m%d";
```

9. Continuing on the same line, enter a shell command to reinitialize the archiver log file. Enter the text **touch /var/adm/hqfs1.archive.log)**.

   In the example, note that the **crontab** entry continues on the same line but wraps around the display area:

```
...
30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh
```

```
10 2 * * * ( find /hqfs1_recovery/dumps -type f -mtime +72 -print | xargs -l1
rm -f; \ cd /hqfs1 ; /opt/SUNWsamfs/sbin/samfsdump -f /zfs1/hqfs1_
recovery/'date +\%y\%m\%d')
15 3 * * 0 ( mv /var/adm/hqfs1.archive.log /var/hqfs1_archlogs/"date +%y%m%d";
touch /var/adm/hqfs1.archiver.log )
```

10. Save the file, and close the editor.

```
# The root crontab should be used to perform accounting data collection.
10 3 * * * /usr/sbin/logadm
15 3 * * 0 [ -x /usr/lib/fs/nfs/nfsfind ] && /usr/lib/fs/nfs/nfsfind
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh
10 2 * * * ( find /hqfs1_recovery/dumps -type f -mtime +72 -print | xargs -l1
rm -f; \ cd /hqfs1 ; /opt/SUNWsamfs/sbin/samfsdump -f /zfs1/hqfs1_
recovery/'date +\%y\%m\%d')
15 3 * * 0 ( mv /var/adm/hqfs1.archive.log /var/hqfs1_archlogs/"date +%y%m%d";
touch /var/adm/hqfs1.archive.log )
:wq
root@mds1:~#
```

11. If you need to enable WORM (Write Once Read Many) capability on the file system, see "Enabling Support for Write Once Read Many (WORM) Files" on page 6-65.
12. If you need to interwork with systems that use LTFS or if you need to transfer large quantities of data between remote sites, see "Enabling Support for the Linear Tape File System (LTFS)" on page 6-67.
13. If you need to be able to verify the data integrity of archival tape volumes, configure archival media validation.
14. If you have additional requirements, such as multiple-host file-system access or high-availability configurations, see "Beyond the Basics" on page 6-70.
15. Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

Configure Archival Media Validation

Media validation is a technique that evaluates the data integrity of tape media using SCSI **verify** commands. The SCSI driver on the host calculates a CRC checksum for the logical blocks of data that it writes to the drive and sends a **verify** command. The drive reads the data blocks, calculates its own checksum, and compares the result with the value supplied by the driver. It returns an error if there is a discrepancy. The drive discards the data it reads as soon as the checksum is complete, so there is no additional I/O-related overhead on the host.

Oracle HSM supports media validation in two ways:

- You can configure Oracle HSM to support Data Integrity Validation (DIV) to validate data on StorageTek T10000 tape media, either manually or automatically under Oracle HSM Periodic Media Verification.
- You can also configure Oracle HSM Periodic Media Verification to automatically validate data on both StorageTek T10000 tape media and other formats, such as LTO Ultrium.

Configure Oracle HSM to Support Data Integrity Validation (DIV)

Data Integrity Validation (DIV) is a feature of Oracle StorageTek tape drives that works with the Oracle HSM software to insure the integrity of stored data. When the feature is enabled (**`div = on`** or **`div = verify`**), both the server host and the drive calculate and compare checksums during I/O. During write operations, the server 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 the server. If the values agree, the drive writes both the data block and the checksum to tape. During read operations, both the drive and the host 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 the application software that an error has occurred.

The **`div = verify`** option provides an additional layer of protection when writing data. When the write operation is complete, the host asks the tape drive to reverify the data. The drive then rescans the data, recalculates checksums, and compares the results to the checksums stored on the tape. The drive performs all operations internally, with no additional I/O (data is discarded), so there is no additional overhead on the host system. You can also use the Oracle HSM **`tpverify`** (tape-verify) command to perform this step on demand.

To configure Data Integrity Validation, proceed as follows:

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

   In the example, the metadata server is named **`samfs-mds`**:

   ```
   root@samfs-mds:~#
   ```

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

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

3. Make sure that the archival storage equipment defined in the Oracle HSM **`mcf`** file includes compatible tape drives: StorageTek T10000C (minimum firmware level 1.53.315) or T10000D.

4. Idle all archiving processes, if any. Use the command **`samcmd aridle`**.

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

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

5. Idle all staging processes, if any. Use the command **`samcmd stidle`**.

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

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

6. Wait for any 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@samfs-mds:~# samcmd a
   Archiver status samcmd     6.0 14:20:34 Feb 22 2015
   samcmd on samfs-mds
   sam-archiverd:  Waiting for :arrun
   sam-arfind: ...
   ```

```
Waiting for :arrun
```

7. Wait for any 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@samfs-mds:~# samcmd u
   Staging queue samcmd      6.0 14:20:34 Feb 22 2015
   samcmd on solaris.demo.lan
   Staging queue by media type: all
   sam-stagerd:  Waiting for :strun
   root@mds1:~#
   ```

8. 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@samfs-mds:~# samcmd 801 idle
   root@samfs-mds:~# samcmd 802 idle
   root@samfs-mds:~# samcmd 803 idle
   root@samfs-mds:~# samcmd 804 idle
   root@samfs-mds:~#
   ```

9. 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@samfs-mds:~# samcmd r
   Removable media samcmd     6.0 14:20:34 Feb 22 2015
   samcmd on samqfs1host
   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@samfs-mds:~#
   ```

10. 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@samfs-mds:~# samd stop
    root@samfs-mds:~#
    ```

11. Open the **`/etc/opt/SUNWsamfs/defaults.conf`** file in a text editor. Uncomment the line **`#div = off`**, if necessary, or add it if it is not present.

    By default, **`div`** (Data Integrity Validation) is **`off`** (disabled).

    In the example, we open the file in the **`vi`** editor and uncomment the line:

    ```
    root@samfs-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)
# and change the value.
...
div = off
```

12. To enable Data Integrity Validation read, write, and verify operations, change the line **`#div = off`** to **`div = on`**, and save the file.

    Data will be verified as each block is written and read, but the Oracle HSM archiver software will not verify complete file copies after they are archived.

```
root@samfs-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)
# and change the value.
...
div = on
:wq
root@samfs-mds:~#
```

13. To enable the verify-after-write option of the Data Integrity Validation feature, change the line **`#div = off`** to **`div = verify`**, and save the file.

    The host and the drive carry out Data Integrity Validation as each block is written or read. In addition, whenever a complete archive request is written out to tape, the drive re-reads the newly stored data and checksums, recalculates, and, compares the stored and calculated results.

```
root@samfs-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)
# and change the value.
...
div = verify
:wq
root@samfs-mds:~#
```

14. Tell the Oracle HSM software to re-read the **`defaults.conf`** file and reconfigure itself accordingly. Use the **`samd config`** command.

```
root@samfs-mds:~# /opt/SUNWsamfs/sbin/samd config
```

15. If you stopped Oracle HSM operations in an earlier step, restart them now using the **`samd start`** command.

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

    Data Integrity Validation is now configured.

16. If you need to automate data integrity validation, go to "Configure Oracle HSM Periodic Media Verification" on page 6-60.
17. If you need to enable WORM (Write Once Read Many) capability on the file system, see "Enabling Support for Write Once Read Many (WORM) Files" on page 6-65.
18. If you need to interwork with systems that use LTFS or if you need to transfer large quantities of data between remote sites, see "Enabling Support for the Linear Tape File System (LTFS)" on page 6-67.
19. If you have additional requirements, such as multiple-host file-system access or high-availability configurations, see "Beyond the Basics" on page 6-70.

Configure Oracle HSM Periodic Media Verification

You can set up Periodic Media Verification (PMV) for Oracle HSM archiving file systems. Periodic Media Verification automatically checks the data integrity of the removable media in a file system. It checks StorageTek T10000 media using StorageTek Data Integrity Validation and other drives using the widely supported SCSI **`verify(6)`** command.

The Periodic Media Verification feature adds an Oracle HSM daemon, **`verifyd`**, that periodically applies the **`tpverify`** command, logs any errors detected, notifies administrators, and automatically performs specified recovery actions. You configure Periodic Media Verification by setting policy directives in a configuration file, **`verifyd.cmd`**. Policies can specify the times when verification scans are run, the types of scan done, the libraries and drives that can be used, the tape volumes that should be scanned, and the actions that Oracle HSM takes when errors are detected. Oracle HSM can, for example, automatically re-archive files that contain errors and/or recycle tape volumes that contain errors.

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

   In the example, the metadata server is named **`samfs-mds`**:

   ```
   root@samfs-mds:~#
   ```

2. If you have not already done so, configure Oracle HSM to support Data Integrity Validation (DIV) before proceeding.

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

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

4. Open the **`/etc/opt/SUNWsamfs/verifyd.cmd`** file in a text editor.

   In the example, we open the file in the **`vi`** editor:

   ```
   root@samfs-mds:~# vi /etc/opt/SUNWsamfs/verifyd.cmd
   # For additional information about the format of the verifyd.cmd file,
   # type "man verifyd.cmd".
   # Enable Oracle HSM Periodic Media Validation (PMV)
   pmv = off
   ```

5. To enable Periodic Media Verification, enter the line **`pmv = on`**.

   By default, Periodic Media Verification is **`off`**. In the example, we set it **`on`**:

   ```
   root@samfs-mds:~# vi /etc/opt/SUNWsamfs/verifyd.cmd
   # For additional information about the format of the verifyd.cmd file,
   # type "man verifyd.cmd".
   # Enable Oracle HSM Periodic Media Validation (PMV)
   pmv = on
   ```

6. Set a run time. Enter the line **`run_time = always`** to run verification continuously or **`run_time =`** *`HHMM hhmm DD dd`*, where *`HHMM`* and *`hhmm`* are, respectively, starting and ending times and where *`DD dd`* are an optional starting and ending day.

   `HH` and *`hh`* are hours of the day in the range `00-24`, *`MM`* and *`mm`* are numbers of minutes in the range `00-60`, and *`DD`* and *`dd`* are days of the week in the range `[0-6]`, where `0` is Sunday and `6` is Saturday. The default is `2200 0500 6 0`.

   But verification will not compete with more immediately important file system operations. The verification process automatically yields tape volumes and/or drives that are required by the archiver and stager. So, in the example, we set the run time to **always**:

```
root@samfs-mds:~# vi /etc/opt/SUNWsamfs/verifyd.cmd
# For additional information about the format of the verifyd.cmd file,
# type "man verifyd.cmd".
# Enable Oracle HSM Periodic Media Validation (PMV)
pmv = on
# Run all of the time.  PMV will yield VSNs and drives when
# resources are wanted by the SAM-QFS archiver and stager.
run_time = always
```

7. Specify a verification method. Enter the line **pmv_method =** *specified-method* where *specified-method* is one of the following:

   - The **standard** method is specifically for use with Oracle StorageTek T10000C and later tape drives. Optimized for speed, the **standard** method verifies the edges, beginning, end, and first 1,000 blocks of the media.
   - The **complete** method is also for use with Oracle StorageTek T10000C and later tape drives. It verifies the media error correction code (ECC) for every block on the media.
   - The **complete plus** is also for use with Oracle StorageTek T10000C and later tape drives. It verifies both the media error correction code (ECC) and the Data Integrity Validation checksum for each block on media (see "Configure Oracle HSM to Support Data Integrity Validation (DIV)" on page 6-56).
   - The **legacy** method can be used with all other tape drives and is used automatically when media is marked bad in the catalog and when drives do not support the method specified in the **verifyd.cmd** file. It runs a 6-byte, fixed-block mode SCSI Verify Command, skipping previously logged defects. When a new permanent media error is found, the **legacy** method skips to the next file and logs the newly discovered error in the media defects database.
   - The **mir rebuild** method rebuilds the media information region (MIR) of an Oracle StorageTek tape cartridge if the MIR is missing or damaged. It works with media that is marked bad in the media catalog and is automatically specified when MIR damage is detected.

   In the example, we are using LTO drives, so we specify **legacy**:

```
root@samfs-mds:~# vi /etc/opt/SUNWsamfs/verifyd.cmd
...
# resources are wanted by the SAM-QFS archiver and stager.
run_time = always
pmv_method = legacy
```

8. To use all available libraries and drives for verification, enter the line **pmv_scan = all**.

```
root@samfs-mds:~# vi /etc/opt/SUNWsamfs/verifyd.cmd
...
pmv_method = legacy
pmv_scan = all
```

9. To use all available drives in a specified library for verification, enter the line **pmv_scan = library** *equipment-number*, where *equipment-number* is the equipment number assigned to the library in the file system's **mcf** file.

   In the example, we let the verification process use all drives in library **800**.

```
root@samfs-mds:~# vi /etc/opt/SUNWsamfs/verifyd.cmd
...
pmv_method = legacy
pmv_scan = library 800
```

10. To limit the number of drives that the verification process can use in a specified library, enter the line **`pmv_scan = library`** *`equipment-number`* **`max_drives`** *`number`*, where *`equipment-number`* is the equipment number assigned to the library in the file system's **`mcf`** file and number is the maximum number of drives that can be used.

    In the example, we let the verification process use at most **`2`** drives in library **`800`**:

```
root@samfs-mds:~# vi /etc/opt/SUNWsamfs/verifyd.cmd
...
pmv_method = legacy
pmv_scan = library 800 max_drives 2
```

11. To specify the drives that the verification process can use in a specified library, enter the line **`pmv_scan = library`** *`equipment-number`* **`drive`** *`drive-numbers`*, where *`equipment-number`* is the equipment number assigned to the library in the file system's **`mcf`** file and *`drive-numbers`* is a space-delimited list of the equipment numbers assigned to the specified drives in the **`mcf`** file.

    In the example, we let the verification process use drives **`903`** and **`904`** in library **`900`**:

```
root@samfs-mds:~# vi /etc/opt/SUNWsamfs/verifyd.cmd
...
pmv_method = legacy
pmv_scan = library 900 drive 903 904
```

12. To specify the drives that the verification process can use in two or more libraries, enter the line **`pmv_scan =`** *`library-specification library-specification`***`...`**, where *`equipment-number`* is the equipment number assigned to the library in the file system's **`mcf`** file and *`drive-numbers`* is a space-delimited list of the equipment numbers assigned to the specified in the **`mcf`** file.

    In the example, we let the verification process use at most **`2`** drives in library **`800`** and drives **`903`** and **`904`** in library **`900`**:

```
root@samfs-mds:~# vi /etc/opt/SUNWsamfs/verifyd.cmd
...
pmv_method = legacy
pmv_scan = library 800 max_drives 2 library 900 drive 903 904
```

13. To disable periodic media verification and prevent it from using any equipment, enter the line **`pmv_scan = off`**.

```
root@samfs-mds:~# vi /etc/opt/SUNWsamfs/verifyd.cmd
...
pmv_method = legacy
pmv_scan = off
```

14. To automatically flag the media for recycling once periodic media verification has detected a specified number of permanent errors, enter the line **`action = recycle perms`** *`number-errors`*, where *`number-errors`* is the number of errors.

    In the example, we configure Oracle HSM to flag the media for recycling after **`10`** errors have been detected:

```
root@samfs-mds:~# vi /etc/opt/SUNWsamfs/verifyd.cmd
```

```
...
pmv_scan = all
action = recycle perms 10
```

15. To automatically re-archive files that contain bad blocks after errors have accumulated for a specified period, enter the line **`action = rearch age`** *`time`*, where *`time`* is a space-delimited list of any combination of *`SECONDS`***`s`**, *`MINUTES`***`m`**, *`HOURS`***`h`**, *`DAYS`***`d`**, and/or *`YEARS`***`y`** and *`SECONDS`*, *`MINUTES`*, *`HOURS`*, *`DAYS`*, and *`YEARS`* are integers.

    The oldest media defect must have aged for the specified period before the file system is scanned for files that need archiving. In the example, we set the re-archiving age to **`1`** (one) minute:

    ```
    root@samfs-mds:~# vi /etc/opt/SUNWsamfs/verifyd.cmd
    ...
    pmv_scan = all
    action = rearch age 1m
    ```

16. To mark the media as bad when periodic media verification detects a permanent media error and take no action otherwise, enter the line **`action = none`**.

    ```
    root@samfs-mds:~# vi /etc/opt/SUNWsamfs/verifyd.cmd
    ...
    pmv_scan = all
    action = none
    ```

17. Specify the tape volumes that should be verified periodically. Enter the line **`pmv_vsns =`** *`selection-criterion`*, where *`selection-criterion`* is **`all`** or a space-delimited list of regular expressions that specify one or more volume serial numbers (VSNs).

    The default is **`all`**. In the example, we supply three regular expressions: **`^VOL0[01][0-9]`** and **`^VOL23[0-9]`** specify two sets volumes with volume serial numbers in the ranges **`VOL000`** to **`VOL019`** and **`VOL230`** to **`VOL239`**, respectively, while **`VOL400`** specifies the volume with that specific volume serial number:

    ```
    root@samfs-mds:~# vi /etc/opt/SUNWsamfs/verifyd.cmd
    ...
    pmv_scan = all
    action = none
    pmv_vsns = ^VOL0[01][0-9] ^VOL23[0-9] VOL400
    ```

    Oracle HSM will not try to verify volumes if they need to be audited, if they are scheduled for recycling, if they are unavailable, if they are foreign (non-Oracle HSM) volumes, or if they do not contain data. Cleaning cartridges, volumes that are unlabeled, and volumes that have duplicate volume serial numbers are also excluded.

18. Define the desired verification policy. Enter the line **`pmv_policy = verified age`** *`vertime`* **`[modified age`** *`modtime`***`]`** **`[mounted age`** *`mnttime`***`]`**, where :

    - **`verified age`** specifies the minimum time that must have passed since the volume was last verified.
    - **`modified age`** (optional) specifies the minimum time that must have passed since the volume was last modified.
    - **`mounted age`** (optional) specifies the minimum time that must have passed since the volume was last mounted.

- The parameter values *vertime*, *modtime*, and *mnttime* are combinations of non-negative integers and the following units of time: **y** (years), **m** (months), **d** (days), **H** (hours), **M** (minutes), and **S** (seconds).

Oracle HSM identifies and ranks candidates for verification based on the amount of time that has passed since the volume was last verified and, optionally, modified and/or mounted. The default policy is the single parameter, **verified age 6m** (six months). In the example, we set the last-verified age to three months and the last-modified age to fifteen months:

```
root@samfs-mds:~# vi /etc/opt/SUNWsamfs/verifyd.cmd
...
pmv_scan = all
action = none
pmv_vsns = ^VOL0[01][0-9] ^VOL23[0-9] VOL400
pmv_policy = verified age 3m modified age 15m
```

19. Save the **/etc/opt/SUNWsamfs/verifyd.cmd** file, and close the editor.

```
root@samfs-mds:~# vi /etc/opt/SUNWsamfs/verifyd.cmd
...
pmv_vsns = ^VOL0[01][0-9] ^VOL23[0-9] VOL400
pmv_policy = verified age 3m modified age 15m
:wq
root@mds1:~#
```

20. Check the **verifyd.cmd** file for errors by entering the **tpverify -x** command. Correct any errors found.

The **tpverify -x** command reads the **verifyd.cmd** and stops if it encounters an error:

```
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: legacy
STA Scan: off
Action: none
PMV VSNs: all
PMV Policy:
     Last Verified Age: 6m
root@mds1:~#
```

21. Restart the verification service using the new **verifyd.cmd** file. Enter the command **tpverify -r** command.

```
root@mds1:~# tpverify -r
root@mds1:~#
```

You have finished configuring periodic media verification.

22. If you need to enable WORM (Write Once Read Many) capability on the file system, see "Enabling Support for Write Once Read Many (WORM) Files" on page 6-65.

23. If you need to interwork with systems that use LTFS or if you need to transfer large quantities of data between remote sites, see "Enabling Support for the Linear Tape File System (LTFS)" on page 6-67.

24. If you have additional requirements, such as multiple-host file-system access or high-availability configurations, see "Beyond the Basics" on page 6-70.
25. Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

Enabling Support for Write Once Read Many (WORM) Files

Write-once read-many (WORM) files are used in many applications for legal and archival reasons. WORM-enabled Oracle HSM file systems support default and customizable file-retention periods, data and path immutability, and subdirectory inheritance of the WORM setting. You can use either of two WORM modes:

- standard compliance mode (the default)

  The standard WORM mode starts the WORM retention period when a user sets UNIX **`setuid`** permission on a directory or non-executable file (**`chmod 4000`** *`directory`*|*`file`*). Since setting **`setuid`** (**`set user ID upon execution`**) permission on an executable file presents security risks, files that also have UNIX execute permission cannot be retained using this mode.

- emulation mode

  The WORM emulation mode starts the WORM retention period when a user makes a writable file or directory read-only (**`chmod 444`** *`directory`*|*`file`*), so executable files can be retained.

Both standard and emulation modes have both a strict WORM implementation and a less restrictive, *lite* implementation that relaxes some restrictions for **`root`** users. Both strict and lite implementations do not allow changes to data or paths once retention has been triggered on a file or directory. The strict implementations do not let anyone shorten the specified retention period (by default, 43,200 minutes/30 days) 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 legal and regulatory compliance requirements. The lite implementations let **`root`** users shorten retention periods, delete files and directories, and delete volumes using the file-system creation command **`sammkfs`**. The lite implementations may thus be better choices when both data integrity and flexible management are primary requirements.

Take care when selecting a WORM implementation and when enabling retention on a file. In general, use the least restrictive option that is consistent with requirements. You cannot change from standard to emulation modes or vice versa. So choose carefully. If management flexibility is a priority or if retention requirements may change at a later date, select a lite implementation. You can upgrade from the lite version of a WORM mode to the strict version, should it later prove necessary. But you cannot change from a strict implementation to a lite implementation. Once a strict WORM implementation is in effect, files must be retained for their full specified retention periods. So set retention to the shortest value consistent with requirements.

Enable the WORM Support on an Oracle HSM File System

You enable WORM support on a file system using mount options. Proceed as follows.

1. Log in as **`root`**.

   ```
   root@solaris:~#
   ```

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

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

3. Open the **/etc/vfstab** file in a text editor and locate the entry for the Oracle HSM file system for which you want to enable WORM support.

   In the example, we open the **/etc/vfstab** file in the **vi** editor and locate the archiving file system **worm1**:

```
root@solaris:~# 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      -
...
worm1     -       /worm1   samfs  -    yes     -
```

4. To enable the strict implementation of the standard WORM compliance mode, enter the **worm_capable** option in the **Mount Options** column of the **vfstab** file.

```
#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      -
...
worm1     -       /worm1   samfs  -    yes     worm_capable
```

5. To enable the lite implementation of the standard WORM compliance mode, enter the **worm_lite** option in the **Mount Options** column of the **vfstab** file.

```
#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      -
...
worm1     -       /worm1   samfs  -    yes     worm_lite
```

6. To enable the strict implementation of the WORM emulation mode, enter the **worm_emul** option in the **Mount Options** column of the **vfstab** file.

```
#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      -
...
worm1     -       /worm1   samfs  -    yes     worm_emul
```

7. To enable the lite implementation of the WORM emulation mode, enter the **emul_lite** option in the **Mount Options** column of the **vfstab** file.

```
#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      -
```

```
...
worm1     -      /worm1    samfs  -    yes     emul_lite
```

8. To change the default retention period for files that are not explicitly assigned a retention period, add the **def_retention=***period* option to the **Mount Options** column of the **vfstab** file, where *period* takes one of the forms explained in the following paragraph.

   The value of *period* can take any of three forms:

   - **permanent** or **0** specifies permanent retention.
   - *YEARS***y***DAYS***d***HOURS***h***MINUTES***m** where *YEARS*, *DAYS*, *HOURS*, and *MINUTES* are non-negative integers and where specifiers may be omitted. So, for example, **5y3d1h4m**, **2y12h**, and **365d** are all valid.
   - *MINUTES* where *MINUTES* is an integer in the range **[1-2147483647]**.

   Set a default retention period if you must set retention periods that extend beyond the year 2038. UNIX utilities such as **touch** use signed, 32-bit integers to represent time as the number of seconds that have elapsed since January 1, 1970. The largest number of seconds that a 32-bit integer can represent translates to January 18, 2038 at 10:14 PM

   If a value is not supplied, **def_retention** defaults to **43200** minutes (30 days). In the example, we set the retention period for a standard WORM-capable file system to **777600** minutes (540 days):

```
#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      -
...
worm1     -       /worm1   samfs  -    no     worm_capable,def_retention=777600
```

9. Save the **vfstab** file, and close the editor.

   The file system is WORM-enabled. Once one or more WORM files are resident in the file system, the Oracle HSM software will update the file system superblock to reflect the WORM capability. Any subsequent attempt to rebuild the file system with **sammkfs** will fail if the file system has been mounted with the strict **worm_capable** or **worm_emul** mount option.

10. If you need to interwork with systems that use LTFS or if you need to transfer large quantities of data between remote sites, see "Enabling Support for the Linear Tape File System (LTFS)" on page 6-67

11. If you have additional requirements, such as multiple-host file-system access or high-availability configurations, see "Beyond the Basics" on page 6-70.

12. Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

Enabling Support for the Linear Tape File System (LTFS)

Oracle HSM can import data from and export data to Linear Tape File System (LTFS) volumes. This capability facilitates interworking with systems that use LTFS as their standard tape format. It also eases transfer of very large volumes of data between remote Oracle HSM sites, when typical wide-area network (WAN) connections are too slow or too expensive for the task.

Note that the Oracle HSM software supports but does not include LTFS functionality. To use LTFS file systems, the host's Solaris operating system must include the **SUNWltfs** package. If necessary, download and install the **SUNWltfs** package before proceeding further.

For information on using and administering LTFS volumes, see the **samltfs** man page and the *Oracle Hierarchical Storage Manager and StorageTek QFS Software Maintenance and Administration Guide*.

To enable Oracle HSM LTFS support, proceed as follows:

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

   ```
   root@samfs-mds:~#
   ```

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

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

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

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

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

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

4. Wait for any 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@samfs-mds:~# samcmd a
   Archiver status samcmd     6.0 14:20:34 Feb 22 2015
   samcmd on samfs-mds
   sam-archiverd:  Waiting for :arrun
   sam-arfind: ...
   Waiting for :arrun
   ```

5. Wait for any 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@samfs-mds:~# samcmd u
   Staging queue samcmd      6.0 14:20:34 Feb 22 2015
   samcmd on solaris.demo.lan
   Staging queue by media type: all
   sam-stagerd:  Waiting for :strun
   root@solaris:~#
   ```

6. 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@samfs-mds:~# samcmd 801 idle
root@samfs-mds:~# samcmd 802 idle
root@samfs-mds:~# samcmd 803 idle
root@samfs-mds:~# samcmd 804 idle
root@samfs-mds:~#
```

7. 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@samfs-mds:~# samcmd r
Removable media samcmd     6.0 14:20:34 Feb 22 2015
samcmd on samqfs1host
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@samfs-mds:~#
```

8. 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@samfs-mds:~# samd stop
root@samfs-mds:~#
```

9. Download and review the LTFS Open Edition (LTFS-OE) documentation for the current version of the software. Documents are available at **`https://oss.oracle.com/projects/ltfs/documentation/`**.

   At a minimum, review the **`README.txt`** file and the installation document for Solaris (the operating system used on the metadata server): **`INSTALL.solaris`**. Check the **`README.txt`** file for hardware compatibility information.

10. Download, install, and configure the LTFS-OE packages. Follow the instructions in the **`INSTALL`** document.

    Download packages from **`https://oss.oracle.com/projects/ltfs/files/`** or as directed in the **`INSTALL`** document.

11. Once LTFS-OE is installed, open the file **`/etc/opt/SUNWsamfs/defaults.conf`** in a text editor.

    In the example, we open the file in the **`vi`** editor:

```
root@samfs-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)
# and change the value.
...
```

12. In the **`defaults.conf`** file, add the line **`ltfs =`** *`mountpoint workers volumes`*, where:

    - *`mountpoint`* is the directory in the host file system where the LTFS file system should be mounted.
    - *`workers`* is an optional maximum number of drives to use for LTFS.

- *volumes* is an optional maximum number of tape volumes per drive.

In the example, we specify the LTFS mount point s **/mnt/ltfs** and accept the defaults for the other parameters:

```
root@samfs-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)
# and change the value.
...
ltfs = /mnt/ltfs
:wq
root@samfs-mds:~#
```

13. save the **defaults.conf** file, and close the editor.

```
...
ltfs = /mnt/ltfs
:wq
root@samfs-mds:~#
```

14. Tell the Oracle HSM software to reread the **defaults.conf** file and reconfigure itself accordingly. Correct any errors reported and repeat as necessary.

```
root@samfs-mds:~# /opt/SUNWsamfs/sbin/samd config
```

15. If you stopped Oracle HSM operations in an earlier step, restart them now using the **samd start** command.

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

16. Oracle HSM support for LTFS is now enabled. If you have additional requirements, such as multiple-host file-system access or high-availability configurations, see "Beyond the Basics" on page 6-70.
17. Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

Beyond the Basics

This completes basic installation and configuration of Oracle HSM file systems. At this point, you have set up fully functional file systems that are optimally configured for a wide range of purposes.

The remaining chapters in this book address more specialized needs. So, before you embark on the additional tuning and feature implementation tasks outlined below, carefully assess your requirements. Then, if you need additional capabilities, such as high-availability or shared file-system configurations, you can judiciously implement additional features starting from the basic configurations. But if you find that the work you have done so far can meet your needs, additional changes are unlikely to be an improvement. They may simply complicate maintenance and administration.

- If applications transfer unusually large or unusually uniform amounts of data to the file system, you may be able to improve file system performance by setting additional mount options. See "Tuning I/O Characteristics for Special Needs" on page 12-1 for details.
- If you need to configure shared access to the file system, see "Accessing File Systems from Multiple Hosts Using Oracle HSM Software" on page 8-1 and/or "Accessing File Systems from Multiple Hosts Using NFS and SMB/CIFS" on page 8-32.

- If you need to configure a high-availability QFS file system or Oracle HSM archiving file system, see "Preparing High-Availability Solutions" on page 9-1.
- If you need to configure an Oracle HSM archiving file system to share archival storage hosted at a remote location, see "Configuring SAM-Remote" on page 7-1.
- If you plan on using the sideband database feature, go to "Configuring the Reporting Database" on page 10-1.
- Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

7 Configuring SAM-Remote

The Oracle Hierarchical Storage Manager software's SAM-Remote feature lets Oracle HSM file-system hosts access tape media and drives that are hosted on a remote, Oracle HSM file-system host. The local host accesses tape resources as a SAM-Remote client of the remote host, which serves as a SAM-Remote server. The client's archiving policies typically maintain one or two copies in a local magnetic or solid state (SSD) disk archive and one or two copies on remote tapes provided by the server. The master configuration file on each host, **`/etc/opt/SUNWsamfs/mcf`**, defines the shared resources and the client/server relationships using special, SAM-Remote equipment types.

You can address a number of archiving and data-protection requirements SAM-Remote clients and servers:

- You can extend the advantages of tape archiving to Oracle HSM hosts that lack libraries and drives.
- You can centralize maintenance and management of tape resources for Oracle HSM file systems hosted in regional offices and satellite campuses.

  In the central, main-office datacenter, Oracle HSM file-system hosts have attached tape libraries and operate as SAM-Remote servers. In smaller, dispersed offices, Oracle HSM file-system hosts have only disk archives and function as SAM-Remote clients. All hosts maintain both local and tape copies of their archived data. But hardware and media inventories are concentrated in the central datacenter, where they can be maintained most efficiently and at least cost.
- You can automatically create and maintain off-site tape copies for backup and disaster-recovery purposes.

  All Oracle HSM file-system hosts have attached tape libraries. Each host operates both as a SAM-Remote client and as a server with respect to an opposite number in an off-site location. Each Oracle HSM host creates local disk and tape copies using local resources. Each host creates remote tape copies using resources provided by its counterpart, and each provides tape resources to its counterpart. So offsite copies of the two file systems are created automatically, as part of the normal archiving process.
- You can configure Oracle HSM file-system hosts to access remote archival storage resources when local resources are unavailable.

  Once again, all Oracle HSM file-system hosts have attached tape libraries, and each operates both as a SAM-Remote client and as a server with respect to an opposite number in another location. Each Oracle HSM host creates local disk and tape copies using local resources. But if a host cannot access its local library, it can still archive and retrieve files using media and resources provided by its remote counterpart.

This chapter outlines the process of configuring a SAM-Remote client/server network. It covers the following tasks:

- Make sure that all SAM-Remote hosts use the same software.
- Stop Oracle HSM processes.
- Configure the SAM-Remote server.
- Configure the SAM-Remote clients.
- Validate the archiving configuration on the SAM-Remote server.
- Validate the archiving configuration on each SAM-Remote client.

Make Sure that All SAM-Remote Hosts Use the Same Software

SAM-Remote clients and servers must have the same revision of the Oracle HSM software installed. Check the revision levels using the procedure below:

1. Log in to the SAM-Remote server host as **root**.

   In the example, the server host is **server1**:

   ```
   root@server1:~#
   ```

2. Log in to the SAM-Remote client hosts as **root**.

   In the example, we open a terminal window and use **ssh** to log in to the host **client1**:

   ```
   root@server1:~# ssh root@client1
   Password: ...
   root@client1:~#
   ```

3. Make sure that Oracle HSM package revision levels are identical on all SAM-Remote servers and clients. On each SAM-Remote host, use the command **samcmd l** to list configuration details. Compare the results.

   In the example, we compare the results on **server1** to those on **client1**. Both use the same release of the Oracle HSM software:

   ```
   root@server1:~# samcmd l
   Usage information samcmd     6.0  10:20:34 Feb 20 2015
   samcmd on server1
   ...
   root@server1:~#

   root@client1:~# samcmd l
   Usage information samcmd     6.0  10:20:37 Feb 20 2015
   samcmd on client1
   ...
   root@server1:~#
   ```

4. Using the procedures in Chapter 4, "Installing Oracle HSM and QFS Software", update host software as necessary until all SAM-Remote servers and clients are at the same revision levels.

5. Next, stop Oracle HSM processes.

Stop Oracle HSM Processes

1. Log in to the SAM-Remote server host as **root**.

In the example, the server is named **server1**:

```
root@server1:~#
```

2. Obtain the equipment ordinal numbers of the configured devices. Use the command **samcmd c**.

   In the example, the devices are numbered **801**, **802**, **803**, and **804**:

```
root@server1:~# samcmd c
Device configuration samcmd     6.0  10:20:34 Feb 20 2015
samcmd on server1
Device configuration:
ty  eq  state   device_name                       fs   family_set
rb  800 on      /dev/scsi/changer/c1t0d5          800  rb800
tp  801 on      /dev/rmt/0cbn                     801  rb800
tp  802 on      /dev/rmt/1cbn                     802  rb800
tp  803 on      /dev/rmt/2cbn                     803  rb800
tp  804 on      /dev/rmt/3cbn                     804  rb800
```

3.

4. Idle all archiving processes, if any. Use the command **samcmd aridle**.

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

```
[samfs-mds]root@solaris:~# samcmd aridle
[samfs-mds]root@solaris:~#
```

5. Idle all staging processes, if any. Use the command **samcmd stidle**.

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

```
[samfs-mds]root@solaris:~# samcmd stidle
[samfs-mds]root@solaris:~#
```

6. Wait for any 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:

```
[samfs-mds]root@solaris:~# samcmd a
Archiver status samcmd     6.0 14:20:34 Feb 22 2015
samcmd on samfs-mds
sam-archiverd:  Waiting for :arrun
sam-arfind: ...
Waiting for :arrun
```

7. Wait for any 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:

```
[samfs-mds]root@solaris:~# samcmd u
Staging queue samcmd      6.0 14:20:34 Feb 22 2015
samcmd on solaris.demo.lan
Staging queue by media type: all
sam-stagerd:  Waiting for :strun
root@solaris:~#
```

8. 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**:

```
[samfs-mds]root@solaris:~# samcmd 801 idle
[samfs-mds]root@solaris:~# samcmd 802 idle
[samfs-mds]root@solaris:~# samcmd 803 idle
[samfs-mds]root@solaris:~# samcmd 804 idle
[samfs-mds]root@solaris:~#
```

9. 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.

```
[samfs-mds]root@solaris:~# samcmd r
Removable media samcmd     6.0 14:20:34 Feb 22 2015
samcmd on samqfs1host
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
[samfs-mds]root@solaris:~#
```

10. 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**.

```
[samfs-mds]root@solaris:~# samd stop
[samfs-mds]root@solaris:~#
```

11. Next, configure the SAM-Remote server.

Configure the SAM-Remote Server

A SAM-Remote server is an Oracle HSM file-system host that makes its attached robotic tape libraries, and tape drives available to remote clients that are themselves Oracle HSM file-system hosts. The SAM-Remote server must mount at least one QFS file system to start Oracle HSM processes.

To configure a SAM-Remote server, carry out the following tasks:

- Define the remotely shared archiving equipment in the SAM-Remote server's **mcf** file.
- Create the **samremote** server configuration file.

Define the Remotely Shared Archiving Equipment in the SAM-Remote Server's `mcf` File

1. Log in to the SAM-Remote server host as **root**.

   In the example, the server is named **server1**:

```
root@server1:~#
```

2. On the server, open the **`/etc/opt/SUNWsamfs/mcf`** file in a text editor, and scroll down to the archiving equipment definitions.

   In the example, we use the **`vi`** editor. The file defines one Oracle HSM archiving file system, **`fs600`**, and a tape library, **`rb800`**, that holds four drives. Note that the example includes clarifying headings that may not be present in actual files and abbreviates lengthy device paths:

```
root@server1:~# vi /etc/opt/SUNWsamfs/mcf
#======================================================================
# Oracle HSM archiving file system fs600
# Equipment                Equipment Equipment Family  Device Additional
# Identifier               Ordinal   Type      Set     State  Parameters
#------------------------  --------- --------- ------  ------ ----------
fs600                      600       ms        fs600   on
 /dev/dsk/c9t60...F4d0s7    610       md        fs600   on
 /dev/dsk/c9t60...81d0s7    611       md        fs600   on
#======================================================================
# Local tape archive rb800
# Equipment                Equipment Equipment Family  Device Additional
# Identifier               Ordinal   Type      Set     State  Parameters
#------------------------  --------- --------- ------  ------ ----------
/dev/scsi/changer/c1t0d5   800       rb        rb800   on
 /dev/rmt/0cbn              801       tp        rb800   on
 /dev/rmt/1cbn              802       tp        rb800   on
 /dev/rmt/2cbn              803       tp        rb800   on
 /dev/rmt/3cbn              804       tp        rb800   on
```

3. At the end of the archiving equipment definitions, start an entry for the server that will make tape resources available to clients. Enter the path to the SAM-Remote server configuration file, **`/etc/opt/SUNWsamfs/samremote`**, in the **`Equipment Identifier`** field, and assign an equipment ordinal number.

   In the example, we add some headings as comments and assign equipment ordinal number **`500`** to the server **`samremote`**:

```
root@server1:~# vi /etc/opt/SUNWsamfs/mcf
...
#=========================================================================
# Server samremote shares tape hardware and media with clients
# Equipment                   Equipment Equipment Family  Device Additional
# Identifier                  Ordinal   Type      Set     State  Parameters
#---------------------------  --------- --------- ------  ------ ----------
/etc/opt/SUNWsamfs/samremote  500
```

4. In the **`Equipment Type`** field of the new entry, enter **`ss`**, for SAM-Remote server equipment.

```
root@server1:~# vi /etc/opt/SUNWsamfs/mcf
...
#=========================================================================
# Server samremote shares tape hardware and media with clients
# Equipment                   Equipment Equipment Family  Device Additional
# Identifier                  Ordinal   Type      Set     State  Parameters
#---------------------------  --------- --------- ------  ------ ----------
/etc/opt/SUNWsamfs/samremote  500       ss
```

5. Assign a **`Family Set`** name that is unique across all hosts and servers, and set the device **`on`**.

   In the example, we assign the family set name **`ss500`** to the new equipment:

```
root@server1:~# vi /etc/opt/SUNWsamfs/mcf
...
#=======================================================================
# Server samremote shares tape hardware and media with clients
# Equipment                  Equipment Equipment Family  Device Additional
# Identifier                 Ordinal   Type      Set     State  Parameters
#--------------------------  --------- --------- ------  ------ ----------
/etc/opt/SUNWsamfs/samremote  500       ss        ss500   on
```

6. If you plan to configure more than ten SAM-Remote clients, add an additional server-equipment (type **ss**) entry for each successive group of one to ten clients.
7. Save the file, and close the editor.

```
root@server1:~# vi /etc/opt/SUNWsamfs/mcf
...
/etc/opt/SUNWsamfs/samremote  500       ss        ss500   on
:wq
root@server1:~#
```

8. Next, create the **samremote** server configuration file.

Create the `samremote` Server Configuration File

The SAM-Remote server configuration file defines the disk buffer characteristics and media to be used for each client. For each server that you need to configure, proceed as follows:

1. Log in to the SAM-Remote server host as **root**.

   In the example, the server is named **server1**:

```
root@server1:~#
```

2. On the server, create an **/etc/opt/SUNWsamfs/samremote** file in a text editor.

   In the example, we create the file with the **vi** editor. We start by documenting the file with some descriptive comments, as indicated by the hash (**#**) signs:

```
root@server1:~# vi /etc/opt/SUNWsamfs/samremote
# Server Configuration File:
# Defines the disk buffer and media that is available to each client.
```

3. Start the first client entry by starting a new line and entering the hostname, IP address, or fully qualified domain name of the client in the first column.

   The client identifier line must start with a non-space character. In the example, we identify the client using the hostname **client1**:

```
root@server1:~# vi /etc/opt/SUNWsamfs/samremote
# Server Configuration File:
# Defines the disk buffer and media that is available to each client.
client1
```

4. Start identifying the media that will be shared with the client. Start a new line of the form *indent* **media**, where *indent* is one or more spaces and **media** is a SAM-remote keyword:

```
root@server1:~# vi /etc/opt/SUNWsamfs/samremote
# Server Configuration File:
# Defines the disk buffer and media that is available to each client.
client1
        media
```

5. Identify each media type and source with a new line of the form *`indent equipment-number media-type VSNs`*, where:

   - *`indent`* is one or more spaces.
   - *`equipment-number`* is the equipment ordinal number that identifies the archival storage equipment in the *`mcf`* file.
   - *`media-type`* is the media identifier for the tape media used by this equipment (see Appendix A, "Glossary of Equipment Types" for a full list of Oracle HSM media types).
   - *`VSNs`* is a space-delimited list of one or more volume serial numbers, which are alphanumeric strings of up to 31 characters.

   In the example, we identify one source of shared media, a range of tape volumes (type **`tp`**) resident in a tape library with equipment ordinal number **`800`**. The available volumes are specified by a regular expression enclosed in parentheses: the expression **`VOL0[0-1][0-9]`** limits **`client1`** to volumes **`VOL000-VOL019`**:

```
client1
       media
         800 tp (VOL0[0-1][0-9])
```

   Note that each line can specify only one type of media. So, if the library were to support more than one media type, you would specify each type in a new entry:

```
      media
         800 ti VOL500 VOL501
         800 li (VOL0[0-1][0-9])
```

6. When you have finished identifying the media that will be shared with the client, close the list by entering the SAM-Remote keyword **`endmedia`**.

   In the example, **`client1`** is now fully configured:

```
client1
       media
         800 tp (VOL0[0-1][0-9])
       endmedia
```

7. If you need to configure additional clients, do so now. Add a new client configuration record for each, up to a maximum of ten (10). Then save the file and close the editor.

   To prevent contention for volumes and possible data loss, *make sure that clients do not share the same removable media volumes*.

   In the example, we configure one additional client, **`client2`**. The second client has access to a range of tape volumes resident in the same tape library as **`client1`**, equipment ordinal number **`800`**. But the regular expression in the configuration specifies a different set of volumes: **`VOL020-VOL039`**.

```
# Server Configuration File:
# Defines the disk buffer and media that is available to each client.
client1
       media
         800 tp (VOL0[0-1][0-9])
       endmedia
client2
       media
         800 tp (VOL02-3][0-9])
```

```
        endmedia
:wq
root@server1:~#
```

8. Next, configure the SAM-Remote clients.

Configure the SAM-Remote Clients

For each SAM-Remote client, perform the following tasks:

- Define the remote archiving equipment in the SAM-Remote client's **mcf** file.
- Create the SAM-Remote client configuration file.
- Configure the **archiver.cmd** file on the SAM-Remote client.

Define the Remote Archiving Equipment in the SAM-Remote Client's mcf File

1. Log in to the SAM-Remote client host as **root**.

   In the example, the SAM-Remote client is named **client1**:

```
root@client1:~#
```

2. On the client, open the **/etc/opt/SUNWsamfs/mcf** file in a text editor, and scroll down to the archiving equipment definitions.

   In the example, we use the **vi** editor. The file defines one Oracle HSM archiving file system, **fs100**. Local copies are stored in disk-archive **DISKVOL1**, a local ZFS file system. Note that the example includes clarifying headings that may not be present in actual files and abbreviates lengthy device paths.

```
root@client1:~# vi /etc/opt/SUNWsamfs/mcf
# Client's /etc/opt/SUNWsamfs/mcf file
#=======================================================================
# Oracle HSM archiving file system "fs100"
# Equipment                Equipment Equipment Family  Device Additional
# Identifier               Ordinal   Type      Set     State  Parameters
#-----------------------   --------- --------- ------  ------ ----------
fs100                      100       ms        fs100   on
 /dev/dsk/c10t60...7Bd0s7   110       md        fs100   on
 /dev/dsk/c10t60...48d0s7   111       md        fs100   on
#=======================================================================
# Disk archive "/diskvols/DISKVOL1" stores local archive copies
```

3. At the end of the archiving equipment definitions, start an entry for the equipment that the server will make available to the client. In the **Equipment Identifier** field, enter the path to the SAM-Remote server configuration file, and assign an equipment ordinal number.

   In the example, we name the client configuration **/etc/opt/SUNWsamfs/sc400** and assign the client the equipment ordinal number **400**. We also add some headings as comments:

```
root@client1:~# vi /etc/opt/SUNWsamfs/mcf
...
# Disk archive "/diskvols/DISKVOL1" stores local archive copies
#
#=======================================================================
# Client "sc400" accesses tape resources on server "samremote" (ss500)
# Equipment                Equipment Equipment Family  Device Additional
# Identifier               Ordinal   Type      Set     State  Parameters
```

```
#-----------------------  --------- --------- ------  ------ ----------
/etc/opt/SUNWsamfs/sc400   400
```

4. In the **`Equipment Type`** field of the new entry, enter **`sc`**, for SAM-Remote client equipment.

```
root@client1:~# vi /etc/opt/SUNWsamfs/mcf
...
#=====================================================================
# Client "sc400" accesses tape resources on server "samremote" (ss500)
# Equipment                Equipment Equipment Family  Device Additional
# Identifier               Ordinal   Type      Set     State  Parameters
#-----------------------  --------- --------- ------  ------ ----------
/etc/opt/SUNWsamfs/ss500   400       sc
```

5. Assign a **`Family Set`** name that is unique across all hosts and servers, and set the device **`on`**.

   In the example, we assign the family set name **`ss500`** to the new equipment.

```
root@client1:~# vi /etc/opt/SUNWsamfs/mcf
...
#=====================================================================
# Client "sc400" accesses tape resources on server "samremote" (ss500)
# Equipment                Equipment Equipment Family  Device Additional
# Identifier               Ordinal   Type      Set     State  Parameters
#-----------------------  --------- --------- ------  ------ ----------
/etc/opt/SUNWsamfs/ss500   400       sc        ss500   on
```

6. For each tape drive that the SAM-Remote server makes available, add a SAM-Remote pseudodevice to the SAM-Remote client **`sc`** equipment. In the **`Equipment Identifier`** field, add an entry of the form **`/dev/samrd/rd`***`device-number`*, where *`device-number`* is an integer.

   In the example, we start entries for two pseudodevices, **`/dev/samrd/rd 0`** and **`/dev/samrd/rd 1`**:

```
root@client1:~# vi /etc/opt/SUNWsamfs/mcf
...
#=====================================================================
# Client "sc400" accesses tape resources on server "samremote" (ss500)
# Equipment                Equipment Equipment Family  Device Additional
# Identifier               Ordinal   Type      Set     State  Parameters
#-----------------------  --------- --------- ------  ------ ----------
/etc/opt/SUNWsamfs/sc400   400       sc        sc400   on
 /dev/samrd/rd0
 /dev/samrd/rd1
```

7. In the **`Equipment Ordinal`** field for each pseudodevice, enter a number in the range that you assigned to the **`sc`** equipment.

   In the example, we assign equipment ordinal **`410`** to **`/dev/samrd/rd0`** and equipment ordinal **`420`** to **`/dev/samrd/rd1`**:

```
root@client1:~# vi /etc/opt/SUNWsamfs/mcf
...
#=====================================================================
# Client "sc400" accesses tape resources on server "samremote" (ss500)
# Equipment                Equipment Equipment Family  Device Additional
# Identifier               Ordinal   Type      Set     State  Parameters
#-----------------------  --------- --------- ------  ------ ----------
/etc/opt/SUNWsamfs/sc400   400       sc        ss500   on
```

```
 /dev/samrd/rd0            410
 /dev/samrd/rd1            420
```

8. In the **`Equipment Type`** field for each SAM-Remote pseudodevice, enter **`rd`**, the equipment type for SAM-Remote pseudodevices.

```
root@client1:~# vi /etc/opt/SUNWsamfs/mcf
...
#======================================================================
# Client "sc400" accesses tape resources on server "samremote" (ss500)
# Equipment                Equipment Equipment Family  Device Additional
# Identifier               Ordinal   Type      Set     State  Parameters
#-----------------------   --------- --------- ------  ------ ----------
/etc/opt/SUNWsamfs/ss500   400       sc        ss500   on
 /dev/samrd/rd0             410       rd
 /dev/samrd/rd1             420       rd
```

9. In the **`Family Set`** field for each pseudodevice, enter the family set name for the **`sc`** equipment.

   In the example, we use the family set name **`ss500`**:

```
root@client1:~# vi /etc/opt/SUNWsamfs/mcf
...
#======================================================================
# Client "sc400" accesses tape resources on server "samremote" (ss500)
# Equipment                Equipment Equipment Family  Device Additional
# Identifier               Ordinal   Type      Set     State  Parameters
#-----------------------   --------- --------- ------  ------ ----------
/etc/opt/SUNWsamfs/ss500   400       sc        ss500   on
 /dev/samrd/rd0             410       rd        ss500
 /dev/samrd/rd1             420       rd        ss500
```

10. In the **`Device State`** field for each pseudodevice, enter **`on`**. Then save the file and close the editor.

    In the example, we assign equipment ordinal **`410`** to **`/dev/samrd/rd 0`** and equipment ordinal **`420`** to **`/dev/samrd/rd 1`**:

```
root@client1:~# vi /etc/opt/SUNWsamfs/mcf
...
#======================================================================
# Client "sc400" accesses tape resources on server "samremote" (ss500)
# Equipment                Equipment Equipment Family  Device Additional
# Identifier               Ordinal   Type      Set     State  Parameters
#-----------------------   --------- --------- ------  ------ ----------
/etc/opt/SUNWsamfs/ss500   400       sc        ss500   on
 /dev/samrd/rd0             410       rd        ss500   on
 /dev/samrd/rd1             420       rd        ss500   on
:wq
root@client1:~#
```

11. Next, create the SAM-Remote client configuration file.

Create the SAM-Remote Client Configuration File

For each SAM-Remote client, proceed as follows:

1. Log in to the SAM-Remote client host as **`root`**.

   In the example, the SAM-Remote client is named **`client1`**:

```
root@client1:~#
```

2. On the client, create an **/etc/opt/SUNWsamfs/***family-set-name* file in a text editor, where *family-set-name* is the family set name for the remote equipment as used in the **mcf** file.

   In the example, we create the file with the **vi** editor and name it for the family set **ss500**. We also document the file with some descriptive comments preceded by hash (**#**) signs:

```
root@client1:~# vi /etc/opt/SUNWsamfs/sc400
# Client's SAM-Remote client configuration file: /opt/SUNWsamfs/sc400
# This file identifies the host of the SAM-Remote server.
```

3. Add a single entry for the server by starting a new line and entering the hostname, IP address, or fully qualified domain name of the server in the first column. Then save the file and close the editor.

   The line must start with a non-space character. In the example, we identify the server using the hostname **server1**:

```
root@client1:~# vi /etc/opt/SUNWsamfs/samremote
# Client's SAM-Remote server configuration file: /opt/SUNWsamfs/sc400
# This file identifies the host of the SAM-Remote server.
server1
:wq
root@client1:~#
```

4. Next, configure the **archiver.cmd** file on the SAM-Remote client.

Configure the `archiver.cmd` file on the SAM-Remote Client

1. Log in to the SAM-Remote client host as **root**.

   In the example, the SAM-Remote client is named **client1**:

```
root@client1:~#
```

2. Open the **/etc/opt/SUNWsamfs/archiver.cmd** file in a text editor, and scroll down to the copy parameter directives, which start at the keyword **params** and end at the keyword **endparams**.

   In the example, we open the file in the **vi** editor:

```
root@client1:~# vi /etc/opt/SUNWsamfs/archiver.cmd
...
#-----------------------------------------------------------------------
# Copy Parameter Directives
params
allsets -sort path -offline_copy direct
allfiles.1 -startage 10m -startsize 500M -drives 10
allfiles.2 -startage 24h -startsize 20G -drives 2 -reserve set
endparams
```

3. Check the copy parameters for all archive sets that will be archived on remote media. If any of them includes **-tapenonstop** and/or **-offline_copy direct** directives, remove these directives now.

   In the example, the **all** parameter specifies the **-offline_copy direct** directive for all copies. So we override this directive by specifying **-offline_copy none** for the copy that we intend to send to remote media, **allfiles.3**:

```
#-----------------------------------------------------------------------
# Copy Parameter Directives
# Copy Parameter Directives
params
allsets -sort path -offline_copy direct
allfiles.1 -startage 10m -startsize 500M -drives 10
allfiles.2 -startage 24h -startsize 20G -drives 2 -reserve set offline_
copy none
endparams
```

4. Scroll down to the VSN directives, which start at the SAM-Remote keyword **vsns** and end at the keyword **endvsns**.

   In the example, we use the **vi** editor. The only copy that currently has media assigned, **allfiles.1**, will be made using the local disk archive volume, **qfs200**:

```
...
endparams
#-----------------------------------------------------------------------
# VSN Directives
vsns
allfiles.1 dk qfs200
endvsns
```

5. Assign archive copies to the remote media, as specified for this client in the server's **/etc/opt/SUNWsamfs/samremote** file. Then save the file and close the editor.

   In the example, we are configuring **client1**. Copy **allfiles.2** will be made using a remote tape volume in the range **VOL000-VOL019**, as specified in the **samremote** server configuration file:

```
...
endparams
#-----------------------------------------------------------------------
# VSN Directives
vsns
allfiles.1 dk qfs200
allfiles.2 tp VOL0[0-1][0-9]
endvsns
:wq
root@client1:~#
```

6. Next, validate the archiving configuration on the SAM-Remote server.

Validate the Archiving Configuration on the SAM-Remote Server

1. Log in to the SAM-Remote server host as **root**.

   In the example, the SAM-Remote server is named **server1**:

```
root@server1:~#
```

2. Start the Oracle HSM processes on the server. Use the command **samd start**:

```
root@server1:~# samd start
```

3. On the server host, check the status of the shared-device server. Use the command **samcmd s**.

   In the example, the SAM-Remote server equipment (type **ss**) with equipment ordinal number **500** is **on** and operating normally:

```
root@server1:~# samcmd s
Device status samcmd     6.0  11:20:34 Feb 20 2015
samcmd on server1
ty    eq  state   device_name                     fs     status
rb    800 on      /dev/scsi/changer/c1t0d5        800    m--------r
tp    801 on      /dev/rmt/0cbn                   800    ---------p
  empty
tp    802 on      /dev/rmt/1cbn                   800    ---------p
  empty
tp    803 on      /dev/rmt/2cbn                   800    ---------p
  empty
tp    804 on      /dev/rmt/3cbn                   800    ---------p
  empty
ss    500 on      /etc/opt/SUNWsamfs/samremote    ss500  -------o-r
root@server1:~#
```

4. If the shared-device server is not **on**, make sure that it is correctly defined in the server host's **/etc/opt/SUNWsamfs/mcf** file. Make sure that the **/etc/opt/SUNWsamfs/samremote** file is correct and in the correct location.

   See the procedures "Define the Remotely Shared Archiving Equipment in the SAM-Remote Server's mcf File" on page 7-4 and "Create the samremote Server Configuration File" on page 7-6.

5. On the server, check the connection status of the SAM-Remote clients. Use the command **samcmd R**.

   In the example, both **client1** and **client2** are in state **0005** and are thus **connected** (state **0004** indicates no connection):

```
root@server1:~# samcmd R
Remote server eq: 500 addr: 00003858 samcmd 6.0  11:20:44 Feb 20 2015
samcmd on server1
message:
Client IPv4: client1 192.10.10.3 port - 5000
 client index - 0 port - 31842 flags - 0005 connected
Client IPv4: client2 10.1.229.97 port - 5000
 client index - 1 port - 32848 flags - 0005 connected
root@server1:~#
```

6. If a shared-device client is not connected (state **0004**), check network connectivity. Make sure that server and client(s) can resolve each other's hostnames and addresses. Make sure that server and client(s) can reach each other.

   In the example, we use **ssh** with the **getent** and **ping** commands to check connectivity from each host to each of the other hosts in the SAM-Remote configuration:

```
root@server1:~# getent hosts client1
192.10.10.3 client1
root@server1:~# getent hosts 192.10.10.3
192.10.10.3 client1
root@server1:~# ping 192.10.10.3
192.10.10.31 is alive
root@server1:~# getent hosts client2
10.1.229.97 client2
root@server1:~# getent hosts 10.1.229.97
10.1.229.97 client2
root@server1:~# ping 10.1.229.97
192.10.10.31 is alive
root@server1:~# ssh root@client1
Password: ...
```

```
root@client1:~# getent hosts server1
192.10.201.12 server1
...
root@client1:~# exit
root@server1:~# ssh root@client2
Password: ...
[client2]root@solaris:~# getent hosts server1
192.10.201.12 server1
...
[client2]root@solaris:~# exit
root@server1:~#
```

7. If a shared-device client is not connected (state **0004**), make sure that it is correctly defined in the client host's **/etc/opt/SUNWsamfs/mcf** file. Make sure that the server host is correctly identified in the **/etc/opt/SUNWsamfs/***family-set-name* file and that the file is in the correct location on the client host. Then make sure that the client hosts are correctly identified in the **/etc/opt/SUNWsamfs/samremote** file on the server host.

   See the procedures "Define the Remote Archiving Equipment in the SAM-Remote Client's `mcf` File" on page 7-8 and "Create the SAM-Remote Client Configuration File" on page 7-10.

8. On the client, make sure that the server host is correctly identified in the **/etc/opt/SUNWsamfs/***family-set-name* file and that the file is in the correct location on the client host.

   See the procedure"Create the SAM-Remote Client Configuration File" on page 7-10.

9. If a shared-device client is not connected (state **0004**) and the client-side configuration files are not the problem, check the server. Make sure that the client hosts are correctly identified in the **/etc/opt/SUNWsamfs/samremote** file.

   See the procedure "Create the `samremote` Server Configuration File" on page 7-6.

10. On the server, make sure that each client can access the catalog for the shared tape library and view the available volumes. Use the command **samcmd v** *equipment-number*, where *equipment-number* is the equipment ordinal that the client's **mcf** file assigns to the SAM-Remote client equipment.

    In the example, we check **client1**, so **400** is the equipment number for the SAM-Remote client equipment, **/etc/opt/SUNWsamfs/sc400**. The output correctly lists the volumes that **client1** can access, **VOL000** to **VOL019**:

```
root@server1:~# samcmd v 400
Robot catalog samcmd     6.0  12:20:40 Feb 20 2015
samcmd on server1
Robot VSN catalog by slot       : eq 400
slot     access time  count use flags         ty vsn
   3     none         0     0%  -il-o-b-----  li VOL000
   7     none         0     0%  -il-o-b-----  li VOL001
...
  24     none         0     0%  -il-o-b-----  li VOL019
root@server1:~#
```

11. If a shared-equipment client cannot see the correct volumes, check the host files. On the server host, make sure that the assigned volumes are correctly identified in the **/etc/opt/SUNWsamfs/samremote** file. On the client host, make sure that the **/etc/opt/SUNWsamfs/***family-set-name* file correctly identifies the server host.

See the procedures "Create the `samremote` Server Configuration File" on page 7-6 and "Create the SAM-Remote Client Configuration File" on page 7-10.

12. Next, validate the archiving configuration on each SAM-Remote client.

Validate the Archiving Configuration on Each SAM-Remote Client

For each SAM-Remote client, proceed as follows:

1. Log in to the SAM-Remote client host as **`root`**.

   In the example, the SAM-Remote client is named **`client1`**:

```
root@client1:~#
```

2. Start the Oracle HSM processes on the client host. Use the command **`samd start`**:

```
root@client1:~# samd start
root@client1:~#
```

3. On the client host, check the status of the shared-device client. Use the command **`samcmd s`**.

   In the example, the SAM-Remote client equipment (type **`sc`**) with equipment ordinal number **`400`** is **`on`** and operating normally:

```
root@client1:~# samcmd s
Device status samcmd     6.0  12:20:49 Feb 20 2015
samcmd on client1
ty    eq  state   device_name                     fs     status
sc    400 on      /etc/opt/SUNWsamfs/sc400        sc400  -------o-r
```

4. If the shared-device client is not **`on`**, make sure that the **`sc`** device is correctly defined. On the client host, check the **`/etc/opt/SUNWsamfs/mcf`** file, and make sure that the **`/etc/opt/SUNWsamfs/`***`family-set-name`* file is correct and in the correct location.

   See the procedures "Define the Remote Archiving Equipment in the SAM-Remote Client's `mcf` File" on page 7-8 and "Create the SAM-Remote Client Configuration File" on page 7-10.

5. On the client host, confirm that the **`/etc/opt/SUNWsamfs/archiver.cmd`** file specifies the correct volume serial numbers for the remote media. List the file using the command **`archiver -A`**.

   In the example, we are configuring **`client1`**. Copy **`allfiles.2`** will be made using one of the remote tape volumes in the range **`VOL000-VOL019`**, as specified in the **`samremote`** server configuration file:

```
root@client1:~# archiver -A
Reading '/etc/opt/SUNWsamfs/archiver.cmd'.
1: # archiver.cmd
2: #-----------------------------------------------------------------------
3: # Global Directives
4: archivemeta = off
5: examine = noscan
...
30: #-----------------------------------------------------------------------
31: # VSN Directives
32: vsns
33: allfiles.1 dk qfs200
34: allfiles.2 tp VOL0[0-1][0-9]
```

```
36: endvsns
root@client1:~#
```

6. If you note any discrepancies in the **archiver.cmd** file, correct them before continuing.
7. If you intend to configure recycling, see configuring recycling for SAM-Remote.

Configuring Recycling for SAM-Remote

When SAM-Remote is configured, you must insure that recycling on one host cannot destroy valid data on another. Any recycling directives that you configure on a SAM-Remote server must recycle only the media that the server uses for its own archive sets. The server must not try to recycle media volumes that it has made available to SAM-Remote clients. Similarly, any recycling directives that you configure on a SAM-Remote client must recycle only the media that holds archived client data, either locally or in the designated volumes made available by the server.

You should thoroughly understand the recycling process before trying to use the recycler in a SAM-Remote environment. So read "Recycling" on page 1-8 and the **sam-recycler**, **archiver.cmd**, **recycler.cmd**, and **recycler.sh** man pages.

Then, when you are familiar with how recycling works, carry out the tasks below:

- Configure recycling on the SAM-Remote server.
- Configure recycling on the SAM-Remote client.

Configure Recycling on the SAM-Remote Server

If you need to configure recycling for file systems that the SAM-Remote server hosts, proceed as follows:

1. Log in to the SAM-Remote server as **root**.

   In the example, the SAM-Remote server is named **server1**:

   ```
   root@server1:~#
   ```

2. Open the **/etc/opt/SUNWsamfs/archiver.cmd** file in a text editor. Scroll down to the **params** section.

   In the example, we open the file in the **vi** editor:

   ```
   root@client1:~# vi /etc/opt/SUNWsamfs/archiver.cmd
   ...
   #-----------------------------------------------------------------------
   # Copy Parameter Directives
   params
   allsets -sort path -offline_copy direct
   allfiles.1 -startage 10m -startsize 500M -drives 10
   allfiles.2 -startage 24h -startsize 20G -drives 2 -reserve set
   endparams
   ```

3. Enter your recycler directives by archive set, in the form `archive-set directive-list`, where archive-set is one of the archive sets and `directive-list` is a space-delimited list of directive name/value pairs (for a full list of recycling directives, see the **archiver.cmd** man page).

   When using SAM-Remote, you must configure recycling by archive sets, in the **params** section of the **archiver.cmd** file. You cannot specify recycling by library.

In the example, we add recycling directives for archive sets **allfiles.1** and **allfiles.2**. The **-recycle_mingain 90** directive does not recycle a volume unless at least 90 percent of the volume's capacity can be recovered. The **-recycle_hwm 60** directive starts recycling when 60 percent of the removable media capacity has been used. The **-recycle_vsncount 1** schedules no more than one removable media volume for recycling at a time:

```
#-----------------------------------------------------------------------
#  Copy Parameters Directives
params
allsetsallfiles. -sort path -offline_copy direct
allfiles.1 -startage 10m -startsize 500M -drives 10
allfiles.1 -recycle_mingain 90
allfiles.2 -startage 24h -startsize 20G -drives 2 -reserve set offline_
copy none
allfiles.2 -recycle_hwm 60 -recycle_mingain 90 -recycle_vsncount 1
endparams
```

Note that the recycling directives defined on the SAM-Remote server apply only to archival volumes that the server uses for its own archive sets. The server's recycling directives do not apply to volumes that are accessible from the clients.

In the example, the server's recycling directives for copy **allfiles.2** apply to the tape volumes listed for the server's use in the **VSN Directives** section, **VOL100-VOL199**. The server's recycling directives do not apply to volumes **VOL000-VOL019**, which are reserved for **client1**, or to volumes **VOL020-VOL039**, which are reserved for **client2**:

```
...
endparams
#-------------------------------------------------------------------------
# VSN Directives
vsns
allfiles.1 dk DISKVOL1
allfiles.2 tp VOL1[0-9][0-9]
endvsns
```

4. Save the **archiver.cmd** file, and close the editor.

```
...
endvsns
:wq
root@server1:~#
```

5. On the server, create the **recycler.cmd** file in a text editor. Specify a path and file name for the recycler log.

   In the example, we use the **vi** editor. We specify the default location for the log file:

```
root@server1:~# vi /etc/opt/SUNWsamfs/recycler.cmd
logfile = /var/adm/recycler.log
```

6. In the **recycler.cmd** file on the server, add a directive of the form **no-recyle** *media-type volumes*, where *media-type* is one of the media types specified in Appendix A, "Glossary of Equipment Types" and where *volumes* is a space-delimited list or regular expression that specifies a volume serial number for every archival storage volume that you have assigned to SAM-Remote clients. Save the file and close the editor.

The **no-recyle** directive provides additional protection for storage resources that are dedicated to client use. It explicitly orders the host recycling processes to skip the specified volumes.

In the example, we add a **no-recyle** directive for media type **tp** (tape) volumes in the ranges **VOL000-VOL019** and **VOL020-VOL039**:

```
root@server1:~# vi /etc/opt/SUNWsamfs/recycler.cmd
logfile = /var/opt/SUNWsamfs/recycler/recycler.log
no_recycle tp VOL0[0-1][0-9] VOL0[2-3][0-9]
:wq
root@server1:~#
```

7. Now, configure recycling on the SAM-Remote client.

Configure Recycling on the SAM-Remote Client

For each client, proceed as follows:

1. Log in to the SAM-Remote client as **root**.

   In the example, the SAM-Remote client is named **client1**:

   ```
   root@client1:~#
   ```

2. On the client, open the **/etc/opt/SUNWsamfs/archiver.cmd** file in a text editor, and scroll down to the copy **params** section.

   In the example, we open the file in the **vi** editor.

   ```
   root@client1:~# vi /etc/opt/SUNWsamfs/archiver.cmd
   ...
   #-----------------------------------------------------------------------
   # Copy Parameters Directives
   params
   allsets -sort path -offline_copy stageahead
   allfiles.1 -startage 6h  -startsize 6G  -startcount 500000
   allfiles.2 -startage 24h -startsize 20G -startcount 500000 -archmax 24G
   endparams
   #-----------------------------------------------------------------------
   # VSN Directives
   vsns
   allfiles.1 dk qfs200
   allfiles.2 tp VOL0[0-1][0-9]
   endvsns
   ```

3. In the **params** section of the **archiver.cmd** file, enter your recycler directives by archive set, in the form *archive-set directive-list*, where archive-set is one of the archive sets and *directive-list* is a space-delimited list of directive name/value pairs (for a list of recycling directives, see the **archiver.cmd** man page). Then save the file and close the editor.

   When using SAM-Remote, you must configure recycling by archive sets, in the **params** section of the **archiver.cmd** file. You cannot specify recycling by library.

   In the example, we add recycling directives for archive sets **allfiles.1** and **allfiles.2**. The **-recycle_mingain 90** directive does not recycle volumes unless at least 90 percent of the volume's capacity can be recovered. The **-recycle_hwm 60** directive starts recycling when 60 percent of the removable media capacity has been used. The **-recycle_vsncount 1** directive schedules no more than one removable media volume for recycling at a time.

```
#-----------------------------------------------------------------------
# Copy Parameters Directives
params
allsets -sort path -offline_copy stageahead
allfiles.1 -startage 6h  -startsize 6G  -startcount 500000
allfiles.1 -recycle_mingain 90
allfiles.2 -startage 24h -startsize 20G -startcount 500000 -archmax 24G
allsets.2 -recycle_hwm 60 -recycle_mingain 90 -recycle_vsncount 1
endparams
```

Note that the recycling directives defined on the client apply only to media that the client uses for its own archive sets. In the example, the client's recycling directives for copy **allfiles.2** apply to server-provided remote tape volumes in the range **VOL000-VOL019**. They do not apply to volumes in the range **VOL020-VOL039**, which are reserved for **client2**, or to volumes in the range **VOL100-VOL119**, which are reserved for the server:

```
...
endparams
#-----------------------------------------------------------------------
# VSN Directives
vsns
allfiles.1 dk qfs200
allfiles.2 tp VOL0[0-1][0-9]
endvsns
:wq
root@client1:~#
```

4. Save the **archiver.cmd** file, and close the editor.

```
...
endvsns
:wq
root@client1:~#
```

5. On the client, create the **recycler.cmd** file in a text editor. Specify a path and file name for the recycler log. Then save the file and close the editor.

   We have configured the server and clients so that the client does not have access any of the archival media used by the server or by **client2**. So we do not need to add **no-recyle** directives.

   In the example, we use the **vi** editor. We specify the default location for the log file:

```
root@client1:~# vi /etc/opt/SUNWsamfs/recycler.cmd
logfile = /var/adm/recycler.log
:wq
root@client1:~#
```

6. Repeat this procedure until all SAM-Remote clients have been configured.

7. Enter the command **sam-recycler -dvxn**, where the parameters have the following effects:

   - **-d** displays volume-selection messages that indicate why each volume was or was not selected for recycling.
   - **-v** lists the files that are resident on each volume that is marked for recycling and will need to be moved.

- **-x** returns an error and stops if it lists any archive copies that are older than the time when the volume was labeled and are thus irrecoverable.
- **-n** prevents actual recycling. The recycling process behaves as if all archive set definitions in the **archiver.cmd** file included the **-recycle_ignore**, so you can test the recycling configuration non-destructively.

8. Once all SAM-Remote clients and servers have been configured, if you plan to use the sideband database feature, go to "Configuring the Reporting Database" on page 10-1.
9. Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

8

Accessing File Systems from Multiple Hosts

Oracle HSM file systems can be shared among multiple hosts in any of several ways. Each approach has particular strengths in some situations and notable drawbacks in others. So the method that you choose depends on your specific requirements. Sharing methods include:

- using Oracle HSM shared file systems
- using NFS and SMB/CIFS.

Accessing File Systems from Multiple Hosts Using Oracle HSM Software

Oracle HSM makes file systems available to multiple hosts by configuring a server and one or more clients that all mount the file system simultaneously. File data is then passed directly from the disk devices to the hosts via high-performance, local-path I/O, without the network and intermediate server latencies associated with NFS and CIFS share. Only one host can be active as a metadata server at any one time, but any number of clients can be configured as potential metadata servers for redundancy purposes. There is no limit to the number of file-system mount points.

Oracle HSM supports multi-host access to both high-performance (**`ma`**) and general-purpose (**`ms`**) file systems in both multi-reader/single-writer and shared configurations, with or without archiving. There are only a few limitations:

- Block (**`b`**–) special files are not supported.
- Character (**`c`**–) special files are not supported.
- FIFO named pipe (**`p`**–) special files are not supported.
- Segmented files are not supported.
- Mandatory locks are not supported.

  An **`EACCES`** error is returned if a mandatory lock is set. Advisory locks are supported, however. For more information about advisory locks, see the **`fcntl`** man page.

Oracle HSM software hosts can access file system data using either of two configurations, each with its own advantages and limitations in any given application.

In a *multi-reader, single-writer* configuration, a single host mounts the file system with read/write access and all other hosts mount it read-only. Configuration is a simple matter of setting mount options. Since a single host makes all changes to the files, file consistency and data integrity are insured, without additional file locking or consistency checks. All hosts read metadata as well as data directly from the disk for best performance. But all hosts must have access to file-system metadata, so all hosts in an **`ma`** file system must have access to both data and metadata devices.

In a *shared* configuration, all hosts can read, write, and append file data, using *leases* that allow a single host to access files in a given way for a given period of time. The metadata server issues *read*, *write*, and *append* leases and manages renewals and conflicting lease requests. Shared file systems offer great flexibility, but configuration is a bit more complex and there is more file-system overhead. All hosts read file data directly from disk, but clients access metadata over the network. So clients that lack access to metadata devices can share an **ma** file system.

To configure access to data from multiple Oracle HSM hosts, select the desired approach and see either "Configuring an Oracle HSM Single-Writer, Multiple-Reader File System" on page 8-2 or "Configuring an Oracle HSM Shared File System" on page 8-7.

Configuring an Oracle HSM Single-Writer, Multiple-Reader File System

To configure a single-writer, multiple-reader file system, carry out the following tasks:

- Create the file system on the writer.
- Configure the readers.

Create the File System on the Writer

Proceed as follows:

1. Log in to the host that will serve as the **writer** using the **root** account.

   In the example, the **writer** host is named **mds-write**:

```
root@mds-write:~#
```

2. On the host that will serve as the **writer**, open the **/etc/opt/SUNWsamfs/mcf** file in a text editor, and add a QFS file system. You can configure either a general-purpose **ms** or high-performance **ma** file system.

   On an **ma** file system with separate metadata devices, configure the metadata server for the file system as the writer. In the example below, we edit the **mcf** file on the host **mds-write** using the **vi** text editor. The example specifies an **ma** file system with the equipment identifier and family set name **swfs1** and the equipment ordinal number **300**:

```
root@mds-write:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment          Equipment  Equipment  Family     Device   Additional
# Identifier         Ordinal    Type       Set        State    Parameters
#------------------  ---------  ---------  ---------  ------   ---------------
swfs1                300        ma         swfs1  on
/dev/dsk/c0t0d0s0    301        mm         swfs1  on
/dev/dsk/c0t3d0s0    302        mr         swfs1  on
/dev/dsk/c0t3d0s1    303        mr         swfs1  on
```

3. Save the **/etc/opt/SUNWsamfs/mcf** file, and quit the editor.

   In the example, we save the changes and exit the **vi** editor:

```
# Equipment          Equipment  Equipment  Family     Device   Additional
# Identifier         Ordinal    Type       Set        State    Parameters
#------------------  ---------  ---------  ---------  ------   ---------------
swfs1                300        ma         swfs1  on
/dev/dsk/c0t0d0s0    301        mm         swfs1  on
/dev/dsk/c0t3d0s0    302        mr         swfs1  on
/dev/dsk/c0t3d0s1    303        mr         swfs1  on
:wq
```

```
root@mds-write:~#
```

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

   The **sam-fsd** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error:

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

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

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

6. Create the file system using the **sammkfs** command and the family set name of the file system, as described in "Configure a High-Performance ma File System" on page 6-5.

   In the example, the command creates the single-writer/multi-reader file system **swfs1**:

```
root@mds-write:~# sammkfs swfs1
Building 'swfs1' will destroy the contents of devices:
  /dev/dsk/c0t0d0s0
  /dev/dsk/c0t3d0s0
  /dev/dsk/c0t3d0s1
Do you wish to continue? [y/N]yes ...
```

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

```
root@mds-write:~# cp /etc/vfstab /etc/vfstab.backup
root@mds-write:~#
```

8. Add the new file system to the operating system's **/etc/vfstab** file, as described in "Configure a High-Performance ma File System" on page 6-5.

   In the example, we open the **/etc/vfstab** file in the **vi** text editor and add a line for the **swfs1** family set device:

```
root@mds-write:~# 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       -
...
swfs1      -        /hsm/swfs1      samfs   -     no
```

9. In the **Mount Options** column of the **/etc/vfstab** file, enter the **writer** mount option.

Caution: Make sure that only one host is the `writer` at any given time. Allowing more than one host to mount a multiple-reader, single-writer file system using the `writer` option can corrupt the file system!

```
#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       -
...
swfs1      -        /hsm/swfs1      samfs   -     no       writer
```

10. Make any other desired changes to the `/etc/vfstab` file. Add mount options using commas as separators.

    For example, to mount the file system in the background if the first attempt does not succeed, add the `bg` mount option to the `Mount Options` field (see the `mount_samfs` man page for a comprehensive list of available 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       -
...
swfs1      -        /hsm/swfs1      samfs   -     no       writer,bg
```

11. Save the `/etc/vfstab` file, and quit the editor.

```
#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       -
...
swfs1      -        /hsm/swfs1      samfs   -     no       writer,bg
:wq
root@mds-write:~#
```

12. 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 on all hosts, and 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/swfs1` mount-point directory and set permissions to `755` (`-rwxr-xr-x`):

```
root@mds-write:~# mkdir /hsm/swfs1
root@mds-write:~# chmod 755 /hsm/swfs1
root@mds-write:~#
```

13. Mount the new file system:

```
root@mds-write:~# mount /hsm/swfs1
root@mds-write:~#
```

14. Once the shared file system has been created, configure the readers.

Configure the Readers

A *reader* is a host that mounts a file system read-only. For each host that you are configuring as a reader, proceed as follows:

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

In the example, the **reader** host is named **reader1]**:

```
root@reader1:~#
```

2. In a terminal window, retrieve the configuration information for the multiple-reader, single-writer 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/dsk/***).

The **samfsconfig** utility retrieves file-system configuration information by reading the identifying superblock that **sammkfs** writes on each device that is included in an Oracle HSM file system. The command returns the correct paths to each device in the configuration starting from the current host and flags devices that cannot be reached (for full information on command syntax and parameters, see the **samfsconfig** man page).

In the example, the **samfsconfig** output shows the same equipment listed in the **mcf** file on **mds-write**, except that the paths to the devices are specified starting from the host **swfs1-reader1**:

```
root@reader1:~# samfsconfig /dev/dsk/*
# Family Set 'swfs1' Created Thu Nov 21 07:17:00 2013
# Generation 0 Eq count 4 Eq meta count 1
#
shrfs                300        ma         shrfs   -
/dev/dsk/c1t0d0s0    301        mm         shrfs   -
/dev/dsk/c1t3d0s0    302        mr         shrfs   -
/dev/dsk/c1t3d0s1    303        mr         shrfs   -
```

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

Alternatively, you could redirect the output of **samfsconfig** to the **mcf** file. Or you could use the **samd buildmcf** command to run **samfsconfig** and create the client **mcf** file automatically.

In the example, the **mcf** file for the host, **swfs1-reader1** looks like this once we add the commented out column headings:

```
root@reader1:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment           Equipment  Equipment  Family     Device   Additional
# Identifier          Ordinal    Type       Set        State    Parameters
#------------------   ---------  ---------  ---------  ------   ---------------
shrfs                 300        ma         shrfs    -
/dev/dsk/c1t0d0s0     301        mm         shrfs    -
/dev/dsk/c1t3d0s0     302        mr         shrfs    -
/dev/dsk/c1t3d0s1     303        mr         shrfs    -
```

4. Make sure that the **Device State** field is set to **on** for all devices. Then save the **mcf** file.

```
root@reader1:~# vi /etc/opt/SUNWsamfs/mcf
```

```
# Equipment          Equipment  Equipment  Family     Device   Additional
# Identifier         Ordinal    Type       Set        State    Parameters
#------------------  ---------  ---------  ---------  ------   ---------------
shrfs                300        ma         shrfs      on
/dev/dsk/c1t0d0s0    301        mm         shrfs      on
/dev/dsk/c1t3d0s0    302        mr         shrfs      on
/dev/dsk/c1t3d0s1    303        mr         shrfs      on
:wq
root@reader1:~#
```

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

   The **sam-fsd** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error:

```
root@reader1:~# sam-fsd
...
Would start sam-stagerd()
Would start sam-amld()
root@reader1:~#
```

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

```
root@reader1:~# cp /etc/vfstab /etc/vfstab.backup
root@reader1:~#
```

7. Add the single-writer, multiple-reader file system to the host operating system's **/etc/vfstab** file.

   In the example, we open the **/etc/vfstab** file in the **vi** text editor and add a line for the **swfs1** family set device:

```
root@reader1:~# 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       -
...
swfs1      -        /hsm/swfs1      samfs   -     no
```

8. In the **Mount Options** column of the **/etc/vfstab** file, enter the **reader** option.

   Caution: Make sure that the host mounts the file system using the **reader** option! Inadvertently using the **writer** mount option on more than one host can corrupt the file system!

```
#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       -
...
swfs1      -        /hsm/swfs1      samfs   -     no       reader
```

9. 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.

```
#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       -
...
swfs1      -        /hsm/swfs1  samfs   -     no       writer,bg
:wq
root@reader1:~#
```

10. 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 on the on all hosts, and 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/swfs1`** mount-point directory and set permissions to **`755`** (**`-rwxr-xr-x`**), just as we did on the writer host:

    ```
    root@reader1:~# mkdir /hsm/swfs1
    root@reader1:~# chmod 755 /hsm/swfs1
    root@reader1:~#
    ```

11. Mount the new file system:

    ```
    root@reader1:~# mount /hsm/swfs1
    root@reader1:~#
    ```

12. Repeat this procedure until all reader hosts have been configured to mount the file system read-only.
13. If you plan on using the sideband database feature, go to "Configuring the Reporting Database" on page 10-1.
14. Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

Configuring an Oracle HSM Shared File System

Oracle HSM shared file systems give multiple Oracle HSM hosts read, write, and append access to files. All hosts mount the file system and have direct connections to the storage devices. In addition, one host, the metadata server (MDS), has exclusive control over file-system metadata and mediates between hosts seeking access to the same files. The server provides client hosts with metadata updates via an Ethernet local network and controls file access by issuing, renewing, and revoking read, write, and append leases. Both non-archiving and archiving file systems of either the high-performance **`ma`** or general-purpose **`ms`** type can be shared.

To configure a shared file system, carry out the following tasks:

- Configure metadata servers for use with a shared file system.
- Configure file system clients.
- If required, configure archival storage for use with a shared file system.

Configuring Metadata Servers for Use with a Shared File System

To configure a metadata server to support a shared file system, carry out the tasks listed below:

- Create a hosts file on all active and potential metadata servers.

- Create the shared file system on the active metadata server.
- Mount the shared file system on the active metadata server.

Create a Hosts File on Active and Potential Metadata Servers

On the active and potential metadata servers, 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. Log in to the server as **root**.

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

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

2. 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.shrfs** 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.shrfs
   # /etc/opt/SUNWsamfs/hosts.shrfs
   #                                             Server   On/  Additional
   #Host Name           Network Interface       Ordinal  Off  Parameters
   #------------------  ----------------------  -------  ---  ----------
   ```

3. Add the hostname and IP address or domain name of the of the metadata server in two columns, separated by whitespace characters.

   ```
   # /etc/opt/SUNWsamfs/hosts.shrfs
   #                                             Server   On/  Additional
   #Host Name           Network Interface       Ordinal  Off  Parameters
   #------------------  ----------------------  -------  ---  ----------
   mds1                 10.79.213.117
   ```

4. Add a third column, separated from the network address by whitespace characters. In this column, enter **1**, the ordinal number for the active metadata server.

   In this example, there is only one metadata server, so we enter **1**:

   ```
   # /etc/opt/SUNWsamfs/hosts.shrfs
   #                                             Server   On/  Additional
   #Host Name           Network Interface       Ordinal  Off  Parameters
   #------------------  ----------------------  -------  ---  ----------
   mds1                 10.79.213.117           1
   ```

5. Add a fourth column, separated from the network address 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 **shrfs** man page).

   ```
   # /etc/opt/SUNWsamfs/hosts.shrfs
   ```

```
#                                            Server   On/  Additional
#Host Name           Network Interface       Ordinal  Off  Parameters
#------------------  ----------------------  -------  ---  ----------
mds1                 10.79.213.117            1        0
```

6. Add a fifth column, separated from the network address by whitespace characters. In this column, enter the keyword **server** to indicate the currently active metadata server:

```
# /etc/opt/SUNWsamfs/hosts.shrfs
#                                            Server   On/  Additional
#Host Name           Network Interface       Ordinal  Off  Parameters
#------------------  ----------------------  -------  ---  ----------
mds1                 10.79.213.117           1        0    server
```

7. 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**:

```
# /etc/opt/SUNWsamfs/hosts.shrfs
#                                            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
```

8. Add a line for each client host, each with a server ordinal value of **0**.

   A server ordinal of **0** identifies the host as a client. In the example, we add two clients, **clnt1** and **clnt2**.

```
# /etc/opt/SUNWsamfs/hosts.shrfs
#                                            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                10.79.213.147           0        0
```

9. 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.shrfs** and exit the **vi** editor:

```
# /etc/opt/SUNWsamfs/hosts.shrfs
#                                            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                10.79.213.147           0        0
:wq
root@mds1:~#
```

10. 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.

11. Now create the shared file system on the active metadata server.

Create the Shared File System on the Active Server

Proceed as follows:

1. Log in to the server as **root**.

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

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

2. On the metadata server (MDS), open the **/etc/opt/SUNWsamfs/mcf** file in a text editor and add a QFS file system. You can configure either a general-purpose **ms** or high-performance **ma** file system.

   In the example below, we edit the **mcf** file on the host **mds1** using the **vi** text editor. The example specifies an **ma** file system with the equipment identifier and family set name **shrfs** and the equipment ordinal number **300**:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment           Equipment  Equipment  Family     Device   Additional
# Identifier          Ordinal    Type       Set        State    Parameters
#------------------   ---------  ---------  ---------  ------   ---------------
shrfs                 300        ma         shrfs      on
/dev/dsk/c0t0d0s0     301        mm         shrfs      on
/dev/dsk/c0t3d0s0     302        mr         shrfs      on
/dev/dsk/c0t3d0s1     303        mr         shrfs      on
```

3. In the **Additional Parameters** field of the row for the **ma** file-system equipment, enter the **shared** parameter:

```
# Equipment           Equipment   Equipment  Family     Device   Additional
# Identifier          Ordinal    Type       Set        State    Parameters
#------------------   ---------  ---------  ---------  ------   ---------------
shrfs                 300         ma        shrfs      on       shared
/dev/dsk/c0t0d0s0     301         mm        shrfs      on
/dev/dsk/c0t3d0s0     302         mr        shrfs      on
/dev/dsk/c0t3d0s1     303         mr        shrfs      on
```

4. Save the **/etc/opt/SUNWsamfs/mcf** file, and quit the editor.

   In the example, we save the changes and exit the **vi** editor:

```
shrfs                 300         ma         shrfs     on      shared
/dev/dsk/c0t0d0s0     301         mm         shrfs     on
/dev/dsk/c0t3d0s0     302         mr         shrfs     on
/dev/dsk/c0t3d0s1     303         mr         shrfs     on
:wq
root@mds1:~#
```

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

   The **sam-fsd** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error:

```
root@mds1:~# sam-fsd
...
Would start sam-stagerd()
Would start sam-amld()
root@mds1:~#
```

6. Tell the Oracle HSM service to reread the **mcf** file and reconfigure itself accordingly. Correct any errors reported and repeat as necessary.

```
root@mds1:~# samd config
root@mds1:~#
```

7. Create the file system using the **sammkfs -S** command and the family set name of the file system, as described in "Configure a High-Performance ma File System" on page 6-5.

   The **sammkfs** command reads the **hosts.***family-set-name* and **mcf** files and creates a shared file system with the specified properties. In the example, the command reads the sharing parameters from the **hosts.shrfs** file and creates the shared file system **shrfs**:

```
root@mds1:~# sammkfs -S shrfs
Building 'shrfs' will destroy the contents of devices:
  /dev/dsk/c0t0d0s0
  /dev/dsk/c0t3d0s0
  /dev/dsk/c0t3d0s1
Do you wish to continue? [y/N]yes ...
root@mds1:~#
```

8. Next, mount the shared file system on the active metadata server.

Mount the Shared File System on the Active Server

1. Log in to the server as **root**.

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

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

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

```
root@mds1:~# cp /etc/vfstab /etc/vfstab.backup
root@mds1:~#
```

3. Add the new file system to the operating system's **/etc/vfstab** file, as described in "Configure a High-Performance ma File System" on page 6-5.

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

```
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       -
...
shrfs      -        /hsm/shrfs     samfs   -     no
```

4. In the **Mount Options** column, enter the **shared** option:

```
#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       -
...
```

```
shrfs       -        /hsm/shrfs     samfs   -     no       shared
```

5. Make any other desired changes to the **/etc/vfstab** file.

   For example, to retry mounting the file system in the background if the initial attempt does not succeed, add the **bg** mount option to the **Mount Options** field (for a full description of available mount options, see the **mount_samfs** man page):

```
#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       -
...
shrfs      -        /hsm/shrfs     samfs   -     no       shared,bg
```

6. Save the **/etc/vfstab** file, and quit the editor.

```
#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       -
...
shrfs      -        /hsm/shrfs     samfs   -     no       shared,bg
:wq
root@mds1:~#
```

7. 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 on the metadata server and on all clients, and 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/shrfs** mount-point directory and set permissions to **755** (**-rwxr-xr-x**):

```
root@mds1:~# mkdir /hsm/shrfs
root@mds1:~# chmod 755 /hsm/shrfs
root@mds1:~#
```

8. Mount the new file system:

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

9. If your hosts are configured with multiple network interfaces, you may want to use local hosts files to route network communications.

10. Otherwise, once the shared file system has been created on the metadata server, configure file system clients for sharing.

Configuring File System Clients for a Shared File System

Clients include both hosts that are configured purely as clients and those that are configured as potential metadata servers. In most respects, configuring a client is much the same as configuring a server. Each client includes exactly the same devices as the server. Only the mount options and the exact path to the devices changes (controller numbers are assigned by each client host and may thus vary).

To configure one or more clients to support a shared file system, carry out the tasks listed below:

- Create the shared file system on the Solaris clients.
- Mount the shared file system on the Solaris clients.
- Create the shared file system on the Linux clients (if any).
- Mount the shared file system on the Linux clients (if any).

Create the Shared File System on the Solaris Clients

For each client, proceed as follows:

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

   In the example, the server is named **clnt1**:

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

2. In a terminal window, enter 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/***).

   The **samfsconfig** command retrieves the configuration information for the shared file system.

   ```
   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, **mds1**. Only the host-assigned device controller numbers differ:

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

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 **clnt2** has access to the data devices only. So the **samfsconfig** output looks like this:

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

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
#------------------   ---------  ---------  ---------  ------   --------------
shrfs                 300        ma         shrfs      -
/dev/dsk/c1t0d0s0     301        mm         shrfs      -
/dev/dsk/c1t3d0s0     302        mr         shrfs      -
/dev/dsk/c1t3d0s1     303        mr         shrfs      -
```

   In the second example, the host, **clnt2**, 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         shrfs   -
# /dev/dsk/c4t3d0s1    303         mr         shrfs   -
```

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 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
#------------------   ---------  ---------  ---------  ------   --------------
shrfs                 300        ma         shrfs      -        shared
/dev/dsk/c1t0d0s0     301        mm         shrfs      -
/dev/dsk/c1t3d0s0     302        mr         shrfs      -
/dev/dsk/c1t3d0s1     303        mr         shrfs      -
```

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
#------------------   ---------  ---------  ---------  ------   --------------
shrfs                 300        ma         shrfs      -        shared
# /dev/dsk/c4t3d0s0     302        mr         shrfs      -
# /dev/dsk/c4t3d0s1     303        mr         shrfs      -
```

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
#------------------   ---------  ---------  ---------  ------   --------------
```

```
shrfs                 300        ma        shrfs      on       shared
nodev                 301        mm        shrfs      on
# /dev/dsk/c4t3d0s0     302        mr        shrfs    -
# /dev/dsk/c4t3d0s1     303        mr        shrfs    -
```

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
#------------------   ---------  ---------  ---------  ------   --------------
shrfs                 300        ma         shrfs      on       shared
nodev                 301        mm         shrfs      on
/dev/dsk/c4t3d0s0     302        mr         shrfs      -
/dev/dsk/c4t3d0s1     303        mr         shrfs      -
```

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

    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
#------------------   ---------  ---------  ---------  ------   ---------------
shrfs                 300        ma         shrfs      on       shared
/dev/dsk/c1t0d0s0     301        mm         shrfs      on
/dev/dsk/c1t3d0s0     302        mr         shrfs      on
/dev/dsk/c1t3d0s1     303        mr         shrfs      on
:wq
root@clnt1:~#
```

    In the second example, the host, **`clnt2`**, does not have access to the metadata devices for the file system, so the **`mcf`** file ends up looking like this:

```
root@clnt2:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment           Equipment  Equipment  Family     Device   Additional
# Identifier          Ordinal    Type       Set        State    Parameters
#------------------   ---------  ---------  ---------  ------   ---------------
shrfs                 300        ma         shrfs      on       shared
nodev                 301        mm         shrfs      on
/dev/dsk/c4t3d0s0     302        mr         shrfs      on
/dev/dsk/c4t3d0s1     303        mr         shrfs      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`** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error. In the example, we check the **`mcf`** file on **`clnt1`**, and it runs without errors:

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

12. At this point, if your hosts are configured with multiple network interfaces, you may want to use local hosts files to route network communications.
13. Next, mount the shared file system on the Solaris clients.

Mount the Shared File System on the Solaris Clients

For each client, proceed as follows:

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

   In the example, the server is named **clnt1**:

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

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

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

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 **shrfs** 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       -
   ...
   shrfs      -        /hsm/shrfs    samfs   -     no
   ```

4. 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 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       -
   ...
   shrfs      -        /hsm/shrfs    samfs   -     no       -
   :wq
   root@clnt1:~#
   ```

5. 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 **/shrfs** mount-point directory and set permissions to **755** (**-rwxr-xr-x**):

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

6. Mount the shared file system:

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

7. If the shared file system includes Linux clients, create the shared file system on the Linux clients.
8. If you are configuring an Oracle HSM shared archiving file system, go to your next task, "Configuring Archival Storage for a Shared File System" on page 8-22.
9. Otherwise, stop here. You have configured the Oracle HSM shared file system.

Create the Shared File System on the Linux Clients

For each client, proceed as follows:

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

   In the example, the Linux client host is named **clntL**:

```
[root@clntL ~]#
```

2. In a terminal window, enter 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/***).

   The **samfsconfig** command retrieves the configuration information for the shared file system. Since Linux hosts do not have access to the metadata devices for the file system, the **samfsconfig** 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 **clntL** looks like this:

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

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, **clntL**, starts out looking like this:

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

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

```
[root@clntL ~]# vi /etc/opt/SUNWsamfs/mcf
```

```
# Equipment           Equipment  Equipment  Family     Device   Additional
# Identifier          Ordinal    Type       Set        State    Parameters
#------------------   ---------  ---------  ---------  ------   ---------------
shrfs                 300        ma         shrfs      -        shared
# /dev/sda4             302         mr         shrfs     -
# /dev/sda5             302         mr         shrfs   -
```

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:

```
[root@clntL ~]# vi /etc/opt/SUNWsamfs/mcf
# Equipment           Equipment  Equipment  Family     Device   Additional
# Identifier          Ordinal    Type       Set        State    Parameters
#------------------   ---------  ---------  ---------  ------   ---------------
shrfs                 300         ma        shrfs      on       shared
nodev                 301         mm        shrfs      on
# /dev/sda4            302         mr         shrfs      -
# /dev/sda5            303         mr         shrfs      -
```

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

```
[root@clntL ~]# vi /etc/opt/SUNWsamfs/mcf
# Equipment           Equipment  Equipment  Family     Device   Additional
# Identifier          Ordinal    Type       Set        State    Parameters
#------------------   ---------  ---------  ---------  ------   ---------------
shrfs                 300         ma        shrfs      on       shared
nodev                 301         mm        shrfs      on
/dev/sda4             302         mr        shrfs      -
/dev/sda5             303         mr        shrfs      -
```

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

```
[root@clntL ~]# vi /etc/opt/SUNWsamfs/mcf
# Equipment           Equipment  Equipment  Family     Device   Additional
# Identifier          Ordinal    Type       Set        State    Parameters
#------------------   ---------  ---------  ---------  ------   ---------------
shrfs                 300         ma        shrfs      on       shared
nodev                 301         mm        shrfs      on
/dev/sda4             302         mr        shrfs      on
/dev/sda5             303         mr        shrfs      on
:wq
[root@clntL ~]#
```

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

   The **sam-fsd** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error. In the example, we check the **mcf** file on the Linux client, **clntL**:

```
[root@clntL ~]# sam-fsd
...
Would start sam-stagerd()
Would start sam-amld()
[root@clntL ~]#
```

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

Mount the Shared File System on the Linux Clients

For each client, proceed as follows:

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

   In the example, the Linux client host is named **clntL**:

   ```
   [root@clntL ~]#
   ```

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

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

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

   In the example, after backing up the **/etc/fstab** file on **clntL**, we open the file in the **vi** text editor and add a line for the **shrfs** family set device:

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

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

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

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

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

```
[root@clntL ~]# vi /etc/fstab
#File
#Device    Mount         System    Mount                        Dump      Pass
#to Mount  Point         Type      Options                      Frequency Number
#--------  -------       --------  -----------------------      --------- ------
...
/proc      /proc         proc     defaults
shrfs      /hsm/shrfs    samfs    shared,noauto                  0         0
:wq
[root@clntL ~]#
```

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 **/hsm/shrfs** mount-point directory and set permissions to **755** (**-rwxr-xr-x**):

```
[root@clntL ~]# mkdir /hsm/shrfs
[root@clntL ~]# chmod 755 /hsm/shrfs
```

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

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

```
[root@clntL ~]# mount /hsm/shrfs
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
[root@clntL ~]#
```

9. If you are configuring an Oracle HSM shared archiving file system, go to your next task, "Configuring Archival Storage for a Shared File System" on page 8-22

10. If you plan on using the sideband database feature, go to "Configuring the Reporting Database" on page 10-1.

11. Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

Use Local Hosts Files to Route Network Communications

Individual hosts do not require local hosts files. The file system 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 Active and Potential Metadata Servers" on page 8-8). 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 looks up the network interfaces for other hosts on the metadata server. Hostnames and IP addresses are listed in the global hosts file for the file system, **/etc/opt/SUNWsamfs/hosts.***family-set-name*, where *family-set-name* is the family set number of the shared file system. Then the host looks for a local hosts file, **/etc/opt/SUNWsamfs/hosts.***family-set-name***.local**.

If there is no local hosts file, the host uses the interface addresses specified in the global hosts file. Hosts are used in the order specified by the global file.

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. Hosts are used in the order specified in the local file.

So, by using different addresses in each file, you can control the interfaces used by different hosts. To configure local hosts files, use the procedure outlined below:

1. On each active and potential metadata server host, edit the global hosts file for the shared file system so that it routes server and host communications in the required way.

   For the examples in this section, the shared file system, **shrfs2**, includes an active metadata server, **mds1**, and one 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 hostnames that Domain Name Service (DNS) can resolve to addresses on the public, local area network (LAN).

   So we edit **/etc/opt/SUNWsamfs/hosts.shrfs2**, the file system's global host file. We specify private network interface addresses for the active and potential servers. But, for the clients, we supply the host names rather than addresses:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/hosts.shrfs2
# /etc/opt/SUNWsamfs/hosts.shrfs2
#                                  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
:wq
root@mds1:~#
```

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 equipment identifier of the shared file system. Only include interfaces for the networks that you want the active and potential servers to use.

   In our 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.shrfs2.local**, lists *private addresses* for only two hosts, the active and the potential metadata servers:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/hosts.shrfs2.local
# /etc/opt/SUNWsamfs/hosts.shrfs2 on mds1
#                                  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.shrfs2.local
# /etc/opt/SUNWsamfs/hosts.shrfs2.local on mds2
#                                      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 equipment identifier of the shared file system. 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 files include *hostnames* for only two hosts, the active and potential metadata servers:

```
root@mds1:~# ssh root@clnt1
Password:
root@clnt1:~# vi /etc/opt/SUNWsamfs/hosts.shrfs2.local
# /etc/opt/SUNWsamfs/hosts.shrfs2.local on clnt1
#                                      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.shrfs2.local
# /etc/opt/SUNWsamfs/hosts.shrfs2.local on clnt2
#                                      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, go to "Mount the Shared File System on the Active Server" on page 8-11.
5. If you started this procedure while configuring a client, you should now "Mount the Shared File System on the Solaris Clients" on page 8-16.

Configuring Archival Storage for a Shared File System

To set up the archival storage for an archiving Oracle HSM shared file system, carry out the following tasks:

- Connect tape drives to server and datamover hosts using persistent bindings.
- Configure the hosts of the archiving file system so that they can use the archival storage.
- Distribute tape I/O across the hosts of the shared archiving file system (if required).

Connect Tape Drives to Server and Datamover Hosts Using Persistent Bindings

In a shared archiving file system, all potential metadata servers must have access to the library and tape drives. If you decide to distribute tape I/O across the hosts of the shared archiving file system, one or more clients will also need access to drives. So you must configure each of these hosts to address each of the drives in a consistent way.

The Solaris operating system attaches drives 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 removable media library. So you need to persistently bind the devices to each 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 procedure below outlines the required steps (for full information on creating persistent bindings, see the Solaris **devfsadm** and **devlinks** man pages and the administration documentation for your version of the Solaris operating system):

1. Log in to the active metadata server as **root**.

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

2. If you do not know the current physical order of the drives in the library, create a mapping file as described in "Determine the Order in Which Drives are Installed in the Library" on page 3-10.

   In the example, the **device-mappings.txt** file looks like this:

   ```
   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, add a line to the **/etc/devlink.tab** file that remaps a starting node in the Solaris tape device tree, **rmt/***node-number*, to the first drive in the library. Enter the line in 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 the device address for the first device in the library, **1**, **w500104f0008120fe**, and see that the device is currently attached to the host at **rmt/1**:

   ```
   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
```

So we create a line in **/etc/devlink.tab** that remaps the non-conflicting node **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:~# vi /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 (**rmt/61**, **rmt/62**, and **rmt/63**), 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. Create the same persistent tape-device links 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 have a potential metadata server, **mds2**, and a datamover client, **clnt1**. So we edit the **/etc/devlink.tab** files on each to match that on the active server, **mds1**. Then we delete the existing links in **/dev/rmt** on **mds2** and **clnt1**, and run **devfsadm -c tape** 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 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. Now, configure the hosts of the archiving file system so that they can use the archival storage.

Configure the Hosts of the Archiving File System to Use the Archival Storage

For the active metadata server and each potential metadata server and datamover client, proceed as follows:

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

   In the examples, we log in to a datamover client named **datamvr**:

   ```
   root@datamvr1:~#
   ```

2. Open the **/etc/opt/SUNWsamfs/mcf** file in a text editor.

   In the example, we use the **vi** editor.

   ```
   root@datamvr1:~# vi /etc/opt/SUNWsamfs/mcf
   # Equipment           Equipment  Equipment  Family     Device   Additional
   # Identifier          Ordinal    Type       Set        State    Parameters
   #------------------   ---------  ---------  ---------  ------   -------------
   shrfs                 100         ms        shrfs      on
   /dev/dsk/c1t3d0s3     101         md        shrfs      on
   /dev/dsk/c1t3d0s4     102         md        shrfs      on
   ...
   ```

3. Following the file system definitions in the **/etc/opt/SUNWsamfs/mcf** file, start a section for the archival storage equipment.

   In the example, we add some headings for clarity:

```
root@datamvr1:~# vi /etc/opt/SUNWsamfs/mcf
...
# Archival storage for copies:
#
# Equipment              Equipment Equipment Family    Device Additional
# Identifier             Ordinal   Type      Set       State  Parameters
#----------------------- --------- --------- --------- ------ ----------------
```

4. To add archival tape storage, start by adding an entry for the library. In the equipment identifier field, enter the device ID for the library and assign an equipment ordinal number:

   In this example, the library equipment identifier is **/dev/scsi/changer/c1t0d5**. We set the equipment ordinal number to **900**, the range following the range chosen for our disk archive:

```
# Archival storage for copies:
#
# Equipment              Equipment Equipment Family    Device Additional
# Identifier             Ordinal   Type      Set       State  Parameters
#----------------------- --------- --------- --------- ------ ----------------
/dev/scsi/changer/c1t0d5 900
```

5. Set the equipment type to **rb**, a generic SCSI-attached tape library, provide a name for the tape library family set, and set the device state **on**.

   In this example, we are using the library **lib1**   :

```
# Archival storage for copies:
#
# Equipment              Equipment Equipment Family    Device Additional
# Identifier             Ordinal   Type      Set       State  Parameters
#----------------------- --------- --------- --------- ------ ----------------
/dev/scsi/changer/c1t0d5 900       rb        lib1      on
```

6. In the **Additional Parameters** column, you can enter an optional, user-defined path and name for the library catalog.

   The optional, non-default path cannot exceed 127 characters. In the example, we use the default path, **var/opt/SUNWsamfs/catalog/**, with the user-defined catalog file name **lib1cat**. Note that, due to document layout limitations, the example abbreviates the path:

```
# Archival storage for copies:
#
# Equipment              Equipment Equipment Family   Device Additional
# Identifier             Ordinal   Type      Set      State  Parameters
#----------------------- --------- --------- -------- ------ ----------------
/dev/scsi/changer/c1t0d5 900       rb        lib1     on     .../lib1cat
```

7. Next, add an entry for each tape drive. Use the persistent equipment identifiers that we established in the procedure "Connect Tape Drives to Server and Datamover Hosts Using Persistent Bindings" on page 8-23.

```
# Archival storage for copies:
#
# Equipment              Equipment Equipment Family   Device Additional
# Identifier             Ordinal   Type      Set      State  Parameters
#----------------------- --------- --------- -------- ------ -----------------
/dev/scsi/changer/c1t0d5  900       rb       lib1     on     .../lib1cat
/dev/rmt/60cbn            901       tp       lib1     on
/dev/rmt/61cbn            902       tp       lib1     on
```

```
/dev/rmt/62cbn            903       tp       lib1     on
/dev/rmt/63cbn            904       tp       lib1     on
```

8. Finally, if you wish to configure an Oracle HSM historian yourself, add an entry using the equipment type **hy**. Enter a hyphen in the family-set and device-state columns and enter the path to the historian's catalog in additional-parameters column.

   The historian is a virtual library that catalogs volumes that have been exported from the archive. If you do not configure a historian, the software creates one automatically using the highest specified equipment ordinal number plus one.

   Note that the example abbreviates the path to the historian catalog for page-layout reasons. The full path is **/var/opt/SUNWsamfs/catalog/historian_cat**:

```
# Archival storage for copies:
#
# Equipment              Equipment Equipment Family   Device Additional
# Identifier             Ordinal   Type      Set      State  Parameters
#----------------------- --------- --------- -------- ------ ----------------
/dev/scsi/changer/c1t0d5 900       rb        lib1      on    ...catalog/lib1cat
/dev/rmt/60cbn           901       tp        lib1      on
/dev/rmt/61cbn           902       tp        lib1      on
/dev/rmt/62cbn           903       tp        lib1      on
/dev/rmt/63cbn           904       tp        lib1      on
historian                999       hy        -         -     .../historian_cat
```

9. Save the **mcf** file, and close the editor.

```
...
 /dev/rmt/3cbn            904       tp        lib1      on
historian                999       hy         -      -      .../historian_cat
:wq
root@datamvr1:~#
```

10. Check the **mcf** file for errors by running the **sam-fsd** command. Correct any errors found.

    The **sam-fsd** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error:

```
root@datamvr1:~# sam-fsd
...
Would start sam-stagealld()
Would start sam-stagerd()
Would start sam-amld()
root@datamvr1:~#
```

11. Tell the Oracle HSM service to reread the **mcf** file and reconfigure itself accordingly. Correct any errors reported and repeat as necessary.

```
root@datamvr1:~# samd config
Configuring SAM-FS
root@datamvr1:~#
```

12. Repeat this procedure until all active and potential metadata servers and all datamover clients have been configured to use the archival storage.
13. If required, distribute tape I/O across the hosts of the shared archiving file system.
14. If you plan on using the sideband database feature, go to "Configuring the Reporting Database" on page 10-1.

15. Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

Distribute Tape I/O Across the Hosts of the Shared Archiving File System

Starting with Oracle HSM Release 6.1.4, any client of a shared archiving file system that runs on Oracle 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).

To distribute tape I/O across shared file-system hosts, proceed as follows:

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

   In the example, the server's hostname is **`mds1`**:

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

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

   ```
   root@mds1:~# uname -r
   5.11
   root@mds1:~#
   ```

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

   In the example, we log in to client hosts **`clnt1`** and **`clnt2`** remotely using **`ssh`** and get the Solaris version from the log-in banner:

   ```
   root@mds1:~# ssh root@clnt1
   Password:
   Oracle Corporation      SunOS 5.11      11.1    September 2013
   root@clnt1:~# exit
   root@mds1:~# ssh root@clnt2
   Password:
   Oracle Corporation      SunOS 5.11      11.1    September 2013
   root@clnt2:~# exit
   root@mds1:~#
   ```

6. Calculate the amount of system memory that can be allocated as buffer space for each tape-drive in the distributed I/O configuration. Divide the total available memory by the number of drives and subtract a sensible safety margin:

   ```
   (total-memory bytes)/(drive-count drives) = memory bytes/drive
   (memory bytes/drive) - (safe-margin bytes/drive) = buffsize bytes/drive
   ```

   Oracle HSM allocates a buffer for each drive used. So make sure that you do not inadvertently configure more buffer space than system memory can provide. In the example, we find that we can allocate no more than 224 kilobytes per drive. So we round down to 128 to allow a margin of safety.

   ```
   ((3584 kilobytes)/(16 drives)) = 224 kilobytes/drive
   buffsize = 128 kilobytes/drive
   ```

7. Once you have calculated the size of the buffer that can be allocated to each drive, calculate an Oracle HSM device block size and a number of blocks that will fit in a buffer of the specified size.

   ```
   (number blocks/buffer)*block-size bytes/block/drive = buffersize bytes/drive
   ```

   Vary the number of blocks and the block size until the product of the two is less than or equal to the calculated buffer size. The number of blocks must be in the range **[2-8192]** In the example, we settle on two blocks of 64 kilobytes each per buffer:

   ```
   (2 blocks/buffer)*(64 kilobytes/block/drive) = 128 kilobytes/drive
   ```

8. On the metadata server, open the **/etc/opt/SUNWsamfs/archiver.cmd** file in a text editor. On a new line in the general directives section at the top of the file, enter **bufsize =** *media-type media-blocks*, where:

   - *media-type* is the type code that the **mcf** file assigns to the drives and media used for distributed I/O.
   - *media-blocks* is the number of blocks per buffer that you calculated above.

   Save the file, and close the editor.

   In the example, we log in to the server **mds1** and use the **vi** editor to add the line **bufsize = ti 2**, where **ti** is the media type for the Oracle StorageTek T10000 drives that we are using and **2** is the number of blocks per drive buffer that we calculated:

   ```
   root@mds1:~# vi /etc/opt/SUNWsamfs/archiver.cmd
   #archiver.cmd
   #-----------------------------------------------------------------------
   # General Directives
   archivemeta = off
   examine = noscan
   bufsize = ti 2
   :wq
   root@mds1:~#
   ```

9. On the metadata server, open the **/etc/opt/SUNWsamfs/defaults.conf** file in a text editor. For each media type that will participate in distributed I/O, enter a line of the form *media-type*_**blksize =***size* where:

   - *media-type* is the type code that the **mcf** file assigns to the drives and media used for distributed I/O.
   - *size* is the block size that you calculated earlier in this procedure.

   By default, the device block size for StorageTek T10000 drives is 2 megabytes or 2048 kilobytes (**ti_blksize = 2048**). So, in the example, we override the default with block size that we calculated, 64 kilobytes:

   ```
   root@mds1:~# 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.
   ...
   #li_blksize = 256
   ti_blksize = 64
   root@mds1:~#
   ```

10. While still in the **/etc/opt/SUNWsamfs/defaults.conf** file, uncomment the line **#distio = off**, if necessary, or add it if it is not present at all.

    By default, **distio** is **off** (disabled). In the example, we add the line **distio = on**:

    ```
    ...
    distio = on
    ```

11. While still in the **/etc/opt/SUNWsamfs/defaults.conf** file, enable each device type that should participate in distributed I/O. On a new line, enter *media-type* _ **distio = on**, where *media-type* is the type code that the **mcf** file assigns to drives and media.

    By default, StorageTek T10000 drives and LTO drives are allowed to participate in distributed I/O (**ti_distio = on** and **li_distio = on**), while all other types are excluded. In the example, we explicitly include StorageTek T10000 drives:

    ```
    ...
    distio = on
    ti_distio = on
    ```

12. While still in the **/etc/opt/SUNWsamfs/defaults.conf** file, disable each device type that should not participate in distributed I/O. On a new line, enter *media-type* **_distio = off**, where *media-type* is the type code that the **mcf** file assigns to drives and media.

    In the example, we exclude LTO drives:

    ```
    ...
    distio = on
    ti_distio = on
    li_distio = off
    ```

13. When you have finished editing the **/etc/opt/SUNWsamfs/defaults.conf** file, save the contents and close the editor.

    ```
    ...
    distio = on
    ti_distio = on
    li_distio = off
    :wq
    root@mds1:~#
    ```

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

15. On each client that will serve as a datamover, open the **/etc/opt/SUNWsamfs/mcf** file in a text editor, and update the file to include 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 use the **vi** editor to configure the **mcf** file on host **clnt1**:

    ```
    root@clnt1:~# vi /etc/opt/SUNWsamfs/mcf
    # Equipment              Equipment Equipment Family      Device Additional
    # Identifier             Ordinal   Type      Set         State  Parameters
    #----------------------- --------- --------- ----------  ------ --------------
    shrfs                    800       ms        shrfs       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
```

16. 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.

    In the example, the library is configured on 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
#----------------------- --------- --------- ----------  ------ --------------
shrfs                    800       ms        shrfs       on
...
# Archival storage for copies:
/dev/scsi/changer/c1t0d5 900       rb        lib1        on     .../lib1cat
/dev/rmt/60cbn           901       ti        lib1        on
/dev/rmt/61cbn           902       ti        lib1        on
/dev/rmt/62cbn           903       ti        lib1        on
/dev/rmt/63cbn           904       ti        lib1        on
:wq
root@clnt1:~#
```

17. 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. Then save the file and close the editor.

    In the example, the library is not configured on host **clnt2**, so we use the hyphen as the family set name for the tape devices:

```
root@clnt2:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment              Equipment Equipment Family      Device Additional
# Identifier             Ordinal   Type      Set         State  Parameters
#----------------------- --------- --------- ----------  ------ --------------
shrfs                    800       ms        shrfs       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:~#
```

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

    In the example, we log in to the server **mds1** and use the **vi** editor to turn distributed I/O **off** for copy **1**:

```
root@mds1:~# vi /etc/opt/SUNWsamfs/archiver.cmd
#archiver.cmd
...
params
allsets -sort path -offline_copy stageahead
```

```
allfiles.1 -startage 10m -startsize 500M -startcount 500000 -distio off
allfiles.2 -startage 24h -startsize 20G  -startcount 500000 -reserve set
:wq
root@mds1:~#
```

19. Check the configuration files for errors by running the **sam-fsd** command. Correct any errors found.

    The **sam-fsd** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error. In the example, we run the command on the server, **mds1**:

```
root@mds1:~# sam-fsd
```

20. Tell the Oracle HSM service to read the modified configuration files and reconfigure itself accordingly. Correct any errors reported and repeat as necessary.

```
root@mds1:~# samd config
```

21. To verify that distributed I/O has been successfully activated, use the command **samcmd g**. If the **DATAMOVER** flag appears in the output for the clients, distributed I/O has been successfully activated.

    In the example, the flag is present:

```
root@mds1:~# samcmd g
Shared clients samcmd 6.0.dist_tapeio 11:09:13 Feb 20 2014
samcmd on mds1
shrfs is shared, server is mds1, 2 clients 3 max
ord hostname             seqno nomsgs status   config  conf1  flags
  1 mds1          14      0   8091  808540d   4051      0 MNT SVR

    config   :   CDEVID      ARCHIVE_SCAN    GFSID   OLD_ARCHIVE_FMT
    "        :   SYNC_META   TRACE    SAM_ENABLED     SHARED_MO
    config1  :   NFSV4_ACL   MD_DEVICES      SMALL_DAUS      SHARED_FS
    flags    :
    status   :   MOUNTED     SERVER   SAM     DATAMOVER
    last_msg :  Wed Jul  2 10:13:50 2014

  2 clnt1     127      0   a0a1  808540d   4041      0 MNT CLI

    config   :   CDEVID      ARCHIVE_SCAN    GFSID   OLD_ARCHIVE_FMT
    "        :   SYNC_META   TRACE    SAM_ENABLED     SHARED_MO
    config1  :   NFSV4_ACL   MD_DEVICES      SHARED_FS
    flags    :
    status   :   MOUNTED     CLIENT   SAM     SRVR_BYTEREV
    "        :   DATAMOVER
...
```

22. If you plan on using the sideband database feature, go to "Configuring the Reporting Database" on page 10-1.

23. Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

Accessing File Systems from Multiple Hosts Using NFS and SMB/CIFS

Multiple hosts can access Oracle HSM file systems using Network File System (NFS) or Server Message Block (SMB)/Common Internet File System (CIFS) shares in place of or in addition to the Oracle HSM software's native support for multiple-host file-system access (see "Accessing File Systems from Multiple Hosts Using Oracle HSM Software" on page 8-1). The following sections outline the basic configuration steps:

- "Sharing Oracle HSM File Systems Using NFS" on page 8-33
- "Sharing Oracle HSM File Systems Using SMB/CIFS" on page 8-41.

Sharing Oracle HSM File Systems Using NFS

Carry out the following tasks:

- Disable delegation before using NFS 4 to share an Oracle HSM shared file system.
- Configure NFS servers and clients to share WORM files and directories (if required).
- Configure the NFS server on the Oracle HSM host.
- Share the Oracle HSM file system as an NFS share.
- Mount the NFS-shared Oracle HSM file system on the NFS clients.

Disable Delegation Before Using NFS 4 to Share an Oracle HSM Shared File System

If you use NFS to share an Oracle HSM shared file system, you need to make sure that the Oracle HSM software controls access to files without interference from NFS. This is not generally a problem, because, when the NFS server accesses files on behalf of its clients, it does so as a client of the Oracle HSM shared file system. Problems can arise, however, if NFS version-4 servers are configured to *delegate* control over read and write access to their clients. Delegation is attractive because the server only needs to intervene to head off potential conflicts. The server's workload is partially distributed across the NFS clients, and network traffic is reduced. But delegation grants access—particularly write access—independently of the Oracle HSM server, which also controls access from its own shared file-system clients. To prevent conflicts and potential file corruption, you need to disable delegation. Proceed as follows.

1. Log in to the metadata server (MDS) host of the Oracle HSM file system that you want to configure as an NFS share. Log in as **`root`**.

   In the examples below, the server name is **`mds1`**.

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

2. If you are using NFS version 4 and the NFS server runs Solaris 11.1 or later, use the **`sharectl set -p`** command of the Service Management Facility (SMF) to turn the NFS **`server_delegation`** property **`off`**.

   ```
   root@mds1:~# sharectl set -p server_delegation=off
   ```

3. If you are using NFS version 4 and the NFS server runs Solaris 11.0 or earlier, disable delegations by opening the **`/etc/default/nfs`** file in a text editor and setting the **`NFS_SERVER_DELEGATION`** parameter **`off`**. Save the file, and close the editor.

   In the example, we use the **`vi`** editor:

   ```
   root@mds1:~# vi /etc/default/nfs
   # ident "@(#)nfs        1.10    04/09/01 SMI"
   # Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
   # Use is subject to license terms.
   ...
   NFS_SERVER_DELEGATION=off
   :wq
   root@mds1:~#
   ```

4. If the Oracle HSM file system that you intend to share supports the Write-Once Read-Many (WORM) feature, configure NFS servers and clients to share WORM files and directories now.
5. Otherwise, configure the NFS server on the Oracle HSM host.

Configure NFS Servers and Clients to Share WORM Files and Directories

1. Log in to the metadata server (MDS) host of the Oracle HSM file system that you want to share using NFS. Log in as **`root`**.

   In the examples below, the server name is **`mds1`** and the client name is **`nfsclnt1`**.

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

2. If the Oracle HSM file system that you intend to share uses the WORM feature and is hosted on a server running under Oracle Solaris 10 or later, make sure that NFS version 4 is enabled on the NFS server and on all clients.

   In the example, we check the server **`mds1`** and the client **`nfsclnt1`**. In each case, we first check the Solaris version level using the **`uname -r`** command. Then we pipe the output of the **`modinfo`** command to **`grep`** and a regular expression that find the NFS version information:

   ```
   root@mds1:~# uname -r
   5.11
   root@mds1:~# modinfo | grep -i "nfs.* version 4"
   258 7a600000  86cd0  28   1  nfs (network filesystem version 4)
   root@mds1:~# ssh root@nfsclnt1
   Pasword: ...
   root@nfsclnt1:~# uname -r
   5.11
   root@nfsclnt1:~# modinfo | grep -i "nfs.* version 4"
   278 fffffffff8cba000  9df68  27   1  nfs (network filesystem version 4)
   root@nfsclnt1:~# exit
   root@mds1:~#
   ```

3. If NFS version 4 is not enabled on a server running under Oracle Solaris 10 or later, log in as **`root`** on the server and on each client. Then use the **`sharectl set`** command to enable NFS 4:

   ```
   root@mds1:~# sharectl set -p server_versmax=4 nfs
   root@mds1:~# ssh root@nfsclnt1
   Password ...
   root@nfsclnt1:~# sharectl set -p server_versmax=4 nfs
   root@nfsclnt1:~# exit
   root@mds1:~#
   ```

4. Next, configure the NFS server on the Oracle HSM host.

Configure the NFS Server on the Oracle HSM Host

Before clients can successfully mount an Oracle HSM file system using Network File System (NFS), you must configure the NFS server so that it does not attempt to share the Oracle HSM file system before the file system has been successfully mounted on the host. Under Oracle Solaris 10 and subsequent versions of the operating system, the Service Management Facility (SMF) manages mounting of file systems at boot time. If you do not configure NFS using the procedure below, either the QFS mount or the NFS share will succeed and the other will fail.

1. Log in to the metadata server (MDS) host of the Oracle HSM file system that you want to configure as an NFS share. Log in as **`root`**.

   In the examples below, the server name is **`mds1`**.

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

2. Export the existing NFS configuration to an XML manifest file by redirecting the output of the **`svccfg export /network/nfs/server`** command.

   In the example, we direct the exported configuration to the manifest file **`/var/tmp/server.xml`**:

   ```
   root@mds1:~# svccfg export /network/nfs/server > /var/tmp/server.xml
   root@mds1:~#
   ```

3. Open the manifest file in a text editor, and locate the **`filesystem-local`** dependency.

   In the example, we open the file in the **`vi`** editor. The entry for the **`filesystem-local`** dependency is listed immediately before the entry for the dependent **`nfs-server_multi-user-server`**:

   ```
   root@mds1:~# vi /var/tmp/server.xml
   <?xml version='1.0'?>
   <!DOCTYPE service_bundle SYSTEM '/usr/share/lib/xml/dtd/service_bundle.dtd.1'>
   <service_bundle type='manifest' name='export'>
     <service name='network/nfs/server' type='service' version='0'>
       ...
       <dependency name='filesystem-local' grouping='require_all' restart_
   on='error' type='service'>
         <service_fmri value='svc:/system/filesystem/local'/>
       </dependency>
       <dependent name='nfs-server_multi-user-server' restart_on='none'
           grouping='optional_all'>
         <service_fmri value='svc:/milestone/multi-user-server'/>
       </dependent>
       ...
   ```

4. Immediately after the **`filesystem-local`** dependency, add a **`qfs`** dependency that mounts the QFS shared file system. Then save the file, and exit the editor.

   This will mount the Oracle HSM shared file system before the server tries to share it via NFS:

   ```
   <?xml version='1.0'?>
   <!DOCTYPE service_bundle SYSTEM '/usr/share/lib/xml/dtd/service_bundle.dtd.1'>
   <service_bundle type='manifest' name='export'>
     <service name='network/nfs/server' type='service' version='0'>
       ...
       <dependency name='filesystem-local' grouping='require_all' restart_
   on='error' type='service'>
         <service_fmri value='svc:/system/filesystem/local'/>
       </dependency>
       <dependency name='qfs' grouping='require_all' restart_on='error'
   type='service'>
         <service_fmri value='svc:/network/qfs/shared-mount:default'/>
       </dependency>
       <dependent name='nfs-server_multi-user-server' restart_on='none'
           grouping='optional_all'>
         <service_fmri value='svc:/milestone/multi-user-server'/>
       </dependent>
   ```

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

5. Validate the manifest file using the **svccfg validate** command.

```
root@mds1:~# svccfg validate /var/tmp/server.xml
```

6. If the **svccfg validate** command reports errors, correct the errors and revalidate the file.

   In the example, the **svccfg validate** command returns XML parsing errors. We inadvertently omitted an ending tag **</dependency>** when saving the file. So we re-open the file in the **vi** editor and correct the problem:

```
root@mds1:~# svccfg validate /var/tmp/server.xml
/var/tmp/server.xml:75: parser error : Opening and ending tag mismatch:
dependency line 29 and service
  </service>
           ^
/var/tmp/server.xml:76: parser error : expected '>'
</service_bundle>
        ^
/var/tmp/server.xml:77: parser error : Premature end of data in tag service_
bundle line 3
^
svccfg: couldn't parse document
root@mds1:~# vi /var/tmp/server.xml
...
:wq
root@mds1:~#
```

7. Once the **svccfg validate** command completes without error, disable NFS using the **svcadm disable nfs/server** command.

   In the example, the **svccfg validate** command returned no output, so the file is valid and we can disable NFS:

```
root@mds1:~# svccfg validate /var/tmp/server.xml
root@mds1:~# svcadm disable nfs/server
```

8. Delete the existing NFS server configuration using the **svccfg delete nfs/server** command.

```
root@mds1:~# svccfg delete nfs/server
```

9. Import the manifest file into the Service Management Facility (SMF) using the **svccfg import** command.

```
root@mds1:~# svccfg import /var/tmp/server.xml
```

10. Re-enable NFS using the **svcadm enable nfs/server** command.

    NFS is configured to use the updated configuration.

```
root@mds1:~# svcadm enable nfs/server
```

11. Confirm that the **qfs** dependency has been applied. Make sure that the command **svcs -d svc:/network/nfs/server:default** displays the **/network/qfs/shared-mount:default** service:

```
root@mds1:~# svcs -d svc:/network/nfs/server:default
STATE          STIME    FMRI
...
```

```
online          Nov_01   svc:/network/qfs/shared-mount:default
...
```

12. Next, share the Oracle HSM file system as an NFS share.

Share the Oracle HSM File System as an NFS Share

Share the Oracle HSM file system using the procedures described in the administration documentation for your version of the Oracle Solaris operating system. The steps below summarize the procedure for Solaris 11.1:

1. Log in to the metadata server (MDS) host of the Oracle HSM file system that you want to share using NFS. Log in as **root**.

   In the examples below, the server name is **mds1**.

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

2. Enter the command line **share -F nfs -o** *sharing-options sharepath* where the **-F** switch specifies the **nfs** sharing protocol and **sharepath** is the path to the shared resource. If the optional **-o** parameter is used, *sharing-options* can include any of the following:

   - **rw** makes *sharepath* available with read and write privileges to all clients.
   - **ro** makes *sharepath* available with read-only privileges to all clients.
   - **rw=***clients* makes *sharepath* available with read and write privileges to *clients*, a colon-delimited list of one or more clients that have access to the share.
   - **ro=***clients* makes *sharepath* available with read-only privileges to *clients*, a colon-delimited list of one or more clients that have access to the share.

   In the example, we share the **hqfs1** file system read/write with clients **nfsclnt1** and **nfsclnt2** and read-only with **nfsclient3**:

   ```
   root@mds1:~# share -F nfs -o rw=nfsclnt1:nfsclnt2 ro=nfsclient3 /hsm/hqfs1
   ...
   root@mds1:~#
   ```

   When you enter the command, the system automatically restarts the NFS server daemon, **nfsd**. See the **share_nfs** man page for additional options and details.

3. Check the sharing parameters using the command line **share -F nfs**.

   In the example, the command output shows that we have correctly configured the share:

   ```
   root@mds1:~# share -F nfs
   /hsm/hqfs1   sec=sys,rw=nfsclnt1:nfsclnt2,ro=nfsclient3
   root@mds1:~#
   ```

4. Next, mount the NFS-shared Oracle HSM file system on the NFS clients.

Mount the NFS-Shared Oracle HSM File System on the NFS Clients

Mount the NFS server's file system at a convenient mount point on client systems. For each client, proceed as follows:

1. Log in to the client as **root**.

   In the example, the NFS client is named **nfsclnt1**:

   ```
   root@nfsclnt1:~#
   ```

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

```
root@nfsclnt1:~# cp /etc/vfstab /etc/vfstab.backup
root@nfsclnt1:~#
```

3. Open the **/etc/vfstab** file in a text editor.

   In the example, we use the **vi** editor.

```
root@nfsclnt1:~# vi /etc/vfstab
#File           Device                          Mount
#Device         to      Mount     System fsck  at     Mount
#to Mount       fsck    Point     Type   Pass  Boot   Options
#------------   ------  --------- ------ ----  -----  ----------------
/devices        -       /devices  devfs  -     no     -
...
```

4. In the first column of the **/etc/vfstab** file, name the file device that you want to mount by specifying the name of the NFS server and the mount point of the file system that you want to share, separated by a colon.

   In the example, the NFS server is named **mds1**, the shared file system is named **hqfs1**, and the mount point on the server is **/hqfs1**:

```
#File           Device                          Mount
#Device         to      Mount     System fsck  at     Mount
#to Mount       fsck    Point     Type   Pass  Boot   Options
#------------   ------  --------- ------ ----  -----  ----------------
/devices        -       /devices  devfs  -     no     -
...
mds1:/hsm/hqfs1
```

5. In the second column of the **/etc/vfstab** file, enter a hyphen (**-**) so that the local system does not try to check the remote file system for consistency:

```
#File           Device                          Mount
#Device         to      Mount     System fsck  at     Mount
#to Mount       fsck    Point     Type   Pass  Boot   Options
#------------   ------  --------- ------ ----  -----  ----------------
/devices        -       /devices  devfs  -     no     -
...
mds1:/hsm/hqfs1 -
```

6. In the third column of the **/etc/vfstab** file, enter the local mount point where you will mount the remote file system.

   In the example, the mount point will be the directory **/mds1**:

```
#File           Device                          Mount
#Device         to      Mount     System fsck  at     Mount
#to Mount       fsck    Point     Type   Pass  Boot   Options
#------------   ------  --------- ------ ----  -----  ----------------
/devices        -       /devices  devfs  -     no     -
...
mds1:/hsm/hqfs1 -       /mds1
```

7. In the fourth column of the **/etc/vfstab** file, enter the file-system type **nfs**.

```
#File           Device                          Mount
#Device         to      Mount     System fsck  at     Mount
#to Mount       fsck    Point     Type   Pass  Boot   Options
#------------   ------  --------- ------ ----  -----  ----------------
```

```
/devices        -       /devices  devfs  -     no      -
...
mds1:/hsm/hqfs1 -       /mds1     nfs
```

We use the **nfs** file-system type, because the client mounts the remote QFS file system as an NFS file system.

8. In the fifth column of the **/etc/vfstab** file, enter a hyphen (**-**), because the local system is not checking the remote file system for consistency.

```
#File           Device                           Mount
#Device         to      Mount     System fsck  at     Mount
#to Mount       fsck    Point     Type   Pass  Boot   Options
#------------   ------  --------- ------ ----  -----  ----------------
/devices        -       /devices  devfs  -     no      -
...
mds1:/hsm/hqfs1 -       /mds1     nfs    -
```

9. In the sixth column of the **/etc/vfstab** file, enter **yes** to mount the remote file system at boot or **no** to mount it manually, on demand.

   In the example, we enter **yes**:

```
#File           Device                           Mount
#Device         to      Mount     System fsck  at     Mount
#to Mount       fsck    Point     Type   Pass  Boot   Options
#------------   ------  --------- ------ ----  -----  ----------------
/devices        -       /devices  devfs  -     no      -
...
mds1:/hsm/hqfs1 -       /mds1     nfs    -     yes
```

10. In the last column of the **/etc/vfstab** file, enter the **hard** and **intr** NFS mount option to force unlimited, uninterruptable retries or set a specified number of retries by entering the **soft**, **retrans**, and **timeo** mount options with **retrans** set to **120** or more and **timeo** set to **3000** tenths of a second.

    Setting the **hard** retry option or specifying the **soft** option with a sufficiently long timeout and sufficient numbers of retries keeps NFS requests from failing when the requested files reside on removable volumes that cannot be immediately mounted. See the Solaris **mount_nfs** man page for more information on these mount options.

    In the example, we enter the **soft** mount option:

```
#File         Device                           Mount
#Device       to      Mount     System fsck  at     Mount
#to Mount     fsck    Point     Type   Pass  Boot   Options
#------------ ------  --------- ------ ----  -----  ----------------
/devices      -       /devices  devfs  -     no     -
...
mds1:/hqfs1   -       /mds1     nfs    -     yes    soft,retrans=120,timeo=3000
```

11. If you are using NFS 2, set the **rsize** mount parameter to **32768**.

    Accept the default value for other versions of NFS.

    The **rsize** mount parameter sets the read buffer size to **32768** bytes (vs. the default, **8192** bytes). The example shows what an NFS 2 configuration would be like:

```
#File         Device                           Mount
#Device       to      Mount     System fsck  at     Mount
#to Mount     fsck    Point     Type   Pass  Boot   Options
```

```
#------------ ------  --------- ------ ----  -----  ----------------
/devices      -       /devices  devfs  -     no     -
...
mds12:/hqfs2  -       /mds12    nfs    -     yes    ...,rsize=32768
```

12. If you are using NFS 2, set the **wsize** mount parameter to **32768**.

    Accept the default value for other versions of NFS.

    The **wsize** mount parameter sets the write buffer size to the specified number of bytes (by default, **8192** bytes). The example shows what an NFS 2 configuration would be like:

```
#File         Device                         Mount
#Device       to      Mount     System fsck  at     Mount
#to Mount     fsck    Point     Type   Pass  Boot   Options
#------------ ------  --------- ------ ----  -----  ----------------
/devices      -       /devices  devfs  -     no     -
...
mds12:/hqfs2  -       /mds12  nfs    -     yes    ...,wsize=32768
```

13. Save the **/etc/vfstab** file, and exit the editor.

```
#File         Device                         Mount
#Device       to      Mount     System fsck  at     Mount
#to Mount     fsck    Point     Type   Pass  Boot   Options
#------------ ------  --------- ------ ----  -----  ----------------
/devices      -       /devices  devfs  -     no     -
...
mds1:/hqfs1 -       /mds1   nfs    -     yes    soft,retrans=120,timeo=3000
:wq
root@nfsclnt1:~#
```

14. Create a mount point directory for the shared file system.

    In the example, we will mount the shared file system on a directory named **/mds1**:

```
root@nfsclnt1:~# mkdir /mds1
root@nfsclnt1:~#
```

15. Create the mount point specified in the **/etc/vfstab** file, 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 **/mds1** mount-point directory and set permissions to **755** (**-rwxr-xr-x**):

```
root@nfsclnt1:~# mkdir /mds1
root@nfsclnt1:~# chmod 755 /mds1
root@nfsclnt1:~#
```

16. Mount the shared file system:

```
root@nfsclnt1:~# mount /mds1
root@nfsclnt1:~#
```

17. If you plan on using the sideband database feature, go to "Configuring the Reporting Database" on page 10-1.
18. Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

Sharing Oracle HSM File Systems Using SMB/CIFS

SMB makes Oracle HSM accessible to Microsoft Windows hosts and provides interoperability features, such as case-insensitivity, support for DOS attributes, and support for NFSv4 Access Control Lists (ACLs). The Oracle Solaris OS provides a Server Message Block (SMB) protocol server and client implementation that includes support for numerous SMB dialects including NT LM 0.12 and Common Internet File System (CIFS).

Oracle HSM supports Windows Security Identifiers (SIDs). Windows identities no longer need to be explicitly defined using the **`idmap`** service or provided by the Active Directory service.

To configure SMB service with Oracle HSM file systems, carry out the following tasks:

- Review Oracle Solaris SMB configuration and administration documentation.
- Explicitly map Windows identities for the SMB server (optional).
- Configure Oracle HSM file systems for sharing with SMB/CIFS.
- Configure the SMB Server for Windows Active Directory Domains or Workgroups.
- Configure the SMB server for Windows Active Directory Domains or Workgroups.

Review Oracle Solaris SMB Configuration and Administration Documentation

The sections below outline the parts of the SMB configuration process as they apply to Oracle HSM file systems. They are not comprehensive and do not cover all possible scenarios. So review the full instructions for configuring Oracle Solaris SMB servers, integrating the servers into an existing Windows environment, and mounting SMB shares on Solaris systems. Full instructions can be found in the volume *Managing SMB and Windows Interoperability in Oracle Solaris* in the *Oracle Solaris Information Library*.

Explicitly Map Windows Identities for the SMB Server (Optional)

While Oracle HSM now fully supports Windows Security Identifiers (SIDs), explicitly defining the relationships between UNIX identities and SIDs continues to have advantages in some situations. For example, in heterogenous environments where users have both UNIX and Windows identities, you may wish to create explicit mappings using the **`idmap`** service or the Active Directory service. For full SMB and Windows interoperability information, see the product documentation for your version of Oracle Solaris.

Configure Oracle HSM File Systems for Sharing With SMB/CIFS

Oracle HSM file systems that are shared using SMB/CIFS must use the new Access Control List (ACL) implementation adopted by Network File System (NFS) version 4 and introduced in Oracle Solaris 11. Older versions of Solaris and NFS used ACLs that were based on a POSIX-draft specification that is not compatible with the Windows ACL implementation.

New file systems that you create with Oracle HSM use NFS version 4 ACLs by default on Solaris 11. But, if you need to share existing Oracle HSM file systems with SMB/CIFS clients, you must convert the existing POSIX-style ACLs using the appropriate procedure:

- Convert an Oracle HSM unshared file system that uses POSIX-style ACLs.
- Convert an Oracle HSM shared file system that uses POSIX-style ACLs.

Convert an Oracle HSM Unshared File System that Uses POSIX-Style ACLs

Proceed as follows:

1. Log in to the host as **`root`**.

   In the example, we log in to the host **`mds1`**:

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

2. Make sure that the host runs Oracle Solaris 11.1 or higher. Use the command **`uname -r`**.

   ```
   root@mds1:~# uname -r
   5.11
   root@mds1:~#
   ```

3. Unmount the file system using the command **`umount mount-point`**, where **`mount-point`** is the mount point of the Oracle HSM file system.

   See the **`umount_samfs`** man page for further details. In the examples below, the server name is **`mds1`** and the file system is **`shrfs1`**:

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

4. Convert the file system using the **`samfsck -F -A`** *`file-system`* command, where the **`-F`** option specifies a check and repair of the file system, the **`-A`** option specifies conversion of the ACLs, and *`file-system`* is the name of the file system that you need to convert.

   The **`-F`** option is required when the **`-A`** option is specified. If the **`samfsck -F -A`** command returns errors, the process aborts and no ACLs are converted (for full descriptions of these options, see the **`samfsck`** man page).

   ```
   root@mds1:~# samfsck -F -A /hsm/shrfs1
   ```

5. If errors are returned and no ACLs are converted, use the **`samfsck -F -a`** *`file-system`* command to forcibly convert the ACLs.

   The **`-a`** option specifies a forced conversion. The **`-F`** option is required when the **`-a`** option is specified (for full descriptions of these options, see the **`samfsck`** man page).

   ```
   root@mds1:~# samfsck -F -a /hsm/shrfs1
   ```

6. Now, configure the SMB server for Windows Active Directory Domains or Workgroups.

Convert an Oracle HSM Shared File System that Uses POSIX-Style ACLs

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

   In the example, we log in to the metadata server **`mds1`**:

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

2. Make sure that the metadata server runs Oracle Solaris 11.1 or higher. Use the command **`uname -r`**.

   ```
   root@mds1:~# uname -r
   5.11
   root@mds1:~#
   ```

3. Log in to each Oracle HSM client as **`root`**, and make sure that each client runs Oracle Solaris 11.1 or higher.

In the example, we open terminal windows and remotely log in to client hosts **clnt1** and **clnt2** using **ssh** to get the Solaris version from the log-in banner:

```
root@mds1:~# ssh root@clnt1
Password:
Oracle Corporation      SunOS 5.11      11.3    October 2015
root@clnt1:~#

root@mds1:~# ssh root@clnt2
Password:
Oracle Corporation      SunOS 5.11      11.3    October 2015
root@clnt2:~#
```

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

   See the **umount_samfs** man page for further details. In the example, we unmount **shrfs1** from our two clients, **clnt1** and **clnt2**:

```
Oracle Corporation      SunOS 5.11      11.3    October 2015
root@clnt1:~# umount /hsm/shrfs1
root@clnt1:~#

Oracle Corporation      SunOS 5.11      11.3    October 2015
root@clnt2:~# umount /hsm/shrfs1
root@clnt1:~#
```

5. Unmount the Oracle HSM shared file system from the metadata server using the command **umount -o await_clients=***interval* *mount-point*, where *mount-point* is the mount point of the Oracle HSM file system and interval is the delay in seconds specified by 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 metadata server **mds1** while allowing **60** seconds for clients to unmount:

```
root@mds1:~# umount -o await_clients=60 /hsm/shrfs1
```

6. Convert the file system from the POSIX-style ACLs to NFS version 4 ACLs. On the metadata server, use the command **samfsck -F -A** *file-system*, where the **-F** option specifies a check and repair of the file system, the **-A** option specifies conversion of the ACLs, and *file-system* is the name of the file system that you need to convert.

   The **-F** option is required when the **-A** option is specified. If **samfsck -F -A** *file-system* command returns errors, the process aborts and no ACLs are converted (for full descriptions of these options, see the **samfsck** man page). In the example, we convert an Oracle HSM file system named **shrfs1**:

```
root@mds1:~# samfsck -F -A /hsm/shrfs1
```

7. If errors are returned and no ACLs are converted, forcibly convert the ACLs. On the metadata server, use the **samfsck -F -a** *file-system* command.

   The **-a** option specifies a forced conversion. The **-F** option is required when the **-a** option is specified (for full descriptions of these options, see the **samfsck** man page). In the example, we forcibly convert the Oracle HSM file system named **/qfsma**:

```
root@mds1:~# samfsck -F -a /hsm/shrfs1
```

8. Now, configure the SMB server for Windows Active Directory Domains or Workgroups.

Configure the SMB Server for Windows Active Directory Domains or Workgroups

Oracle Solaris SMB services can operate in either of two, mutually exclusive modes: domain or workgroup. Choose one or the other based on your environment and authentication needs:

- If you need to give Active Directory domain users access to the Solaris SMB service, configure the SMB server in Domain Mode.
- If you need to give local Solaris users access to the SMB service and either do not have Active Directory domains or do not need to give Active Directory domain users access to the service, configure the SMB server in Workgroup Mode.

Configure the SMB Server in Domain Mode

1. Contact the Windows Active Directory administrator and obtain the following information:
   - the name of the authenticated Active Directory user account that you need to use when joining the Active Directory domain
   - the organizational unit that you need to use in place of the default **Computers** container for the account (if any)
   - the fully qualified LDAP/DNS domain name for the domain where the Oracle HSM file system is to be shared.
2. Log in to the host of the Oracle HSM file system that you want to configure as an SMB/CIFS share. Log in as **root**.

   If the file system is an Oracle HSM shared file system, log in to the metadata server for the file system. In the examples below, the server name is **mds1**.

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

3. Open-source Samba and SMB servers cannot be used together on a single Oracle Solaris system. So see if the Samba service is running. Pipe the output of the services status command **svcs** into **grep** and the regular expression **samba**.

   In the example, the output of the **svcs** command contains a match for the regular expression, so the SMB service is running:

   ```
   root@mds1:~# svcs | grep samba
   legacy_run     Nov_03   lrc:/etc/rc3_d/S90samba
   ```

4. If the Samba service (**svc:/network/samba**) is running, disable it along with the Windows Internet Naming Service/WINS (**svc:/network/wins**), if running. Use the command **svcadm disable**.

   ```
   root@mds1:~# svcadm disable svc:/network/samba
   root@mds1:~# svcadm disable svc:/network/wins
   ```

5. Now use the **svcadm enable -r smb/server** command to start the SMB server and any services on which it depends.

   ```
   root@mds1:~# svcadm enable -r smb/server
   ```

6. Make sure that the system clock on the Oracle HSM host is within five minutes of the system clock of the Microsoft Windows domain controller:

   - If the Windows domain controller uses Network Time Protocol (NTP) servers, configure the Oracle HSM host to use the same servers. Create an **`/etc/inet/ntpclient.conf`** file on the Oracle HSM host and start the **`ntpd`** daemon using the command **`svcadm enable ntp`** (see the **`ntpd`** man page and your Oracle Solaris administration documentation for full information).
   - Otherwise, synchronize the Oracle HSM host with the domain controller by running the command **`ntpdate`** *`domain-controller-name`* (see the **`ntpdate`** man page for details) or manually set the system clock on the Oracle HSM host to the time displayed by the domain controller's system clock.

7. Join the Windows domain using the command **`smbadm join -u`** *`username`* **`-o`** *`organizational-unit domain-name`*, where *`username`* is the name of the user account specified by the Active Directory administrator, the optional *`organizational-unit`* is the account container specified (if any), and *`domain-name`* is the specified, fully qualified, LDAP or DNS domain name.

   In the example, we join the Windows domain **`this.example.com`** using the user account

```
root@mds1:~# smbadm join -u admin -o smbsharing this.example.com
```

8. Now share the Oracle HSM file system as an SMB/CIFS share.

Configure the SMB Server in Workgroup Mode

1. Contact the Windows network administrator and obtain the name of the Windows workgroup that the host of the Oracle HSM file system should join.

   The default workgroup is named **`WORKGROUP`**.

2. Log in to the host of the Oracle HSM file system. Log in as **`root`**.

   If the file system is an Oracle HSM shared file system, log in to the metadata server for the file system. In the examples below, the server name is **`mds1`**.

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

3. Open-source Samba and SMB servers cannot be used together on a single Oracle Solaris system. So see if Samba service is running. Pipe the output of the **`svcs`** services status command into **`grep`** and the regular expression **`samba`**.

   In the example, the output of the **`svcs`** command contains a match for the regular expression, so the SMB service is running:

```
root@mds1:~# svcs | grep samba
legacy_run     Nov_03   lrc:/etc/rc3_d/S90samba
```

4. If the Samba service (**`svc:/network/samba`**) is running, disable it along with the Windows Internet Naming Service/WINS (**`svc:/network/wins`**) services, if running. Use the command **`svcadm disable`**.

   Samba and SMB servers cannot be used together on a single Oracle Solaris system.

```
root@mds1:~# svcadm disable svc:/network/samba
root@mds1:~# svcadm disable svc:/network/wins
```

5. Now use the command **`svcadm enable -r smb/server`** to start the SMB server and any services on which it depends.

```
root@mds1:~# svcadm enable -r smb/server
```

6. Join the workgroup. Use the command **smbadm join** with the **-w** (workgroup) switch and the name of the workgroup specified by the Windows network administrator.

   In the example, the specified workgroup is named **crossplatform**.

   ```
   root@mds1:~# smbadm join -w crossplatform
   ```

7. Configure the Oracle HSM host for encryption of SMB passwords. Open the **/etc/pam.d/other** file in a text editor, add the command line **password required pam_smb_passwd.so.1 nowarn**, and save the file.

   In the example, we use the **vi** editor:

   ```
   root@mds1:~# vi /etc/pam.d/other
   # Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
   #
   # PAM configuration
   #
   # Default definitions for Authentication management
   # Used when service name is not explicitly mentioned for authentication
   #
   auth definitivepam_user_policy.so.1
   ...
   password requiredpam_authtok_store.so.1
   password required pam_smb_passwd.so.1 nowarn
   :wq
   root@mds1:~#
   ```

   See the **pam_smb_passwd** man page for further details.

8. Once the **pam_smb_passwd** module has been installed, use the command **passwd** *local-username* to generate an encrypted version of the password for user *local-username* so that the SMB server can log in to the Windows workgroup.

   The SMB server cannot authenticate users using the same encrypted versions of passwords that the Solaris operating system uses. In the example, we generate an encrypted SMB password for the user **smbsamqfs**:

   ```
   root@mds1:~# passwd smbsamqfs
   ```

9. Now share the Oracle HSM file system as an SMB/CIFS share.

Share the Oracle HSM File System as an SMB/CIFS Share

Share the Oracle HSM file system using the procedures described in the administration documentation for your version of the Oracle Solaris operating system. The steps below summarize the procedure for Solaris 11.1:

1. Log in to the host of the Oracle HSM file system that you want to configure as an SMB/CIFS share. Log in as **root**.

   If the file system is an Oracle HSM shared file system, log in to the metadata server for the file system. In the examples below, the server name is **mds1**.

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

2. Configure the share. Use the command **share -F smb -o** *specific-options sharepath sharename*, where the **-F** switch specifies the **smb** sharing protocol, *sharepath* is the path to the shared resource, and *sharename* is the name that you

want to use for the share. The value of the optional `-o` parameter, *`sharing-options`*, is a comma-delimited list that includes any of the following:

- **`abe=[true|false]`**

  When the access-based enumeration (ABE) policy for a share is **`true`**, directory entries to which the requesting user has no access are omitted from directory listings returned to the client.

- **`ad-container=cn=`***`user`***`,ou=`***`organization`***`,dc=`***`domain-dns`*

  The Active Directory container limits the share access to domain objects specified by the Lightweight Directory Access Protocol (LDAP) relative distinguished name (RDN) attribute values: **`cn`** (user object class), **`ou`** (organizational unit object class), and **`dc`** (domain DNS object class).

  For full information on using Active Directory containers with SMB/CIFS, consult *Internet Engineering Task Force Request For Comment (RFC) 2253* and your Microsoft Windows directory services documentation.

- **`catia=[true|false]`**

  When CATIA character substitution is **`true`**, any characters in a CATIA version 4 file name that are illegal in Windows are replaced by legal equivalents. See the **`share_smb`** man page for a list of substitutions.

- **`csc=[manual|auto|vdo|disabled]`**

  A client-side caching (**`csc`**) policy controls client-side caching of files for offline use. The **`manual`** policy lets clients cache files when requested by users, but disables automatic, file-by-file reintegration (this is the default). The **`auto`** policy lets clients automatically cache files and enables automatic file-by-file reintegration. The **`vdo`** policy lets clients automatically cache files for offline use, enables file-by-file reintegration, and lets clients work from the local cache even while offline. The **`disabled`** policy does not allow client-side caching.

- **`dfsroot=[true|false]`**

  In a Microsoft Distributed File System (DFS), a root share (**`dfsroot=true`**) is the share that organizes a group of widely distributed shared folders into a single DFS file system that can be more easily managed. For full information, see your Microsoft Windows Server documentation.

- **`guestok=[true|false]`**

  When the **`guestok`** policy is **`true`**, the locally defined **`guest`** account can access the share. When it is **`false`** or left undefined (the default), the **`guest`** account cannot access the share. This policy lets you map the Windows **`Guest`** user to a locally defined, UNIX user name, such as **`guest`** or **`nobody`**:

  ```
  # idmap add winname:Guest unixuser:guest
  ```

  The locally defined account can then be authenticated against a password stored in **`/var/smb/smbpasswd`**, if desired. See the **`idmap`** man page for more information.

- **`rw=[*|[[-]`***`criterion`***`][:[-]`***`criterion`***`]`**...

  The **`rw`** policy grants or denies access to any client that matches the supplied access list.

  Access lists contain either a single asterisk (**`*`**) meaning *all* or a colon-delimited list of client access criteria, where each *`criterion`* consists of an optional minus sign (**`-`**), meaning *deny*, followed by a host name, a network group, a full LDAP or DNS domain name, and/or the symbol **@** plus all or part of an IP address or domain name. Access lists are evaluated left to right until the client satisfies one of the criteria. See the **share_smb** man page for further details.

- **ro=[*|[[-]***criterion***][:[-]***criterion*]...

  The **ro** policy grants or denies read-only access to any client that matches the access list.

- **none=[*|[[-]***criterion***][:[-]***criterion*]...

  The **none** policy denies access to any client that matches the access list. If the access list is an asterisk (*****), the **ro** and **rw** policies can override the **none** policy.

In the example, we share the **shrfs1** file system read/write with clients **smbclnt1** and **smbclnt2** and read-only with **smbclient3**:

```
root@mds1:~# share -F smb -o rw=smbclnt1:smbclnt2,ro=smbclient3 /hsm/shrfs1
```

When you enter the command, the system automatically restarts the SMB server daemon, **smbd**.

3. Check the sharing parameters. Use the command **share -F smb -A**.

   In the example, the command output shows that we have correctly configured the share:

```
root@mds1:~# share -F smb /hsm/shrfs1
sec=sys,rw=smbclnt1:smbclnt2,ro=smbclient3
root@mds1:~#
```

4. If you plan on using the sideband database feature, go to "Configuring the Reporting Database" on page 10-1.
5. Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

9

Preparing High-Availability Solutions

Oracle Hierarchical Storage Manager and StorageTek QFS Software high-availability configurations are designed to maintain uninterrupted file-system and archiving services. In a high-availability solution, Oracle Hierarchical Storage Manager or QFS software is integrated with Oracle Solaris Cluster software, redundant hardware, and redundant communications. So, if a host system or component fails or is taken out of service by administrators, Oracle HSM services automatically fail over to an alternative host that users and applications can access. High-availability configurations thus minimize downtime due to equipment and system failure.

High-availability configurations are complex, however, and must be carefully designed and deployed to prevent unforeseen interactions and, possibly, data corruption. So this chapter starts with an explanation of the supported configurations. Study this section and select the configuration that best addresses your availability requirements. Subsequent sections can then explain how you set up your selected configuration.

Note that you cannot mix hardware architectures in a shared Oracle Solaris Cluster configuration. All of the nodes must use either the SPARC architecture, the x86-64 architecture (Solaris 11.1 only), or the 32-bit x86 architecture (Solaris 10 and earlier).

Understanding the Supported High-Availability Configurations

In Oracle HSM high-availability configurations, Solaris Cluster merely initiates failover. It does not involve itself in the inner workings of the shared file system. If the cluster detects the failure of an active Oracle HSM metadata server on one of its nodes, it uses the SUNW.qfs data-service to activate the potential metadata server on the surviving node. Thereafter, Oracle HSM software controls the file system and client behavior.

In clustered, multihost solutions, interactions between the file systems, applications, operating systems, clustering software, and storage have to be carefully controlled to insure the integrity of stored data. To minimize complexity and potential risk, supported high-availability Oracle HSM configurations are thus tailored to four specific sets of deployment requirements:

- HA-QFS, a highly available QFS unshared, standalone file system
- HA-COTC, a QFS shared file system with highly available metadata servers
- HA-HSM, a highly available QFS shared, archiving file system
- SC-RAC, a highly available QFS shared file system for Oracle RAC.

HA-QFS, a High-Availability QFS Unshared, Standalone File-System Configuration

The High Availability QFS (HA-QFS) configuration insures that a locally mounted QFS file system remains accessible following failover to a standby host. The QFS file system is configured on both nodes of a two-node cluster but mounted on only one at any given time. If the host node fails, Solaris Cluster and **`SUNW.HAStoragePlus`** data-service software automatically initiate failover and re-mount the local QFS file system on the healthy node. File-system users and applications access data using network file sharing, with the active cluster node acting as a file server. The HA-QFS configuration supports Network File System (NFS) shares, high availability NFS (HA-NFS) shares, Server Message Block/Common Internet File System (SMB/CIFS, SAMBA) shares, and high availability SMB/CIFS (HA-SAMBA) shares.

Before proceeding, make sure that you have correctly configured the storage for use with **`SUNW.HAStoragePlus`**. See "Configure Solaris Cluster Nodes for Multipath I/O" on page 3-16", "Configure Linux Clients for Multipath I/O" on page 3-17, and the documentation for the Solaris Cluster **`SUNW.HAStoragePlus`** data service.

Then, for implementation instructions, see "High-Availability QFS Unshared File Systems" on page 9-3.

HA-COTC, a QFS Shared File System with High-Availability Metadata Servers

The High Availability-Clients Outside the Cluster (HA-COTC) configuration insures that the metadata server of a shared QFS file system remains accessible to clients following failover to a standby host. The shared file system is configured on both nodes of a two-node cluster and on file-system clients that are *not* part of the cluster. For failover purposes, the cluster nodes are configured as active and potential QFS metadata servers. While the active metadata server node remains healthy, the potential metadata server remains on standby and never performs I/O as a QFS client.

In this configuration, metadata server hosts access the data devices as a defined resource within the cluster. Client hosts access the devices as specified by the QFS file-system configuration. If the active metadata server fails, the cluster initiates failover by activating the potential metadata server. The QFS metadata server and clients then re-establish communications and complete the failover.

HA-COTC configurations must use high performance **`ma`** file systems with physically separate **`mm`** metadata devices and **`mr`** data devices. The Solaris Cluster software fences off devices that are in use within the cluster. So clients outside the cluster would be unable to access data stored in a standard QFS **`ms`** file system, where data and metadata reside on the same **`md`** disk devices. In an HA-COTC configuration, Solaris Cluster fences off the **`mm`** metadata devices of the **`ma`** file system, while leaving **`mr`** data devices unfenced and accessible to clients.

You can use standard Network File System (NFS) or SMB/CIFS (SAMBA) to share HA-COTC file systems with additional clients. But HA-NFS and HA-SAMBA are *not* supported.

For implementation instructions, see "High-Availability QFS Shared File Systems, Clients Outside the Cluster" on page 9-6.

HA-HSM, a High-Availability, Archiving, QFS Shared File-System Configuration

The High-Availability Oracle Hierarchical Storage Manager (HA-HSM) configuration maintains the availability of an archiving 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.

If the active Oracle HSM metadata server node fails, the clustering software automatically activates the potential metadata server node and initiates failover. Since the QFS file system is shared and already mounted on all nodes, access to data and metadata remains uninterrupted.

Clients access data via Network File System (HA-NFS), NFS, or SMB/CIFS shares, with the active cluster node acting as a file server.

For implementation instructions, see "High-Availability Oracle HSM Shared Archiving File Systems" on page 9-23.

SC-RAC, a High-Availability QFS Shared File-System Configuration for Oracle RAC

The Solaris Cluster-Oracle Real Application Cluster (SC-RAC) configuration supports high-availability database solutions that use QFS file systems. 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 in the cluster. Solaris Cluster software manages the cluster as a resource of type **`SUNW.qfs`**. 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.

For implementation instructions, see "High-Availability QFS Shared File Systems and Oracle RAC" on page 9-43.

High-Availability QFS Unshared File Systems

To configure a high-availability QFS (HA-QFS) file system, you set up two identical hosts in a two-node, Solaris Cluster, managed as a resource of type **`SUNW.HAStoragePlus`**. You then configure a QFS unshared file system on both nodes. Only one node mounts the file system at any given time. But, if one node fails, the clustering software automatically initiates fail over and re-mounts the file system on the surviving node.

To set up a high-availability QFS (HA-QFS) file system, proceed as follows:

- Make sure that you have correctly configured the storage for use with **`SUNW.HAStoragePlus`**. See "Configure Solaris Cluster Nodes for Multipath I/O" on page 3-16", "Configure Linux Clients for Multipath I/O" on page 3-17, and the documentation for the Solaris Cluster **`SUNW.HAStoragePlus`** data service.
- Create unshared QFS file systems on both Solaris Cluster nodes.
- Configure the high-availability QFS file system.
- If required, configure high-availability, HA-NFS and/or HA-SAMBA.

Create Unshared QFS File Systems on Both Cluster Nodes

1. Log in to one of the cluster nodes as **`root`**.

In the example, the hosts are **qfs1mds-node1** and **qfs1mds-node2**. We log in to the host **qfs1mds-node1**:

```
root@qfs1mds-node1:~#
```

2. Configure the desired QFS file system on the host, but do not mount it.

   Configure the file system using the instructions in "Configure a General-Purpose `ms` File System" on page 6-1 or "Configure a High-Performance `ma` File System" on page 6-5. The HA-QFS configuration does not support QFS shared file systems.

3. Log in to the remaining cluster node as **root**.

   In the example, we log in to the host **qfs1mds-node2** using **ssh**:

```
root@qfs1mds-node1:~# ssh root@qfs1mds-node2
Password:
root@qfs1mds-node2:~#
```

4. Configure an identical QFS file system on the second node.
5. Next, configure the high-availability QFS file system.

Configure the High-Availability QFS File System

Proceed as follows:

1. Log in to one of the cluster nodes as **root**.

   In the example, the hosts are **qfs1mds-node1** and **qfs1mds-node2**. We log in to the host **qfs1mds-node1**:

```
root@qfs1mds-node1:~#
```

2. See if the **SUNW.HAStoragePlus** resource type has been registered with the cluster. Use the command **clresourcetype show**.

   HAStoragePlus is the Solaris Cluster resource type that defines and manages dependencies between disk device groups, cluster file systems, and local file systems. It coordinates start-up of data services following failovers, so that all required components are ready when the service tries to restart. See the **SUNW.HAStoragePlus** man page for further details.

   In the example, the **SUNW.HAStoragePlus** has already been registered:

```
root@qfs1mds-node1:~# clresourcetype show
=== Registered Resource Types ===
...
Resource Type:                  SUNW.HAStoragePlus:11
  RT_description:               HA Storage Plus
  RT_version:                   11
...
root@qfs1mds-node1:~#
```

3. If the **SUNW.HAStoragePlus** resource type has not been registered, register it now. Use the command **clresourcetype register SUNW.HAStoragePlus**.

```
root@qfs1mds-node1:~# clresourcetype register SUNW.HAStoragePlus
root@qfs1mds-node1:~#
```

4. If registration fails because the registration file cannot be found, make a symbolic link from the directory where Solaris Cluster keeps resource registration files to the directory where Oracle HSM keeps registration information. Change to the

directory **/usr/cluster/lib/rgm/rtreg/**, and create the link with the command **ln -s SUNW.HAStoragePlus /opt/SUNWsamfs/sc/etc/SUNW.HAStoragePlus/**.

Registration failed because you did not install Oracle Solaris Cluster software before installing Oracle HSM software. When Oracle HSM detects Solaris Cluster during installation, it automatically creates the required link. If you install Oracle HSM first, you need to create the link manually.

```
root@hsm1mds-node1:~# cd /usr/cluster/lib/rgm/rtreg/
root@hsm1mds-node1:~# ln -s SUNW.HAStoragePlus
/opt/SUNWsamfs/sc/etc/SUNW.HAStoragePlus
root@hsm1mds-node1:~#
```

5. Create a new Solaris Cluster resource of type **SUNW.HAStoragePlus** and a new resource group to contain it. Use the command **/usr/global/bin/clresource create -g** *resource-group* **-t SUNW.HAStoragePlus -x FilesystemMountPoints=/global/***mount-point* **-x FilesystemCheckCommand=/bin/true** *QFS-resource*, where:

   - *resource-group* is the name that you have chosen for the file-system resource group.
   - *mount-point* is the directory where the QFS file system is mounted.
   - *QFS-resource* is the name that you have chosen for the **SUNW.HAStoragePlus** resource.

   In the example, we create the resource group **qfsrg** with the mount-point directory **/global/qfs1** and the **SUNW.HAStoragePlus** resource **haqfs**:

```
root@qfs1mds-node1:~# clresource create -g qfsrg -t SUNW.HAStoragePlus
-x FilesystemMountPoints=/global/hsmqfs1/qfs1
-x FilesystemCheckCommand=/bin/true haqfs
root@qfs1mds-node1:~#
```

6. Display the nodes in the cluster. Use the command **clresourcegroup status**.

   In the example, the QFS file-system host nodes are **qfs1mds-1** and **qfs1mds-2**. Node **qfs1mds-1** is **Online**, so it is the primary node that mounts the file system and hosts the **qfsrg** resource group:

```
root@qfs1mds-node1:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name    Node Name    Suspended   Status
----------    ---------    ---------   ------
qfsrg         qfs1mds-1    No          Online
              qfs1mds-2    No          Offline
root@qfs1mds-node1:~#
```

7. Make sure that the resource group fails over correctly by moving the resource group to the secondary node. Use the Solaris Cluster command **clresourcegroup switch -n** *node2 group-name*, where *node2* is the name of the secondary node and *group-name* is the name that you have chosen for the HA-QFS resource group. Then use **clresourcegroup status** to check the result.

   In the example, we move the **haqfs** resource group to **qfs1mds-node2** and confirm that the resource group comes online on the specified node:

```
root@qfs1mds-node1:~# clresourcegroup switch -n qfs1mds-node2 qfsrg
root@qfs1mds-node1:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name    Node Name    Suspended   Status
----------    ---------    ---------   ------
```

```
qfsrg        qfs1mds-1    No          Offline
             qfs1mds-2    No          Online
root@qfs1mds-node1:~#
```

8. Move the resource group back to the primary node. Use the Solaris Cluster command **`clresourcegroup switch -n`** *`node1 group-name`*, where *`node1`* is the name of the primary node and *`group-name`* is the name that you have chosen for the HA-QFS resource group. Then use **`clresourcegroup status`** to check the result.

   In the example, we successfully move the **`qfsrg`** resource group back to **`qfs1mds-node1`**:

```
root@qfs1mds-node1:~# clresourcegroup switch -n qfs1mds-node1 qfsrg
root@qfs1mds-node1:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name  Node Name      Suspended   Status
----------  -------------  ---------   ------
qfsrg       qfs1mds-node1  No          Online
            qfs1mds-node2  No          Offline
root@qfs1mds-node1:~#
```

9. If you need to configure High-Availability Network File System (HA-NFS) sharing, do so now. For instructions, see the *Oracle Solaris Cluster Data Service for Network File System (NFS) Guide* that is included in the *Oracle Solaris Cluster* online documentation library.
10. If you need to share the HA-QFS file system with HA-NFS or HA-SAMBA, go to "Sharing HA-HSM or HA-QFS Configurations with HA-NFS or HA-SAMBA" on page 9-63.
11. If you plan on using the sideband database feature, go to "Configuring the Reporting Database" on page 10-1.
12. Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

High-Availability QFS Shared File Systems, Clients Outside the Cluster

The High Availability-Clients Outside the Cluster (HA-COTC) configuration is a non-archiving, QFS shared file system that hosts the crucial metadata server (MDS) on the nodes of a high-availability cluster managed by Solaris Cluster software. The client hosts remain outside the cluster and outside the control of the Solaris Cluster software.

In this configuration, metadata server hosts access the data devices as a defined cluster resource of type **`SUNW.qfs`**. Client hosts access the devices as specified by the QFS file-system configuration. If the active metadata server fails, the cluster initiates failover by activating the potential metadata server. The QFS metadata server and clients then re-establish communications and complete the failover.

To configure an HA-COTC file system, carry out the tasks below:

- Create a QFS hosts file on both HA-COTC cluster nodes
- Create local hosts files on the QFS server nodes and on clients outside the cluster.
- Configure an active QFS metadata server on the primary HA-COTC cluster node.
- Configure a potential QFS metadata server on the secondary HA-COTC cluster node.
- Configure failover of the HA-COTC metadata server.

- Configure hosts outside the HA-COTC cluster as QFS shared file system clients.
- If required, configure Network File System (NFS) and/or SMB/CIFS (SAMBA) shares, as described in "Accessing File Systems from Multiple Hosts Using NFS and SMB/CIFS" on page 8-32.

  Note that the HA-COTC configuration does *not* support *high availability* NFS (HA-NFS) or *high availability* SMB/CIFS (HA-SAMBA).

Create a QFS Shared File System Hosts File on Both HA-COTC Cluster Nodes

In a QFS shared file system, you must configure a hosts file on the metadata servers, so that all hosts can access the metadata for the 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. Log in to the primary node of the HA-COTC cluster as **root**.

   In the example, the hosts are **qfs1mds-1** and **qfs1mds-2**. We log in to the host **qfs1mds-1**:

   ```
   root@qfs1mds-1:~#
   ```

2. Display the cluster configuration. Use the **/usr/global/bin/cluster show** command. Locate the record for each **Node Name**, and then note the **privatehostname**, the **Transport Adapter** name, and the **ip_address** property of each network adapter.

   The outputs of the commands can be quite lengthy, so, in the examples below, long displays are abbreviated using ellipsis (...) marks.

   In the examples, each node has two network interfaces, **qfe3** and **hme0**:

   - The **hme0** adapters have IP addresses on the private network that the cluster uses for internal communication between nodes. The Solaris Cluster software assigns a private hostname corresponding to each private address.

     By default, the private hostname of the primary node is **clusternode1-priv**, and the private hostname of the secondary node is **clusternode2-priv**.

   - The **qfe3** adapters have public IP addresses and hostnames—**qfs1mds-1** and **qfs1mds-2**—that the cluster uses for data transport.

   ```
   root@qfs1mds-1:~# cluster show
   ...
     === Cluster Nodes ===
     Node Name:                                    qfs1mds-1...
       privatehostname:                               clusternode1-priv...
       Transport Adapter List:                        qfe3, hme0...
       Transport Adapter:                          qfe3...
         Adapter Property(ip_address):                172.16.0.12...
       Transport Adapter:                          hme0...
         Adapter Property(ip_address):                10.0.0.129...
     Node Name:                                    qfs1mds-2...
       privatehostname:                               clusternode2-priv...
       Transport Adapter List:                        qfe3, hme0...
         Adapter Property(ip_address):                172.16.0.13...
       Transport Adapter:                          hme0
         Adapter Property(ip_address):                10.0.0.122
   ```

3. Using a text editor, create the file **/etc/opt/SUNWsamfs/hosts.***family-set-name* on the metadata server, where *family-set-name* is the name of the family-set name of the file-system.

   In the example, we create the file **hosts.qfs1** using the **vi** text editor. We add some optional headings to show the columns in the hosts table, starting each line with a hash sign (#), indicating a comment:

```
root@qfs1mds-1:~# vi /etc/opt/SUNWsamfs/hosts.qfs1
# /etc/opt/SUNWsamfs/hosts.qfs1
#                                             Server   On/  Additional
#Host Name     Network Interface              Ordinal  Off  Parameters
#------------  -----------------------------  -------  ---  ----------
```

4. In the first column of the table, enter the hostnames of the primary and secondary metadata server nodes followed by some spaces. Place each entry on a separate line.

   In a hosts file, the lines are rows (records) and spaces are column (field) separators. In the example, the **Host Name** column of the first two rows contains the values **qfs1mds-1** and **qfs1mds-2**, the hostnames of the cluster nodes that host the metadata servers for the file system:

```
#                                             Server   On/  Additional
#Host Name     Network Interface              Ordinal  Off  Parameters
#------------  -----------------------------  -------  ---  ----------
qfs1mds-1
qfs1mds-2
```

5. In the second column of each line, start supplying **Network Interface** information for host **Host Name**. Enter each HA-COTC server host's private hostname or private network address, followed by a comma.

   The HA-COTC server nodes use the private hostnames for server-to-server communications within the high-availability cluster. In the example, we use the private hostnames **clusternode1-priv** and **clusternode2-priv**, which are the default names assigned by the Solaris Cluster software:

```
#                                             Server   On/  Additional
#Host Name     Network Interface              Ordinal  Off  Parameters
#------------  -----------------------------  -------  ---  ----------
qfs1mds-1   clusternode1-priv,
qfs1mds-2   clusternode2-priv,
```

6. Following the comma in the second column of each line, enter the public network hostname for the corresponding active or potential metadata server, followed by spaces.

   In the example, we enter the host names shown in the first column, **qfs1mds-1** and **qfs1mds-2**:

```
#                                             Server   On/  Additional
#Host Name     Network Interface              Ordinal  Off  Parameters
#------------  -----------------------------  -------  ---  ----------
qfs1mds-1   clusternode1-priv,qfs1mds-1
qfs1mds-2   clusternode2-priv,qfs1mds-2
```

7. In the third column of each line, enter the ordinal number of the server (**1** for the active metadata server, and **2** for the potential metadata server), followed by spaces.

In this example, there is only one metadata server, the primary node, **qfs1mds-1**, is the active metadata server, so it is ordinal **1** and the secondary node, **qfs1mds-2**, is ordinal **2**:

```
#                                              Server   On/  Additional
#Host Name     Network Interface               Ordinal  Off  Parameters
#------------  -----------------------------  -------  ---  ----------
qfs1mds-1      clusternode1-priv,qfs1mds-1     1
qfs1mds-2      clusternode2-priv,qfs1mds-2     2
```

8. In the fourth column of each line, enter **0** (zero), followed by spaces.

   A **0** (zero), **-** (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 **samsharefs** man page).

```
#                                              Server   On/  Additional
#Host Name     Network Interface               Ordinal  Off  Parameters
#------------  -----------------------------  -------  ---  ----------
qfs1mds-1      clusternode1-priv,qfs1mds-1     1        0
qfs1mds-2      clusternode2-priv,qfs1mds-2     2        0
```

9. In the fifth column of the line for the primary node, enter the keyword **server**.

   The **server** keyword identifies the default, active metadata server:

```
#                                              Server   On/  Additional
#Host Name     Network Interface               Ordinal  Off  Parameters
#------------  -----------------------------  -------  ---  ----------
qfs1mds-1      clusternode1-priv,qfs1mds-1     1        0    server
qfs1mds-2      clusternode2-priv,qfs1mds-2     2        0
```

10. Add a line for each client host, setting the **Server Ordinal** value to **0**. Then save the file and close the editor.

    A server ordinal of **0** identifies the host as a client rather than a server. HA-COTC clients are not members of the cluster and thus communicate only over the cluster's public, data network. They only have public IP addresses. In the example, we add two clients, **qfs1client1** and **qfs1client2**, using their public IP addresses, **172.16.0.133** and **172.16.0.147** rather than hostnames:

```
#                                              Server   On/  Additional
#Host Name     Network Interface               Ordinal  Off  Parameters
#------------  -----------------------------  -------  ---  ----------
qfs1mds-1      clusternode1-priv,qfs1mds-1     1        0    server
qfs1mds-2      clusternode2-priv,qfs1mds-2     2        0
qfs1client1    172.16.0.133                    0        0
qfs1client2    172.16.0.147                    0        0
:wq
root@qfs1mds-1:~#
```

11. Place a copy of the global **/etc/opt/SUNWsamfs/hosts.***family-set-name* file on the QFS potential metadata server (the second HA-COTC Cluster node).

12. Now, create local hosts files on the QFS server nodes and on clients outside the cluster.

Create Local Hosts Files on the QFS Server Nodes and on Clients Outside the Cluster

In a high-availability configuration that shares a file system with clients outside the cluster, you need to insure that the clients only communicate with the file system servers using the public, data network defined by the Solaris Cluster software. You do this by using specially configured QFS local hosts files to selectively route network traffic between clients and multiple network interfaces on the server.

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. 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 different arrangements in each file, you can thus control the interfaces used by different hosts.

To configure local hosts files, use the procedure outlined below:

1. Log in to the primary node of the HA-COTC cluster as **root**.

   In the example, the hosts are **qfs1mds-1** and **qfs1mds-2**. We log in to the host **qfs1mds-1**:

   ```
   root@qfs1mds-1:~#
   ```

2. Create a local hosts file on each of the active and potential metadata servers. Use the path and file name **/etc/opt/SUNWsamfs/hosts.***family-set-name***.local**, where *family-set-name* is the equipment identifier for the shared file system. *Only include interfaces for the networks that you want the active and potential servers to use.*

   In our example, we want the active and potential metadata servers to communicate with each other over the private network and with clients via the public network. So the local hosts file on the active and potential servers, **hosts.qfs1.local**, lists only cluster private addresses for the active and potential servers:

   ```
   root@qfs1mds-1:~# vi /etc/opt/SUNWsamfs/hosts.qfs1.local
   # /etc/opt/SUNWsamfs/hosts.qfs1.local
   #                                             Server   On/  Additional
   #Host Name     Network Interface             Ordinal  Off  Parameters
   #------------  ----------------------------  -------  ---  ----------
   qfs1mds-1      clusternode1-priv             1        0    server
   qfs1mds-2      clusternode2-priv             2        0
   qfs1client1    172.16.0.133                  0        0
   qfs1client2    172.16.0.147                  0        0
   :wq
   root@qfs1mds-1:~# ssh root@qfs1mds-2
   Password:
   root@qfs1mds-2:~# vi /etc/opt/SUNWsamfs/hosts.qfs1.local
   # /etc/opt/SUNWsamfs/hosts.qfs1.local
   #                                             Server   On/  Additional
   #Host Name     Network Interface             Ordinal  Off  Parameters
   #------------  ----------------------------  -------  ---  ----------
   qfs1mds-1      clusternode1-priv             1        0    server
   qfs1mds-2      clusternode2-priv             2        0
   qfs1client1    172.16.0.133                  0        0
   qfs1client2    172.16.0.147                  0        0
   :wq
   ```

```
root@qfs1mds-2:~# exit
root@qfs1mds-1:~#
```

3. Using a text editor, create a local hosts file on each of the clients. Use the path and file name **`/etc/opt/SUNWsamfs/hosts.`**`family-set-name`**`.local`**, where `family-set-name` is the equipment identifier for the shared file system. *Only include interfaces for the networks that you want the clients to use*. Then save the file and close the editor.

   In our example, we use the **`vi`** editor. We want the clients to communicate only with the servers and only via the public, data network. So the file includes only the logical host name for the active metadata server, **`qfs1mds`**.The Solaris Cluster software will route requests for **`qfs1mds`** to whichever server node is active:

```
root@qfs1mds-1:~# ssh root@qfsclient1
Password:
root@qfs1client-1:~# vi /etc/opt/SUNWsamfs/hosts.qfs1.local
# /etc/opt/SUNWsamfs/hosts.qfs1.local
#                                              Server   On/  Additional
#Host Name     Network Interface               Ordinal  Off  Parameters
#------------  -----------------------------  -------  ---  ----------
qfs1mds        qfs1mds                         1        0    server
:wq
root@qfs1client-1:~# exit
root@qfs1mds-1:~# ssh root@qfs1client2
Password:
root@qfs1client-2:~# vi /etc/opt/SUNWsamfs/hosts.qfs1.local
# /etc/opt/SUNWsamfs/hosts.qfs1.local
#                                              Server   On/  Additional
#Host Name     Network Interface               Ordinal  Off  Parameters
#------------  -----------------------------  -------  ---  ----------
qfs1mds        qfs1mds                         1        0    server
:wq
root@qfs1client-2:~# exit
root@qfs1mds-1:~#
```

4. Next, configure an active QFS metadata server on the primary HA-COTC cluster node.

Configure an Active QFS Metadata Server on the Primary HA-COTC Cluster Node

To configure the active metadata server, carry out the following tasks:

- Create a high-performance QFS file system on the primary HA-COTC cluster node.
- Exclude QFS data devices from cluster control.
- Mount the QFS file system on the primary HA-COTC cluster node.

Create a High-Performance QFS File System on the Primary HA-COTC Cluster Node

1. Select the cluster node that will serve as both the primary node for the HA-COTC cluster and the active metadata server for the QFS shared file system. Log in as **`root`**.

   In the example, **`qfs1mds-1`** is the primary node and active metadata server:

```
root@qfs1mds-1:~#
```

2. Select the global storage devices that will be used for the QFS file system. Use the Solaris Cluster command **/usr/global/bin/cldevice list -v**.

   Solaris Cluster software assigns unique Device Identifiers (DIDs) to all devices that attach to the cluster nodes. *Global* devices are accessible from all nodes in the cluster, while *local* devices are accessible only from the hosts that mount them. Global devices remain accessible following failover. Local devices do not.

   In the example, note that devices **d1**, **d2**, **d7**, and **d8** are not accessible from both nodes. So we select from devices **d3**, **d4**, and **d5** when configuring the high-availability QFS shared file system:

```
root@qfs1mds-1:~# cldevice list -v
DID Device          Full Device Path
----------          ----------------
d1                  qfs1mds-1:/dev/rdsk/c0t0d0
d2                  qfs1mds-1:/dev/rdsk/c0t6d0
d3                  qfs1mds-1:/dev/rdsk/c1t1d0
d3                  qfs1mds-2:/dev/rdsk/c1t1d0
d4                  qfs1mds-1:/dev/rdsk/c1t2d0
d4                  qfs1mds-2:/dev/rdsk/c1t2d0
d5                  qfs1mds-1:/dev/rdsk/c1t3d0
d5                  qfs1mds-2:/dev/rdsk/c1t3d0
d6                  qfs1mds-2:/dev/rdsk/c0t0d0
d7                  qfs1mds-2:/dev/rdsk/c0t1d0
```

3. On the selected primary node, create a high-performance **ma** file system that uses **md** or **mr** data devices. In a text editor, open the **/etc/opt/SUNWsamfs/mcf** file.

   In the example, we configure the file system **qfs1**. We configure device **d3** as the metadata device (equipment type **mm**), and use **d4** and **d5** as data devices (equipment type **mr**):

```
root@qfs1mds-1:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment           Equipment  Equipment  Family   Device   Additional
# Identifier          Ordinal    Type       Set      State    Parameters
#------------------   ---------  ---------  -------  ------   ----------------
qfs1                  100        ma         qfs1     -
/dev/did/dsk/d3s0     101        mm         qfs1     -
/dev/did/dsk/d4s0     102        mr         qfs1     -
/dev/did/dsk/d5s1     103        mr         qfs1     -
```

4. In the **/etc/opt/SUNWsamfs/mcf** file, enter the **shared** parameter in the **Additional Parameters** column of the file system entry. Save the file.

```
root@qfs1mds-1:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment           Equipment  Equipment  Family   Device   Additional
# Identifier          Ordinal    Type       Set      State    Parameters
#------------------   ---------  ---------  -------  ------   ----------------
qfs1                  100        ma         qfs1     -        shared
/dev/did/dsk/d3s0     101        mm         qfs1     -
/dev/did/dsk/d4s0     102        mr         qfs1     -
/dev/did/dsk/d5s1     103        mr         qfs1     -
:wq
root@qfs1mds-1:~#
```

5. Check the **mcf** file for errors. Use the command **/opt/SUNWsamfs/sbin/sam-fsd**, and correct any errors found.

   The **sam-fsd** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error. In the example, we check the **mcf** file on host **qfs1mds-1**:

```
root@qfs1mds-1:~# sam-fsd
...
Would start sam-archiverd()
Would start sam-stagealld()
Would start sam-stagerd()
Would start sam-amld()
root@qfs1mds-1:~#
```

6. Create the file system. Use the command **`/opt/SUNWsamfs/sbin/sammkfs -S`** *`family-set-name`*, where *`family-set-name`* is the equipment identifier for the file-system.

   The **`sammkfs`** command reads the **`hosts.`***`family-set-name`* and **`mcf`** files on the primary node, **`qfs1mds-1`**, and creates a shared file system with the specified properties.

```
root@qfs1mds-1:~# sammkfs -S qfs1
Building 'qfs1' will destroy the contents of devices:
  ...
Do you wish to continue? [y/N]yes ...
root@qfs1mds-1:~#
```

7. Now exclude QFS data devices from cluster control.

Exclude QFS Data Devices from Cluster Control

By default, the Solaris Cluster software fences off disk devices for the exclusive use of the cluster. In HA-COTC configurations, however, only the metadata (**`mm`**) devices are part of the cluster. Data (**`mr`**) devices are shared with file-system clients outside the cluster and directly attached to the client hosts. So you have to place the data (**`mr`**) devices outside the cluster software's control. This can be achieved in either of two ways:

- Disable fencing of QFS data devices in the HA-COTC cluster.
- Place shared data devices in a local-only device group in the HA-COTC cluster.

Disable Fencing of QFS Data Devices in the HA-COTC Cluster

1. Log in to the primary node of the HA-COTC cluster and the active metadata server for the QFS shared file system. Log in as **`root`**.

   In the example, **`qfs1mds-1`** is the primary node and active metadata server:

```
root@qfs1mds-1:~#
```

2. For each data (**`mr`**) device defined in the **`/etc/opt/SUNWsamfs/mcf`** file, disable fencing. Use the command **`cldevice set -p default_fencing=nofencing-noscrub`** *`device-identifier`*, where *`device-identifier`* is the device identifier listed for the device in the first column of the **`mcf`** file.

   Do not disable fencing for metadata (**`mm`**) devices! In the HA-COTC configurations, the QFS metadata (**`mm`**) devices are part of the cluster, while the QFS shared data (**`mr`**) devices not. Data devices are directly attached to clients outside the cluster. For this reason HA-COTC data (**`mr`**) devices must be managed as local devices that are not managed by the Solaris Cluster software. Otherwise, the Solaris Cluster software and QFS could work at cross purposes and corrupt data.

   In the examples above, we configured devices **`d4`** and **`d5`** as the data devices for the file system **`qfs1`**. So we globally disable fencing for these devices:

```
root@qfs1mds-1:~# cldevice set -p default_fencing=nofencing-noscrub d4
```

```
root@qfs1mds-1:~# cldevice set -p default_fencing=nofencing-noscrub d5
```

3. Now go to "Mount the QFS File System on the Primary HA-COTC Cluster Node" on page 9-14.

Place Shared Data Devices in a Local-Only Device Group on the HA-COTC Cluster

1. Log in to the primary node of the HA-COTC cluster and the active metadata server for the QFS shared file system. Log in as **root**.

   In the example, **qfs1mds-1** is the primary node and active metadata server:

```
root@qfs1mds-1:~#
```

2. Place all data (**mr**) devices that are part of the file system in a **localonly** device group. Use the command **cldevicegroup set -d** *device-identifier-list* **-p localonly=true -n** *active-mds-node device-group*, where *device-list* is a comma-delimited list of device identifiers, *active-mds-node* is the primary node where the active metadata server normally resides, and *device-group* is the name you choose for your device group.

   In the following example, we place data devices **d4** and **d5** (**mcf** equipment numbers **102** and **103**) in the local device group **mdsdevgrp** on the primary node:

```
root@qfs1mds-1:~# cldevicegroup set -d d4,d5 -p localonly=true -n node1mds
mdsdevgrp
root@qfs1mds-1:~#
```

3. Next, mount the QFS file system on the primary HA-COTC cluster node.

Mount the QFS File System on the Primary HA-COTC Cluster Node

1. Log in to the primary node of the HA-COTC cluster and the active metadata server for the QFS shared file system. Log in as **root**.

   In the example, **qfs1mds-1** is the primary node and active metadata server:

```
root@qfs1mds-1:~#
```

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

```
root@qfs1mds-1:~# cp /etc/vfstab /etc/vfstab.backup
```

3. Open the operating system's **/etc/vfstab** file in a text editor, and start a line for the new file system. Enter the file system name in the first column (**Device to Mount**), followed by one or more spaces.

   In the example, use the **vi** text editor. We start a line for the **qfs1** file system:

```
root@qfs1mds-1:~# 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       -
...
qfs1       -
```

4. In the second column of the **/etc/vfstab** file (**Device to fsck**), enter a hyphen (**-**), followed by one or more spaces.

The hyphen tells the operating to skip file-system integrity checking. These checks are intended for UFS rather than SAMFS file systems.

```
root@qfs1mds-1:~# 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       -
...
qfs1       -
```

5. In the third column of the **/etc/vfstab** file, enter the mount point of the file system relative to the cluster. Select a subdirectory that is not directly beneath the system root directory.

   Mounting a shared QFS file system immediately under root can cause failover issues when using the **SUNW.qfs** resource type. In the example, we set the mount point on the cluster to **/global/ha_cotc/qfs1**:

```
#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       -
...
qfs1       -        /global/ha_cotc/qfs1
```

6. Populate the remaining fields of the **/etc/vfstab** file record as you would for any shared QFS file system. Then save the file, and close the editor.

```
#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       -
...
qfs1       -        /global/ha_cotc/qfs1  samfs   -     no       shared
:wq
root@qfs1mds-1:~#
```

7. Create mount point for the high-availability shared file system.

   The **mkdir** command with the **-p** (*parents*) option creates the **/global** directory if it does not already exist:

```
root@qfs1mds-1:~# mkdir -p /global/ha_cotc/qfs1
```

8. Mount the high-availability shared file system on the primary node.

```
root@qfs1mds-1:~# mount /global/ha_cotc/qfs1
```

9. Next, configure a potential QFS metadata server on the secondary HA-COTC cluster node.

Configure a Potential QFS Metadata Server on the Secondary HA-COTC Cluster Node

The secondary node of the two-node cluster serves as the potential metadata server. A potential metadata server is a host that can access to the metadata devices and can, therefore, assume the duties of a metadata server. So, if the active metadata server on the primary node fails, the Solaris Cluster software can failover to the secondary node and activate the potential metadata server. To configure the potential metadata server, carry out the following tasks:

- Create a high-performance QFS file system on the secondary HA-COTC cluster node.
- Mount the QFS file system on the secondary HA-COTC cluster node.

Create a High-Performance QFS File System on the Secondary HA-COTC Node

1. Log in to the secondary node of the HA-COTC cluster as **root**.

   In the example, **qfs1mds-2** is the secondary node and the potential metadata server:

   ```
   root@qfs1mds-2:~#
   ```

2. Copy the **/etc/opt/SUNWsamfs/mcf** file from the primary node to the secondary node.
3. Check the **mcf** file for errors. Use the command **/opt/SUNWsamfs/sbin/sam-fsd**, and correct any errors found.

   The **sam-fsd** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error. In the example, we check the **mcf** file on host **qfs1mds-2**:

   ```
   root@qfs1mds-2:~# sam-fsd
   ...
   Would start sam-archiverd()
   Would start sam-stagealld()
   Would start sam-stagerd()
   Would start sam-amld()
   root@qfs1mds-2:~#
   ```

4. Next, mount the QFS file system on the secondary HA-COTC cluster node.

Mount the QFS File System on the Secondary HA-COTC Node

1. Log in to the secondary node of the HA-COTC cluster as **root**.

   In the example, **qfs1mds-2** is the secondary node:

   ```
   root@qfs1mds-2:~#
   ```

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

   ```
   root@qfs1mds-2:~# cp /etc/vfstab /etc/vfstab.backup
   ```

3. Open the operating system's **/etc/vfstab** file in a text editor, and add the line for the new file system. Then save the file, and close the editor.

   In the example, we use the **vi** editor:

   ```
   root@qfs1mds-2:~# 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       -
   ...
   ```

```
qfs1        -        /global/ha_cotc/qfs1  samfs   -     no       shared
:wq
```

4. Create the mount point for the high-availability shared file system on the secondary node.

```
root@qfs1mds-2:~# mkdir -p /global/ha_cotc/qfs1
root@qfs1mds-2:~#
```

5. Mount the high-availability shared file system on the secondary node.

```
root@qfs1mds-2:~# mount /global/ha_cotc/qfs1
root@qfs1mds-2:~#
```

6. Now configure failover of the HA-COTC metadata server.

Configure Failover of the HA-COTC Metadata Server

When you host an Oracle HSM shared file system in a cluster managed by Solaris Cluster software, you configure failover of the metadata servers by creating a **SUNW.qfs** cluster resource, a resource type defined by the Oracle HSM software (see the **SUNW.qfs** man page for details). To create and configure the resource for an HA-COTC configuration, proceed as follows:

1. Log in to the primary node in the HA-COTC cluster as **root**.

   In the example, **qfs1mds-1** is the primary node:

```
root@qfs1mds-1:~#
```

2. See if the **SUNW.qfs** resource type has been registered with the cluster. Use the command **clresourcetype show**.

   In the example, the **SUNW.qfs** has already been registered:

```
root@qfs1mds-1:~# clresourcetype show
=== Registered Resource Types ===
...
Resource Type:                  SUNW.qfs:5
  RT_description:               SAM-QFS Agent on Solaris Cluster
...
root@qfs1mds-1:~#
```

3. If the **SUNW.qfs** resource type is not registered, register it now. Use the command **clresourcetype register SUNW.qfs**.

```
root@qfs1mds-1:~# clresourcetype register SUNW.qfs
root@qfs1mds-1:~#
```

4. If registration fails because the registration file cannot be found, place a symbolic link to the **/opt/SUNWsamfs/sc/etc/** directory in the directory where Solaris Cluster keeps resource-type registration files, **/opt/cluster/lib/rgm/rtreg/**.

   You did not install Oracle Solaris Cluster software before installing Oracle HSM software. Normally, Oracle HSM automatically provides the location of the **SUNW.qfs** registration file when it detects Solaris Cluster during installation. So you need to create a link manually.

```
root@qfs1mds-1:~# cd /opt/cluster/lib/rgm/rtreg/
root@qfs1mds-1:~# ln -s /opt/SUNWsamfs/sc/etc/SUNW.qfs SUNW.qfs
root@qfs1mds-1:~#
```

5. Create a resource group for the QFS metadata server. Use the Solaris Cluster command **`clresourcegroup create -n`** *`node-list group-name`*, where *`node-list`* is a comma-delimited list of the two cluster node names and *`group-name`* is the name that we want to use for the resource group.

   In the example, we create the resource group **`qfsrg`** with the HA-COTC server nodes as members:

```
root@qfs1mds-1:~# clresourcegroup create -n qfs1mds-1,qfs1mds-2 qfsrg
root@qfs1mds-1:~#
```

6. In the new resource group, set up a logical host name for the active metadata server. Use the command **`clreslogicalhostname create -g`** *`group-name virtualMDS`*, where:

   - *`group-name`* is the name of the QFS resource group.
   - *`virtualMDS`* is the logical host name.

     The cluster maps the logical host name to the public network interface on the currently active node, so that user and client access to resources does not depend on the availability of a specific physical network interface.

   Use the same logical host name that you used in the hosts files for the shared file system. In the example, we create the virtual host **`qfs1mds`** in the **`qfsr`** resource group:

```
root@qfs1mds-1:~# clreslogicalhostname create -g qfsrg qfs1mds
```

7. Add the QFS file-system resources to the resource group. Use the command **`clresource create -g`** *`group-name`* **`-t SUNW.qfs -x QFSFileSystem=`***`mount-point`* **`-y Resource_dependencies=`***`virtualMDS resource-name`*, where:

   - *`group-name`* is the name of the QFS resource group.
   - *`mount-point`* is the mount point for the file system in the cluster, a subdirectory that is not directly beneath the system root directory.

     Mounting a shared QFS file system immediately under root can cause failover issues when using the **`SUNW.qfs`** resource type.
   - *`virtualMDS`* is the logical host name of the active metadata server.
   - *`resource-name`* is the name that you want to give to the resource.

   In the example, we create a resource named **`haqfs`** of type **`SUNW.qfs`** in the resource group **`qfsrg`**. We set the **`SUNW.qfs`** extension property **`QFSFileSystem`** to the **`/global/ha_cotc/qfs1`** mount point, and set the standard property **`Resource_dependencies`** to the logical host for the active metadata server, **`qfs1mds`**:

```
root@qfs1mds-1:~# clresource create -g qfsrg -t SUNW.qfs
-x QFSFileSystem=/global/ha_cotc/qfs1 -y Resource_dependencies=qfs1mds haqfs
...
root@qfs1mds-1:~#
```

8. The cluster should not make the QFS file system available if users and applications cannot reach the active metadata server. So make the **`SUNW.qfs`** resource depend on the logical host name. Use the Solaris Cluster command **`clresource set -p Resource_dependencies=`***`virtualMDS resource-name`*, where:

   - *`virtualMDS`* is the cluster's logical host name for the active file-system metadata server.

The logical host name always points to the public network interface on the active metadata server, so that user and client access to the file system does not depend on the availability of a specific physical network interface.

- *resource-name* is the name of the **SUNW.qfs** resource.

In the example, the logical host name that we created when we set up the **SUNW.qfs** resource is **qsm1mds**. The resource itself is named **haqfs**:

```
root@qfs1mds-1:~# clresource set -p Resource_dependencies=qsm1mds haqfs
root@qfs1mds-1:~#
```

9. Bring the resource group online. Use the command **clresourcegroup online -emM** *group-name*, where *group-name* is the name of the QFS resource group.

   In the example, we bring the **qfsr** resource group online:

```
root@qfs1mds-1:~# clresourcegroup manage qfsrg
root@qfs1mds-1:~# clresourcegroup online -emM qfsrg
```

10. Make sure that the QFS resource group is online. Use the Solaris Cluster **clresourcegroup status** command.

    In the example, the **qfsrg** resource group is **online** on the primary node, **hsm1mds-node1**:

```
root@qfs1mds-1:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name  Node Name       Suspended   Status
----------  -------------   ---------   ------
qfsrg       qfs1mds-1   No          Online
            qfs1mds-2   No          Offline
```

11. Make sure that the resource group fails over correctly by moving the resource group to the secondary node. Use the Solaris Cluster command **clresourcegroup switch -n** *node2 group-name*, where *node2* is the name of the secondary node and *group-name* is the name that you have chosen for the HA-QFS resource group. Then use **clresourcegroup status** to check the result.

    In the example, we move the **qfsrg** resource group to **qfs1mds-2** and confirm that the resource group comes online on the specified node:

```
root@qfs1mds-1:~# clresourcegroup switch -n qfs1mds-2 qfsrg
root@qfs1mds-1:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name  Node Name       Suspended   Status
----------  -------------   ---------   ------
qfsrg       qfs1mds-1   No          Offline
            qfs1mds-2   No          Online
```

12. Move the resource group back to the primary node. Use the Solaris Cluster command **clresourcegroup switch -n** *node1 group-name*, where *node1* is the name of the primary node and *group-name* is the name that you have chosen for the HA-QFS resource group. Then use **clresourcegroup status** to check the result.

    In the example, we successfully move the **qfsrg** resource group back to **qfs1mds-1**:

```
root@qfs1mds-1:~# clresourcegroup switch -n qfs1mds-1 qfsrg
root@qfs1mds-1:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name  Node Name       Suspended   Status
----------  -------------   ---------   ------
```

```
qfsrg       qfs1mds-1   No          Online
            qfs1mds-2   No          Offline
```

13. Next, configure hosts outside the HA-COTC cluster as QFS shared file system clients.

Configure Hosts Outside the HA-COTC Cluster as QFS Shared File System Clients

Configure each host as a QFS client that *does not* have access to the file system's metadata devices, so that the clients do not interfere with the high-availability configuration of the metadata servers inside the cluster.

For each client of the HA-COTC shared file system, proceed as follows:

1. Log in to the primary node in the HA-COTC cluster. Log in as **`root`**.

```
root@qfs1mds-1:~#
```

2. Display the device configuration for the cluster. Use the Solaris Cluster command **`/usr/global/bin/cldevice list -v`** command.

```
root@qfs1mds-1:~# cldevice list -v
DID Device          Full Device Path
----------          ----------------
d1                  qfs1mds-1:/dev/rdsk/c0t0d0
d2                  qfs1mds-1:/dev/rdsk/c0t6d0
...
d7                  qfs1mds-2:/dev/rdsk/c0t1d0
root@qfs1mds-1:~#
```

3. Examine the output of the **`cldevice list -v`** command. Make a note of the **`/dev/rdsk/`** path corresponding to the device identifier for each QFS data (**`mr`**) device.

   In the example, the QFS data devices are **`d4`** and **`d5`**:

```
root@qfs1mds-1:~# cldevice list -v
DID Device          Full Device Path
----------          ----------------
d1                  qfs1mds-1:/dev/rdsk/c0t0d0
d2                  qfs1mds-1:/dev/rdsk/c0t6d0
d3                  qfs1mds-1:/dev/rdsk/c1t1d0
d3                  qfs1mds-2:/dev/rdsk/c1t1d0
d4                  qfs1mds-1:/dev/rdsk/c1t2d0
d4                  qfs1mds-2:/dev/rdsk/c1t2d0
d5                  qfs1mds-1:/dev/rdsk/c1t3d0
d5                  qfs1mds-2:/dev/rdsk/c1t3d0
d6                  qfs1mds-2:/dev/rdsk/c0t0d0
d7                  qfs1mds-2:/dev/rdsk/c0t1d0
root@qfs1mds-1:~#
```

4. Log in to the client host of the HA-COTC cluster as **`root`**.

   In the example, **`qfs1client1`** is the client host:

```
root@qfs1mds-1:~# ssh root@qfs1client1
root@qfs1client-1:~#
```

5. On the client host, retrieve the configuration information for the shared file system. Use the **`samfsconfig /dev/rdsk/*`** command.

The **samfsconfig /dev/rdsk/*** command searches the specified path for attached devices that belong to a QFS file system. In the example, the command finds the paths to the **qfs1** data (**mr**) devices. As expected, it does not find the metadata (**mm**) devices, so it returns the **Missing slices** and **Ordinal 0** messages before listing the shared data devices:

```
root@qfs1client-1:~# samfsconfig /dev/rdsk/*
# Family Set 'qfs1' Created Thu Dec 21 07:17:00 2013
# Missing slices
# Ordinal 0
# /dev/rdsk/c1t2d0s0   102        mr       qfs1  -
# /dev/rdsk/c1t3d0s1   103        mr       qfs1  -
```

6. Compare the output of the **samfsconfig** command with the output of the Solaris Cluster **cldevice list** command on the server. Make sure that both report the same device paths for the **mr** data devices.

   The **samfsconfig** and **cldevice list** commands should indicate the same devices, though the controller numbers (**c***N*) may differ. In the example, the **samfsconfig** and **cldevice list** commands do point to the same devices.

   On the metadata server node, the **/etc/opt/SUNWsamfs/mcf** file identifies shared **mr** data devices **102** and **103** using cluster device identifiers **d4** and **d5**:

```
 /dev/did/dsk/d4s0     102        mr         qfs1  -
 /dev/did/dsk/d5s1     103        mr         qfs1  -
```

   The **cldevice list** command on the metadata server node maps cluster device identifiers **d4** and **d5** to the paths **/dev/rdisk/c1t2d0** and **/dev/rdisk/c1t3d0**:

```
 d4                  qfs1mds-1:/dev/rdsk/c1t2d0
 d5                  qfs1mds-1:/dev/rdsk/c1t3d0
```

   On the client node, the **samfsconfig** command also identifies shared **mr** data devices **102** and **103** with the paths **/dev/rdisk/c1t2d0** and **/dev/rdisk/c1t3d0**:

```
 /dev/rdsk/c1t2d0s0    102        mr       qfs1  -
 /dev/rdsk/c1t3d0s1    103        mr       qfs1  -
```

7. Open the client's **/etc/opt/SUNWsamfs/mcf** file in a text editor. Add an entry for the HA-COTC shared file system. The entry should exactly match the corresponding entries in the metadata server **mcf** files.

   In the example, we use the **vi** editor to create an entry for the file QFS share system **qfs1** (equipment ordinal number **100**):

```
root@qfs1client-1:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment           Equipment  Equipment  Family   Device   Additional
# Identifier          Ordinal    Type       Set      State    Parameters
#------------------   ---------  ---------  -------  ------   ----------------
qfs1                  100        ma         qfs1     -        shared
```

8. On a new line, start an entry for the HA-COTC shared file system's metadata (**mm**) devices. In the first column (**Equipment Identifier**), enter keyword **nodev**.

```
# Equipment           Equipment  Equipment  Family   Device   Additional
# Identifier          Ordinal    Type       Set      State    Parameters
#------------------   ---------  ---------  -------  ------   ----------------
qfs1                  100        ma         qfs1     -        shared
nodev
```

9. Populate the remaining fields for the HA-COTC file system's metadata (**mm**) devices with the same equipment ordinal numbers, family set, and device state parameters used in the metadata server **mcf** files.

```
# Equipment          Equipment  Equipment  Family   Device  Additional
# Identifier         Ordinal    Type       Set      State   Parameters
#------------------  ---------  ---------  -------  ------  ----------------
qfs1                 100        ma         qfs1     -       shared
nodev                101        mm         qfs1     -
```

10. Copy the complete entries for the data (**mr**) devices from the **samfsconfig** output. Paste the entries into the client's **/etc/opt/SUNWsamfs/mcf** file. Remove the leading comment (**#**) marks that **samfsconfig** inserts. Then save the file, and close the editor.

```
# Equipment          Equipment  Equipment  Family   Device  Additional
# Identifier         Ordinal    Type       Set      State   Parameters
#------------------  ---------  ---------  -------  ------  -----------------
qfs1                 100        ma         qfs1     -       shared
nodev                101        mm         qfs1     -
/dev/rdsk/c1t2d0s0   102        mr         qfs1     -
/dev/rdsk/c1t3d0s1   103        mr         qfs1     -
:wq
root@qfs1client-1:~#
```

11. Check the **mcf** file for errors. Use the command **/opt/SUNWsamfs/sbin/sam-fsd**, and correct any errors found.

    The **sam-fsd** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error. In the example, we check the **mcf** file on host **qfs1client1**:

```
root@qfs1client-1:~# sam-fsd
...
Would start sam-archiverd()
Would start sam-stagealld()
Would start sam-stagerd()
Would start sam-amld()
root@qfs1client-1:~#
```

12. Open the client operating system's **/etc/vfstab** file in a text editor, and add an entry for the new file system, using the same parameters used on the server. Then save the file, and close the editor.

    In the example, we use the **vi** editor:

```
root@qfs1client1:~# 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       -
...
qfs1       -        /global/ha_cotc/qfs1 samfs   -     no       shared
:wq
root@qfs1client-1:~#
```

13. Create the mount point for the high-availability shared file system on the client.

```
root@qfs1client-1:~# mkdir -p /global/qfs1
root@qfs1client-1:~#
```

14. Mount the high-availability shared file system on the client.

    In the example

```
root@qfs1client-1:~# mount /global/qfs1
root@qfs1client-1:~#
```

15. Repeat this procedure until all HA-COTC clients have been configured.
16. If you plan on using the sideband database feature, go to "Configuring the Reporting Database" on page 10-1.
17. Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

High-Availability Oracle HSM Shared Archiving File Systems

The High-Availability Oracle Hierarchical Storage Manager (HA-HSM) configuration maintains the availability of an archiving 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. If the active cluster node fails, the clustering software automatically activates the potential Oracle HSM server on the surviving node and transfers control over running operations. Since the QFS file system and the Oracle HSM application's local storage directories are shared and already mounted, access to data and metadata remains uninterrupted.

The HA-HSM configuration insures file-system consistency in a clustered environment by sending all I/O through the active metadata server. You share the HA-HSM file system purely for accessibility reasons. You cannot use the potential metadata server host as a file-system client, as you would in other SAM-QFS shared file-system configurations. The potential metadata server does not perform I/O unless it is activated during node failover. You can share an HA-HSM file system with clients using NFS. But you must insure that the shares are exported exclusively from the active metadata server node.

To failover successfully, the HA-HSM configuration must transfer operations from a failed host to a standby while maintaining consistent system and application state and full access to user data. Three Solaris Cluster Data Services resource types assist with these tasks:

- A **`SUNW.qfs`** resource manages failover of the metadata server for the shared QFS file system.
- A **`SUNW.hasam`** resource manages failover of the Oracle Hierarchical Storage Manager archiving application.
- A **`SUNW.HAStoragePlus`** resource manages failover of the host's local storage, so that critical application state and configuration information remains available following host failover.

The **`SUNW.qfs`** and **`SUNW.hasam`** software are included in the Oracle HSM software distribution, while **`SUNW.HAStoragePlus`** is included in the Solaris Cluster software as a standard resource type (for more information on resource types, see the *Data Services Planning and Administration* documentation in the *Oracle Solaris Cluster Documentation Library* and the man pages for each resource type).

To configure instances of the required components and integrate them into a working HA-HSM archiving configuration, carry out the following tasks:

- Make sure that you have correctly configured the storage for use with **`SUNW.HAStoragePlus`**.

  See "Configure Solaris Cluster Nodes for Multipath I/O" on page 3-16", "Configure Linux Clients for Multipath I/O" on page 3-17, and the documentation for the Solaris Cluster **`SUNW.HAStoragePlus`** data service.
- Configure Oracle HSM network communications.
- Create QFS file systems and configure metadata servers for the HA-HSM solution.
- Configure failover for the Oracle HSM file systems and software
- Bring the HA-HSM resource group online and test the configuration.
- If required, configure HA-NFS and/or HA-SAMBA.

Configure Oracle HSM Network Communications

During the initial creation of an Oracle HSM shared file system, the **`sammkfs -S`** command configures sharing using the settings stored in hosts files on the active and potential metadata servers. The hosts files are stored alongside the **`mcf`** file in the **`/etc/opt/SUNWsamfs/`** directory. So create these files now, using the procedure below, before creating the file system.

Configure a global and a local hosts file on each host, so that you can selectively route network traffic between the network interfaces on the servers. Use one network on each server node for the private network that connects the nodes of the Solaris Cluster and the other for the general network communications.

To configure global and local hosts files, carry out the following tasks:

- Create a global hosts file on each of the HA-HSM cluster nodes.
- Create a local hosts file on each of the HA-HSM cluster nodes.

Create a Global Hosts File on Both HA-HSM Cluster Nodes

1. Log in to the primary node of the HA-HSM cluster as **`root`**.

   In the example, **`hsm1mds-node1`** is the primary node:

   ```
   root@hsm1mds-node1:~#
   ```

2. Display the cluster configuration. Use the **`/usr/global/bin/cluster show`** command. In the output, locate the record for each **`Node Name`**, and note the **`privatehostname`**, **`Transport Adapter`** name, and **`ip_address`** property of each network adapter.

   In the example, each node has two network interfaces, **`hme0`** and **`qfe3`**:

   - The **`hme0`** adapters have IP addresses on the private network that the cluster uses for internal communication between nodes. The Solaris Cluster software assigns a **`privatehostname`** corresponding to each private address.

     By default, the private hostname of the primary node is **`clusternode1-priv`** and the private hostname of the secondary node is **`clusternode2-priv`**.
   - The **`qfe3`** adapters have public IP addresses and public hostnames—**`hsm1mds-node1`** and **`hsm1mds-node2`**—that the cluster uses for data transport.

   Note that the display has been abbreviated using ellipsis (**`...`**) marks:

   ```
   root@hsm1mds-node1:~# cluster show
   ```

```
...
  === Cluster Nodes ===
  Node Name:                                  hsm1mds-node1...
    privatehostname:                             clusternode1-priv...
    Transport Adapter List:                      qfe3, hme0...
    Transport Adapter:                        qfe3...
      Adapter Property(ip_address):              172.16.0.12...
    Transport Adapter:                        hme0...
      Adapter Property(ip_address):              10.0.0.129...
  Node Name:                                  hsm1mds-node2...
    privatehostname:                             clusternode2-priv...
    Transport Adapter List:                      qfe3, hme0...
      Adapter Property(ip_address):              172.16.0.13...
    Transport Adapter:                        hme0...
      Adapter Property(ip_address):              10.0.0.122
```

3. Using a text editor, create the file **/etc/opt/SUNWsamfs/hosts.***family-set-name*, where *family-set-name* is the family-set name that the **/etc/opt/SUNWsamfs/mcf** file assigns to the file-system equipment.

   In the example, we create the file **hosts.hsm1** using the **vi** text editor. We add some optional headings to show the columns in the hosts table, starting each line with a hash sign (**#**) to indicate a comment:

```
[root@hsm1mds-node1:~# vi /etc/opt/SUNWsamfs/hosts.hsm1
# /etc/opt/SUNWsamfs/hosts.hsm1
#                                             Server   On/  Additional
#Host Name     Network Interface              Ordinal  Off  Parameters
#------------  -----------------------------  -------  ---  ----------
```

4. In the first column of the table, enter the hostnames of the primary and secondary metadata server nodes followed by some spaces, with each entry on a separate line.

   In a hosts file, the lines are rows (records) and spaces are column (field) separators. In the example, the **Host Name** column of the first two rows contains the values **hsm1mds-node1** and **hsm1mds-node2**, the hostnames of the cluster nodes that host the metadata servers for the file system:

```
#                                             Server   On/  Additional
#Host Name     Network Interface              Ordinal  Off  Parameters
#------------  -----------------------------  -------  ---  ----------
hsm1mds-node1
hsm1mds-node2
```

5. In the second column of each line, start supplying **Network Interface** information for the hosts listed in the **Host Name** column. Enter each HA-HSM cluster node's Solaris Cluster private hostname or private network address followed by a comma.

   The HA-HSM server nodes use the private hostnames for server-to-server communications within the high-availability cluster. In the example, we use the private hostnames **clusternode1-priv** and **clusternode2-priv**, which are the default names assigned by the Solaris Cluster software:

```
#                                             Server   On/  Additional
#Host Name     Network Interface              Ordinal  Off  Parameters
#------------  -----------------------------  -------  ---  ----------
hsm1mds-node1  clusternode1-priv,
hsm1mds-node2  clusternode2-priv,
```

6. Following the comma in the second column of each line, enter the public hostname for the active metadata server followed by spaces.

   The HA-HSM server nodes use the public data network to communicate with hosts outside the cluster. Since the IP address and hostname of the active metadata server changes during failover (from **hsm1mds-node1** to **hsm1mds-node2** and vice versa), we use a logical host name—**hsm1mds**—for both. Later, we will configure the Solaris Cluster software to always route requests for **hsm1mds** to the active metadata server:

```
#                                              Server   On/  Additional
#Host Name     Network Interface               Ordinal  Off  Parameters
#------------  ----------------------------    -------  ---  ----------
hsm1mds-node1  clusternode1-priv,hsm1mds
hsm1mds-node2  clusternode2-priv,hsm1mds
```

7. In the third column of each line, enter the ordinal number of the server (**1** for the active metadata server, and **2** for the potential metadata server), followed by spaces.

   In this example, there is only one metadata server, the primary node, **hsm1mds-node1**, is the active metadata server, so it is ordinal **1** and the secondary node, **hsm1mds-node2**, is ordinal **2**:

```
#                                              Server   On/  Additional
#Host Name     Network Interface               Ordinal  Off  Parameters
#------------  ----------------------------    -------  ---  ----------
hsm1mds-node1  clusternode1-priv,hsm1mds       1
hsm1mds-node2  clusternode2-priv,hsm1mds       2
```

8. In the fourth column of each line, enter **0** (zero), followed by spaces.

   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 **samsharefs** man page).

```
#                                              Server   On/  Additional
#Host Name     Network Interface               Ordinal  Off  Parameters
#------------  ----------------------------    -------  ---  ----------
hsm1mds-node1  clusternode1-priv,hsm1mds       1        0
hsm1mds-node2  clusternode2-priv,hsm1mds       2        0
```

9. In the fifth column of the line for the primary node, enter the keyword **server**. Then save the file and close the editor.

   The server keyword identifies the default, active metadata server:

```
#                                              Server   On/  Additional
#Host Name     Network Interface               Ordinal  Off  Parameters
#------------  ----------------------------    -------  ---  ----------
hsm1mds-node1  clusternode1-priv,hsm1mds       1        0    server
hsm1mds-node2  clusternode2-priv,hsm1mds       2        0
:wq
root@hsm1mds-node1:~#
```

10. Place a copy of the global **/etc/opt/SUNWsamfs/hosts.***family-set-name* file on the potential metadata server.

11. Now, create a local hosts file on each of the HA-HSM cluster nodes.

Create a Local Hosts File on Each of the HA-HSM Cluster Nodes

In a high-availability archiving shared file system, you need to insure that the servers communicate with each other using the private network defined by the Solaris Cluster software. You do this by using specially configured local hosts files to selectively route network traffic between the network interfaces on the servers.

To identify the network interfaces the other file-system hosts, each host first checks the **/etc/opt/SUNWsamfs/hosts.***family-set-name* file on the metadata server. 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 different arrangements in each file, you can thus control the interfaces used by each host.

To configure local hosts files, use the procedure outlined below:

1. Log in to the primary node of the HA-HSM cluster as **root**.

   In the example, **hsm1mds-node1** is the primary node:

   ```
   root@hsm1mds-node1:~#
   ```

2. Using a text editor, create a local hosts file on the active metadata server, using the path and file name **/etc/opt/SUNWsamfs/hosts.***family-set-name***.local**, where *family-set-name* is the family set name that the **/etc/opt/SUNWsamfs/mcf** file assigns to the file system equipment. *Only include network interfaces that you want the active server to use when communicating with the potential server*. Then save the file and close the editor.

   In our example, we want the active and potential metadata servers to communicate with each other over the private network. So the local hosts file on the active metadata server, **hosts.hsm1.local**, lists only cluster private addresses for the active and potential servers:

   ```
   root@hsm1mds-node1:~# vi /etc/opt/SUNWsamfs/hosts.hsm1.local
   #                                             Server   On/  Additional
   #Host Name     Network Interface             Ordinal  Off  Parameters
   #------------  ----------------------------  -------  ---  ----------
   hsm1mds-node1  clusternode1-priv             1        0    server
   hsm1mds-node2  clusternode2-priv             2        0
   :wq
   root@hsm1mds-node1:~#
   ```

3. Log in to the secondary cluster node as **root**.

   In the example, **hsm1mds-node2** is the secondary node:

   ```
   root@hsm1mds-node1:~# ssh root@hsm1mds-node2
   Password:
   root@hsm1mds-node2:~#
   ```

4. Using a text editor, create a local hosts file on the potential metadata server. Use the path and file name **/etc/opt/SUNWsamfs/hosts.***family-set-name***.local**, where *family-set-name* is the family-set name that the **/etc/opt/SUNWsamfs/mcf** file assigns to the file-system equipment. *Only include network interfaces that you want the potential server to use when communicating with the active server*. Then save the file and close the editor.

In our example, we want the active and potential metadata servers to communicate with each other over the private network. So the local hosts file on the potential metadata server, **hosts.hsm1.local**, lists only cluster private addresses for the active and potential servers:

```
root@hsm1mds-node2:~# vi /etc/opt/SUNWsamfs/hosts.hsm1.local
#                                             Server   On/  Additional
#Host Name     Network Interface              Ordinal  Off  Parameters
#------------  ----------------------------  -------  ---  ----------
hsm1mds-node1  clusternode1-priv              1        0    server
hsm1mds-node2  clusternode2-priv              2        0
:wq
root@hsm1mds-node2:~# exit
root@hsm1mds-node1:~#
```

5. Next, create the metadata servers and QFS file systems for HA-HSM.

Configure Metadata Servers and Create QFS File Systems for the HA-HSM Solution

Carry out the following tasks below:

- Create the QFS file system on the HA-HSM primary cluster node.
- Create the QFS file system the HA-HSM secondary cluster node.

Create the QFS File System on the HA-HSM Primary Cluster Node

The primary node of the two-node cluster serves as the active metadata server (MDS) that controls the shared file system during normal operation.

1. Select the cluster node that will serve as both the primary node for the HA-HSM cluster and the active metadata server for the QFS shared file system. Log in as **root**.

   In the example, **hsm1mds-node1** is the primary node:

```
root@hsm1mds-node1:~#
```

2. Select the global storage devices that will be used for the QFS file system. Use the command **/usr/global/bin/cldevice list -v**.

   Solaris Cluster software assigns unique Device Identifiers (DIDs) to all devices that attach to the cluster nodes. *Global* devices are accessible from all nodes in the cluster, while *local* devices are accessible only from the hosts that mount them. Global devices remain accessible following failover. Local devices do not.

   In the example, note that devices **d1**, **d2**, **d7**, and **d8** are not accessible from both nodes. So we select from devices **d3**, **d4**, and **d5** when configuring the high-availability QFS shared file system:

```
root@hsm1mds-node1:~# cldevice list -v
DID Device          Full Device Path
----------          ----------------
d1                  hsm1mds-node1:/dev/rdsk/c0t0d0
d2                  hsm1mds-node1:/dev/rdsk/c0t6d0
d3                  hsm1mds-node1:/dev/rdsk/c1t1d0
d3                  hsm1mds-node2:/dev/rdsk/c1t1d0
d4                  hsm1mds-node1:/dev/rdsk/c1t2d0
d4                  hsm1mds-node2:/dev/rdsk/c1t2d0
d5                  hsm1mds-node1:/dev/rdsk/c1t3d0
d5                  hsm1mds-node2:/dev/rdsk/c1t3d0
d6                  hsm1mds-node2:/dev/rdsk/c0t0d0
```

```
d7                  hsm1mds-node2:/dev/rdsk/c0t1d0
```

3. On the selected primary node, create a high-performance **ma** file system that uses **mr** data devices. In a text editor, open the **/etc/opt/SUNWsamfs/mcf** file.

   In the example, we configure the file system **hsm1**. We configure device **d3** as the metadata device (equipment type **mm**), and use **d4** and **d5** as data devices (equipment type **mr**):

```
root@hsm1mds-node1:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment           Equipment  Equipment  Family   Device   Additional
# Identifier          Ordinal    Type       Set      State    Parameters
#------------------   ---------  ---------  -------  ------   -----------------
hsm1                  100        ma         hsm1     -
/dev/did/dsk/d3s0     101        mm         hsm1     -
/dev/did/dsk/d4s0     102        mr         hsm1     -
/dev/did/dsk/d5s1     103        mr         hsm1     -
```

4. In the **/etc/opt/SUNWsamfs/mcf** file, enter the **shared** parameter in the **Additional Parameters** column of the file system entry. Save the file.

```
root@hsm1mds-node1:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment           Equipment  Equipment  Family   Device   Additional
# Identifier          Ordinal    Type       Set      State    Parameters
#------------------   ---------  ---------  -------  ------   -----------------
hsm1                  100        ma         hsm1     -        shared
/dev/did/dsk/d3s0     101        mm         hsm1     -
/dev/did/dsk/d4s0     102        mr         hsm1     -
/dev/did/dsk/d5s1     103        mr         hsm1     -
:wq
root@hsm1mds-node1:~#
```

5. Check the **mcf** file for errors. Use the command **/opt/SUNWsamfs/sbin/sam-fsd**, and correct any errors found.

   The **sam-fsd** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error. In the example, we check the **mcf** file on host **hsm1mds-node1**:

```
root@hsm1mds-node1:~# sam-fsd
...
Would start sam-archiverd()
Would start sam-stagealld()
Would start sam-stagerd()
Would start sam-amld()
root@hsm1mds-node1:~#
```

6. Create the file system. Use the command **/opt/SUNWsamfs/sbin/sammkfs -S** *family-set-name*, where *family-set-name* is the family-set name that the **/etc/opt/SUNWsamfs/mcf** file assigns to the file-system equipment.

   The **sammkfs** command reads the **hosts.***family-set-name* and **mcf** files and creates an Oracle HSM file system with the specified properties.

```
root@hsm1mds-node1:~# sammkfs -S hsm1
Building 'hsm1' will destroy the contents of devices:
  ...
Do you wish to continue? [y/N]yes ...
root@hsm1mds-node1:~#
```

7. Open the operating system's **/etc/vfstab** file in a text editor, and start a line for the new file system. Enter the file system name in the first column, spaces, a hyphen in the second column, and more spaces.

   In the example, we use the **vi** text editor. We start a line for the **hsm1** file system. The hyphen keeps the operating system from attempting to check file system integrity using UFS tools:

```
root@hsm1mds-node1:~# 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       -
...
hsm1       -
```

8. In the third column of the **/etc/vfstab** file, enter the mount point of the file system relative to the cluster. Select a subdirectory that is not directly beneath the system root directory.

   Mounting a shared QFS file system immediately under root can cause failover issues when using the **SUNW.qfs** resource type. In the example, we set the mount point on the cluster to **/global/ha_hsmfs/hsm1**:

```
#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       -
...
hsm1       -        /global/ha_hsmfs/hsm1
```

9. Populate the remaining fields of the **/etc/vfstab** file record as you would for any Oracle HSM shared file system. Then save the file, and close the editor.

```
#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       -
...
hsm1       -        /global/ha_hsmfs/hsm1 samfs    -     no       shared
:wq
root@hsm1mds-node1:~#
```

10. Create mount point for the high-availability file system.

    The **mkdir** command with the **-p** (*parents*) option creates the **/global** directory if it does not already exist:

```
root@hsm1mds-node1:~# mkdir -p /global/ha_hsmfs/hsm1
```

11. Mount the high-availability shared file system on the primary node.

```
root@hsm1mds-node1:~# mount /global/ha_hsmfs/hsm1
```

12. Next, create the QFS file system the HA-HSM secondary cluster node.

Create the HA-HSM QFS File System the Secondary Cluster Node

The secondary node of the two-node cluster serves as a potential metadata server (MDS) that stands by to take control of the shared file system should the Solaris Cluster software fail over to the secondary node.

1. Log in to the secondary node of the HA-HSM cluster as **root**.

   In the example, **hsm1mds-node2** is the secondary node:

   ```
   root@hsm1mds-node2:~#
   ```

2. Copy the **/etc/opt/SUNWsamfs/mcf** file from the primary node to the secondary node.

3. Check the **mcf** file for errors. Use the command **/opt/SUNWsamfs/sbin/sam-fsd**, and correct any errors found.

   The **sam-fsd** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error. In the example, we check the **mcf** file on host **hsm1mds-node1**:

   ```
   root@hsm1mds-node2:~# sam-fsd
   ...
   Would start sam-archiverd()
   Would start sam-stagealld()
   Would start sam-stagerd()
   Would start sam-amld()
   root@hsm1mds-node2:~#
   ```

4. Create the file system. Use the command **/opt/SUNWsamfs/sbin/sammkfs -S** *family-set-name*, where *family-set-name* is the family-set name that the **/etc/opt/SUNWsamfs/mcf** file assigns to the file-system equipment.

   The **sammkfs** command reads the **hosts.***family-set-name* and **mcf** files and creates an Oracle HSM file system with the specified properties.

   ```
   root@hsm1mds-node2:~# sammkfs hsm1
   Building 'hsm1' will destroy the contents of devices:
     ...
   Do you wish to continue? [y/N]yes ...
   root@hsm1mds-node2:~#
   ```

5. Open the operating system's **/etc/vfstab** file in a text editor, and add the line for the new file system. Then save the file, and close the editor.

   In the example, we use the **vi** editor:

   ```
   root@hsm1mds-node2:~# 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       -
   ...
   hsm1       -        /global/ha_hsmfs/hsm1 samfs   -     no       shared
   :wq
   root@hsm1mds-node2:~#
   ```

6. Create the mount point for the high-availability shared file system on the secondary node.

   ```
   root@hsm1mds-node2:~# mkdir -p /global/ha_hsmfs/hsm1
   ```

7. Mount the high-availability shared file system on the secondary node.

```
root@hsm1mds-node2:~# mount /global/ha_hsmfs/hsm1
```

8. Now configure failover for the Oracle HSM file systems and software.

Configure Failover for the Oracle HSM File Systems and Software

To configure failover for both the QFS file systems and the Oracle Hierarchical Storage Manager software, carry out the following tasks:

- Create the HA-HSM cluster resource group.
- Create a high-availability local file system for software configuration files.
- Move Oracle HSM configuration files to the high availability local file system.
- Configure failover for the high availability local file system.
- Configure failover of the QFS file-system metadata server.
- Configure failover of the Oracle Hierarchical Storage Manager application.
- Define the cluster resource dependencies for the HA-HSM solution.

Create the HA-HSM Cluster Resource Group

Create the resource group that will manage the high-availability resources for your HA-HSM solution.

1. Log in to the primary cluster node of the HA-HSM cluster as **`root`**.

   In the example, the primary node is **`hsm1mds-node1`**:

```
root@hsm1mds-node1:~#
```

2. Create a Solaris Cluster resource group to manage the HA-HSM solution resources. Use the command **`clresourcegroup create -n`** *`node1`*,*`node2 groupname`*, where:

   - *`node1`* is the hostname of the primary cluster node.
   - *`node2`* is the hostname of the secondary cluster node.
   - *`groupname`* is the name that you have chosen for the HA-HSM resource group.

   In the example, we create a resource group named **`hsmrg`** and include hosts **`hsm1mds-node1`** and **`hsm1mds-node2`**:

```
root@hsm1mds-node1:~# clresourcegroup create -n hsm1mds-node1,hsm1mds-node2
hsmrg
root@hsm1mds-node1:~#
```

3. Next, create a high availability local file system for software configuration files.

Create a High-Availability Local File System for Software Configuration Files

When the cluster fails over to a secondary, replacement node, high-availability software configurations on the replacement node must have access to the pre-failover state of their counterparts on the failed node. For example, the Oracle HSM software needs to access the archive media catalogs and continue staging operations that were in progress when failover occurred.

Oracle HSM normally keeps configuration and state information in the active metadata server's local file system.The local file system on a failed node is, of course, inaccessible. So, to keep the software available, you must create a highly available local file system that is always accessible from either node. Proceed as follows:

1. Log in to the primary node of the cluster as **root**.

   In the example, the primary node is **hsm1mds-node1**:

   ```
   root@hsm1mds-node1:~#
   ```

2. On the primary cluster node, create a UFS file system on a free slice of a global device. Use the command **newfs /dev/global/dsk/d***X***s***Y*, where *X* is the Device Identifier (DID) number of the global device and *Y* is the slice number.

   In the example, we create the new file system on **/dev/global/dsk/d10s0**:

   ```
   root@hsm1mds-node1:~# newfs /dev/global/dsk/d10s0
   newfs: construct a new file system /dev/global/dsk/d10s0: (y/n)? y
   /dev/global/dsk/d10s0: 1112940 sectors in 1374 cylinders of 15 tracks,
   54 sectors 569.8MB in 86 cyl groups (16 c/g, 6.64MB/g, 3072 i/g)
   super-block backups(for fsck -b #) at:
   32, 13056, 26080, 39104, 52128, 65152, 78176, 91200, 104224, . . .
   root@hsm1mds-node1:~#
   ```

3. On the primary cluster node, open the operating system's **/etc/vfstab** file in a text editor. Add a line for the new UFS file system. Save the file, and close the editor.

   The new line should be a space-delimited list of the form **/dev/global/dsk/d***X***s***Y* **/dev/global/dsk/d***X***s***Y* **/global/***mountpoint* **ufs 5 no global**, where:

   - *X* is the Device Identifier (DID) number of the global device that holds the file system.
   - *Y* is the number of the slice that holds the file system.
   - **/dev/global/dsk/d***X***s***Y* is the name of the file-system device that will be mounted.
   - **/dev/global/dsk/d***X***s***Y* is the name of the file-system device that will be checked by the **fsck** command.
   - *mountpoint* is the name of the subdirectory where the UFS file system is to be mounted.
   - **ufs** is the file system type.
   - **5** is the recommended **fsck** pass number.
   - **no** tells the operating system that it should not mount the file system when starting up.
   - **global** mounts the file system so that both nodes have access.

   In the example, the name of the file system is **/dev/global/dsk/d10s0** and the mount point is **/global/ha_config**. Note that the file-system entry is a single line:

   ```
   root@hsm1mds-node1:~# 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       -
   ```

```
...
hsm1       -        /global/hsm1        samfs   -     no       shared
/dev/global/dsk/d10s0   /dev/global/rdsk/d10s0  /global/ha_config  ufs    5
no  global
:wq
root@hsm1mds-node2:~#
```

4. On the primary cluster node, create the mount point for the high-availability local file system. Use the command **`mkdir -p /global/`***`mountpoint`*, where *`mountpoint`* is the selected mount point directory.

   In the example, we create the directory **`/global/ha_config`**:

```
root@hsm1mds-node1:~# mkdir -p /global/ha_config
```

5. Log in to the secondary cluster node as **`root`**.

   In the example, the secondary node is **`hsm1mds-node2`**. We log in using **`ssh`**:

```
root@hsm1mds-node1:~# ssh root@hsm1mds-node2
Password:
root@hsm1mds-node2:~#
```

6. On the secondary node, open the operating system's **`/etc/vfstab`** file in a text editor. Add an identical entry for the new UFS file system. Save the file, and close the editor.

   Note that the file-system entry is a single line:

```
root@hsm1mds-node2:~# 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       -
...
hsm1       -        /global/hsm1        samfs   -     no       shared
/dev/global/dsk/d10s0   /dev/global/rdsk/d10s0  /global/ha_config  ufs    5
no  global
:wq
root@hsm1mds-node1:~#
```

7. On the secondary node, create the same mount point.

   In the example, we create the **`/global/ha_config`** directory. Then we close the **`ssh`** session and resume working on the primary node:

```
root@hsm1mds-node2:~# mkdir -p /global/ha_config
root@hsm1mds-node2:~# exit
root@hsm1mds-node1:~#
```

8. On the primary node, mount the high-availability local file system. Use the command **`mount /global/`***`mountpoint`*, where *`mountpoint`* is the selected mount point directory.

   The command mounts the UFS file system on both nodes. In the example, we mount the file system on **`/global/ha_config`**:

```
root@hsm1mds-node1:~# mount /global/ha_config
root@hsm1mds-node1:~#
```

9. If you are configuring HA-HSM, move the Oracle HSM configuration and state files to the high-availability local file system.

10. If you are adding support for HA-NFS and/or HA-SAMBA to a completed HA-QFS or HA-HSM configuration, return to "Sharing HA-HSM or HA-QFS Configurations with HA-NFS or HA-SAMBA" on page 9-63.

Move Oracle HSM Configuration Files to the High-Availability Local File System

1. Log in to the primary node of the HA-HSM cluster as **root**.

   In the example, the primary node is **hsm1mds-node1**:

```
root@hsm1mds-node1:~#
```

2. On the primary node, create a subdirectory to hold Oracle HSM state and configuration data. Use the command **mkdir -p /global/***mountpoint***/***hasamdata*, where:

   - *mountpoint* is the selected mount point directory for the high-availability local file system.
   - *hasamdata* is the name of directory where the HA-HSM state information will be maintained.

   In the example, the mount point directory is **ha_config/** and the data directory is **hsm/**:

```
root@hsm1mds-node1:~# mkdir /global/ha_config/hsm
root@hsm1mds-node1:~#
```

3. In the state and configuration data directory on the primary node, create a subdirectory to hold Oracle HSM staging information. Use the command **mkdir -p /global/***mountpoint***/***hasamdata***/stager**.

   In the example, the mount point directory is **ha_config/** and the HA-HSM data directory is **hsm/**:

```
root@hsm1mds-node1:~# mkdir /global/ha_config/hsm/stager
root@hsm1mds-node1:~#
```

4. In the state and configuration data directory on the primary node, create a subdirectory to hold Oracle HSM archive catalogs. Use the command **mkdir -p /global/***mountpoint***/***hasamdata***/catalog**.

   In the example, the mount point directory is **ha_config/** and the HA-HSM data directory is **hsm/**:

```
root@hsm1mds-node1:~# mkdir /global/ha_config/hsm/catalog
root@hsm1mds-node1:~#
```

5. On the primary node, copy the **catalog/** and **stager/** directories from their default locations in **/var/opt/SUNWsamfs/** to a temporary location.

   In the example, we recursively copy the directories to **/var/tmp/**:

```
root@hsm1mds-node1:~# cp -r /var/opt/SUNWsamfs/catalog /var/tmp/catalog
root@hsm1mds-node1:~# cp -r /var/opt/SUNWsamfs/stager /var/tmp/stager
root@hsm1mds-node1:~#
```

6. On the primary node, delete the **catalog/** and **stager/** directories from **/var/opt/SUNWsamfs/**.

```
root@hsm1mds-node1:~# rm -rf /var/opt/SUNWsamfs/catalog
```

```
root@hsm1mds-node1:~# rm -rf /var/opt/SUNWsamfs/stager
root@hsm1mds-node1:~#
```

7. On the primary node, create a symbolic link from the default location of the catalog information to the new location in the high-availability UFS local file system. Use the command **ln -s /global/***mountpoint***/***hasamdata***/catalog /var/opt/SUNWsamfs/catalog**, where:

   - *mountpoint* is the name of the subdirectory where high-availability local file system is attaches to the node's root file system.
   - *hasamdata* is the name of directory where the HA-HSM state information will be maintained.
   - **/var/opt/SUNWsamfs/catalog** is the default location.

   The symbolic link will automatically redirect requests for catalog information to the new location. In the example, we create a **catalog** link that points to the new location **/global/ha_config/hsm/catalog**:

```
root@hsm1mds-node1:~# ln -s /global/ha_config/hsm/catalog
/var/opt/SUNWsamfs/catalog
root@hsm1mds-node1:~#
```

8. On the primary node, create a symbolic link from the default location of the staging information to the new location in the high-availability UFS local file system. Use the command **ln -s /global/***mountpoint***/***hasamdata***/stager /var/opt/SUNWsamfs/stager**, where:

   - *mountpoint* is the name of the subdirectory where high-availability local file system is attaches to the node's root file system.
   - *hasamdata* is the name of directory where the HA-HSM state information will be maintained.
   - **/var/opt/SUNWsamfs/stager** is the default location.

   The symbolic link will automatically redirect requests for stager information to the new location. In the example, we create a **stager** link that points to the new location **/global/ha_config/hsm/stager**:

```
root@hsm1mds-node1:~# ln -s /global/ha_config/hsm/stager
/var/opt/SUNWsamfs/stager
root@hsm1mds-node1:~#
```

9. On the primary node, make sure that symbolic links have replaced the default **/var/opt/SUNWsamfs/catalog** and **/var/opt/SUNWsamfs/stager** directories. Make sure that the links point to the new locations in the high-availability file system.

   In the example, the links are correct:

```
root@hsm1mds-node1:~# ls -l /var/opt/SUNWsamfs/catalog
lrwxrwxrwx ... /var/opt/SUNWsamfs/catalog -> /global/ha_config/hsm/catalog
root@hsm1mds-node1:~# ls -l /var/opt/SUNWsamfs/stager
lrwxrwxrwx ... /var/opt/SUNWsamfs/stager -> /global/ha_config/hsm/stager
root@hsm1mds-node1:~#
```

10. Copy the contents of the **catalog/** and **stager/** directories from the temporary location to the high-availability file system.

    In the example, we copy the **catalog/** and **stager/** directories from **/var/tmp/** to the new location, **/global/ha_config/stager**:

```
root@hsm1mds-node1:~# cp -rp /var/tmp/catalog/*  /var/opt/SUNWsamfs/hsm/catalog
```

```
root@hsm1mds-node1:~# cp -rp /var/tmp/stager/* /var/opt/SUNWsamfs/hsm/stager
root@hsm1mds-node1:~#
```

11. Log in to the secondary node of the HA-HSM cluster as **`root`**.

    In the example, we use **`ssh`** (secure shell) to log in to **`hsm1mds-node2`**, the secondary node:

    ```
    root@hsm1mds-node1:~# ssh root@hsm1mds-node2
    Password:
    root@hsm1mds-node2:~#
    ```

12. On the secondary node, create a symbolic link from the default location of the catalog information to the new location in the high-availability UFS local file system. Use the command **`ln -s /global/`***`mountpoint`***`/`***`hasamdata`***`/catalog /var/opt/SUNWsamfs/catalog`**, where:

    - *`mountpoint`* is the name of the subdirectory where high-availability local file system is attaches to the node's root file system.
    - *`hasamdata`* is the name of directory where the HA-HSM state information will be maintained.
    - **`/var/opt/SUNWsamfs/catalog`** is the default location.

    The symbolic link will automatically redirect requests for catalog information to the new location. In the example, we create a **`catalog`** link that points to the new location **`/global/ha_config/hsm/catalog`**:

    ```
    root@hsm1mds-node2:~# ln -s /global/ha_config/hsm/catalog
    /var/opt/SUNWsamfs/catalog
    root@hsm1mds-node2:~#
    ```

13. On the secondary node, create a symbolic link from the default location of the staging information to the new location in the high-availability UFS local file system. Use the command **`ln -s /global/`***`mountpoint`***`/`***`hasamdata`***`/stager /var/opt/SUNWsamfs/stager`**, where:

    - *`mountpoint`* is the name of the subdirectory where high-availability local file system is attaches to the node's root file system.
    - *`hasamdata`* is the name of directory where the HA-HSM state information will be maintained.
    - **`/var/opt/SUNWsamfs/stager`** is the default location.

    The symbolic link will automatically redirect requests for stager information to the new location. In the example, we create a **`stager`** link that points to the new location **`/global/ha_config/hsm/stager`**:

    ```
    root@hsm1mds-node2:~# ln -s /global/ha_config/hsm/stager
    /var/opt/SUNWsamfs/stager
    root@hsm1mds-node2:~#
    ```

14. On the secondary node, make sure that symbolic links have replaced the default **`/var/opt/SUNWsamfs/catalog`** and **`/var/opt/SUNWsamfs/stager`** directories. Make sure that the links point to the new locations in the high-availability file system.

    In the example, the links are correct. So we close the **`ssh`** session, and resume work on the primary node:

    ```
    root@hsm1mds-node2:~# ls -l /var/opt/SUNWsamfs/catalog
    lrwxrwxrwx ... /var/opt/SUNWsamfs/catalog -> /global/ha_config/hsm/catalog
    root@hsm1mds-node2:~# ls -l /var/opt/SUNWsamfs/stager
    ```

```
lrwxrwxrwx ... /var/opt/SUNWsamfs/stager -> /global/ha_config/hsm/stager
root@hsm1mds-node2:~# exit
root@hsm1mds-node1:~#
```

15. Next, configure failover for the high availability local file system.

Configure Failover for the High-Availability Local File System

1. See if the **SUNW.HAStoragePlus** resource type has been registered with the cluster. On the primary node of the HA-HSM cluster, use the command **clresourcetype show**.

In the example, the **SUNW.HAStoragePlus** has already been registered:

```
root@qfs1mds-1:~# clresourcetype show
=== Registered Resource Types ===
...
Resource Type:                  SUNW.HAStoragePlus:11
  RT_description:               HA Storage Plus
...
root@qfs1mds-1:~#
```

2. If the **SUNW.HAStoragePlus** resource type has not been registered with the cluster, register it now. Use the Solaris Cluster command **clresourcetype register SUNW.HAStoragePlus**.

```
root@hsm1mds-node1:~# clresourcetype register SUNW.HAStoragePlus
root@hsm1mds-node1:~#
```

3. If registration fails because the registration file cannot be found, place a symbolic link to the **/opt/SUNWsamfs/sc/etc/** directory in the directory where Solaris Cluster keeps resource-type registration files, **/opt/cluster/lib/rgm/rtreg/**.

You did not install Oracle Solaris Cluster software before installing Oracle HSM software. Normally, Oracle HSM automatically provides the location of the **SUNW.HAStoragePlus** registration file when it detects Solaris Cluster during installation. So you need to create a link manually.

```
root@hsm1mds-node1:~# cd /opt/cluster/lib/rgm/rtreg/
root@hsm1mds-node1:~# ln -s /opt/SUNWsamfs/sc/etc/SUNW.HAStoragePlus
SUNW.HAStoragePlus
root@hsm1mds-node1:~#
```

4. Create a new instance of the **SUNW.HAStoragePlus** resource type and associate it with a Solaris Cluster resource group. Use the command **clresource create -g** *groupname* **-t SUNW.HAStoragePlus -x FilesystemMountPoints=***mountpoint* **-x AffinityOn=TRUE** *resourcename*, where:

- *groupname* is the name that you have chosen for the HA-HSM resource group.
- **SUNW.HAStoragePlus** is the Solaris Cluster resource type that supports failover of local file systems.
- *mountpoint* is the mount point for the high-availability local file system that will hold the catalogs and stager files.
- *resourcename* is the name that you have chosen for the resource itself.

In the example, we create a resource named **halocal** of type **SUNW.HAStoragePlus**. We add the new resource to the resource group **hsmrg**. Then we configure the resource extension properties. We set **FilesystemMountPoints** to **/global/ha_config** and **AffinityOn** to **TRUE**:

```
root@hsm1mds-node1:~# clresource create -g hsmrg -t SUNW.HAStoragePlus
-x FilesystemMountPoints=/global/ha_config -x AffinityOn=TRUE halocal
root@hsm1mds-node1:~#
```

5. Next, configure failover of the QFS file-system metadata server.

Configure Failover of the QFS File-System Metadata Server

You configure failover of the metadata servers by creating a **SUNW.qfs** cluster resource, a resource type defined by the Oracle HSM software (see the **SUNW.qfs** man page for details). To create and configure the resource for an HA-HSM configuration, proceed as follows:

1. Log in to the primary cluster node of the HA-HSM cluster as **root**.

   In the example, the primary node is **hsm1mds-node1**:

   ```
   root@hsm1mds-node1:~#
   ```

2. See if the **SUNW.qfs** resource type has been registered with the cluster. Use the command **clresourcetype show**.

   In the example, the **SUNW.qfs** has already been registered:

   ```
   root@qfs1mds-1:~# clresourcetype show
   === Registered Resource Types ===
   ...
   Resource Type:                  SUNW.qfs:5
     RT_description:               SAM-QFS Agent on Solaris Cluster
   ...
   root@qfs1mds-1:~#
   ```

3. If the **SUNW.qfs** resource type has not been registered with the cluster, register it now. Use the command **clresourcetype register SUNW.qfs**.

   ```
   root@hsm1mds-node1:~# clresourcetype register SUNW.qfs
   root@hsm1mds-node1:~#
   ```

4. If registration fails because the registration file cannot be found, place a symbolic link to the **/opt/SUNWsamfs/sc/etc/** directory in the directory where Solaris Cluster keeps resource-type registration files, **/opt/cluster/lib/rgm/rtreg/**.

   Registration will fail if you did not install Oracle Solaris Cluster software before installing Oracle HSM software. Normally, Oracle HSM automatically provides the location of the **SUNW.qfs** registration file when it detects Solaris Cluster during installation. In the example, we create the link manually.

   ```
   root@hsm1mds-node1:~# cd /opt/cluster/lib/rgm/rtreg/
   root@hsm1mds-node1:~# ln -s /opt/SUNWsamfs/sc/etc/SUNW.qfs SUNW.qfs
   root@hsm1mds-node1:~#
   ```

5. In the new resource group, set up a logical host name for the active metadata server. Use the Solaris Cluster command **clreslogicalhostname create -g** *group-name virtualMDS*, where:

   - *group-name* is the name of the QFS resource group.
   - *virtualMDS* is the logical host name.

     The cluster maps the logical host name to the public network interface on the currently active node, so that user and client access to resources does not depend on the availability of a specific physical network interface.

Use the same logical host name that you used in the hosts files for the shared file system. In the example, we create the logical host name **hsm1mds** in the **hsmrg** resource group:

```
root@hsm1mds-node1:~# clreslogicalhostname create -g hsmrg hsm1mds
root@hsm1mds-node1:~#
```

6. Add the Oracle HSM file-system resources to the resource group. Use the command **clresource create -g** *groupname* **-t SUNW.qfs -x QFSFileSystem=***mount-point*, where:

   - *groupname* is the name that you have chosen for the HA-HSM resource group.
   - **SUNW.qfs** is the Solaris Cluster resource type that supports failover of the QFS file-system metadata servers.
   - *mount-point* is the mount point for the file system in the cluster, a subdirectory that is not directly beneath the system root directory.

     Mounting a shared QFS file system immediately under root can cause failover issues when using the **SUNW.qfs** resource type.
   - *resource-name* is the name that you have chosen for the resource itself.

   In the example, we create a resource named **haqfs** of type **SUNW.qfs** in the resource group **hsmrg**. We set the **SUNW.qfs** extension property **QFSFileSystem** to the **/global/ha_hsmfs/hsm1** mount point. We set the standard property **Resource_ dependencies** to **hsm1mds**, the logical host name that represents the active metadata server:

```
root@hsm1mds-node1:~# clresource create -g hsmrg -t SUNW.qfs
-x QFSFileSystem=/global/ha_hsmfs/hsm1 -y Resource_dependencies=hsm1mds haqfs
root@hsm1mds-node1:~#
```

7. Next, configure failover of the Oracle Hierarchical Storage Manager application.

Configure Failover of the Oracle Hierarchical Storage Manager Application

You configure failover of the Oracle Hierarchical Storage Manager application by creating an Oracle HSM **SUNW.hasam** resource. This resource type coordinates the orderly shut-down and restart of Oracle HSM processes.

To configure failover of the Oracle HSM application, proceed as follows:

1. Log in to the primary cluster node of the HA-HSM cluster as **root**.

   In the example, the primary node is **hsm1mds-node1**:

```
root@hsm1mds-node1:~#
```

2. Define the resource type, **SUNW.hasam**, for the Solaris Cluster software. Use the command **clresourcetype register SUNW.hasam**.

```
root@hsm1mds-node1:~# clresourcetype register SUNW.hasam
root@hsm1mds-node1:~#
```

3. Add the Oracle HSM **SUNW.hasam** resource to the resource group. Use the command **clresource create -g** *groupname* **-t SUNW.hasam -x QFSName=***fs-name* **-x CatalogFileSystem=***mount-point resource-name*, where:

   - *groupname* is the name that you have chosen for the HA-HSM resource group.
   - **SUNW.hasam** is the Solaris Cluster resource type that supports failover of the Oracle Hierarchical Storage Manager application.

- *mount-point* is the mount point for the global file system that holds the Oracle HSM archive catalogs.
- *resource-name* is the name that you have chosen for the resource itself.

In the example, we create a resource named **hahsm** of type **SUNW.hasam** in the resource group **hsmrg**. We set the **SUNW.hasam** extension property **QFSName** to the QFS file-system name specified in the **mcf** file, **hsm1**. We set the **SUNW.hasam** extension property **CatalogFileSystem** to the **/global/ha_config** mount point.:

```
root@hsm1mds-node1:~# clresource create -g hsmrg -t SUNW.hasam -x QFSName=hsm1
-x CatalogFileSystem=/global/ha_config hahsm
root@hsm1mds-node1:~#
```

4. Next, define the cluster resource dependencies for the HA-HSM solution.

Define the Cluster Resource Dependencies for the HA-HSM Solution

1. Log in to the primary cluster node of the HA-HSM cluster as **root**.

In the example, the primary node is **hsm1mds-node1**:

```
root@hsm1mds-node1:~#
```

2. The cluster should not make the QFS file system available if the high-availability local file system is unavailable. So make the **SUNW.qfs** resource depend upon the **SUNW.HAStoragePlus** resource. Use the Solaris Cluster command **clresource set -p Resource_dependencies=***dependency resource-name*, where:

- *dependency* is name of the **SUNW.HAStoragePlus** resource.
- *resource-name* is the name of the **SUNW.qfs** resource.

```
root@hsm1mds-node1:~# clresource set -p Resource_dependencies=halocal haqfs
root@hsm1mds-node1:~#
```

3. The cluster should not make the QFS file system available if users and applications cannot reach the active metadata server. So make the **SUNW.qfs** resource depend on the logical host name. Use the Solaris Cluster command **clresource set -p Resource_dependencies=***virtualMDS resource-name*, where:

- *virtualMDS* is the cluster's logical host name for the active file-system metadata server.

  The logical host name always points to the public network interface on the active metadata server, so that user and client access to the file system does not depend on the availability of a specific physical network interface.
- *resource-name* is the name of the **SUNW.qfs** resource.

In the example, the logical host name that we created when we set up the **SUNW.qfs** resource is **hsm1mds**. The resource itself is named **haqfs**:

```
root@hsm1mds-node1:~# clresource set -p Resource_dependencies=hsm1mds haqfs
root@hsm1mds-node1:~#
```

4. Make the **SUNW.hasam** resource depend on the **SUNW.qfs** resource. Use the Solaris Cluster command **clresource set -p Resource_dependencies=***dependency resource-group-name*, where:

- *dependency* is name of the **SUNW.qfs** resource.
- *resource-group-name* is the name of the **SUNW.hasam** resource.

In the example, we make the **SUNW.hasam** resource **hahsm** depend on the **SUNW.qfs** resource **haqfs**:

```
root@hsm1mds-node1:~# clresource set -p Resource_dependencies=haqfs hahsm
root@hsm1mds-node1:~#
```

5. Next, bring the HA-HSM resource group online and test the configuration.

Bring the HA-HSM Resource Group Online and Test the Configuration

1. Log in to the primary cluster node of the HA-HSM cluster as **root**.

   In the example, the primary node is **hsm1mds-node1**:

```
root@hsm1mds-node1:~#
```

2. Bring the resource group online. Use the Solaris Cluster commands **clresourcegroup manage** *groupname*, and **clresourcegroup online -emM** *groupname*, where *groupname* is the name of the HA-HSM resource group.

   In the example, we bring the **hsmrg** resource group online:

```
root@hsm1mds-node1:~# clresourcegroup manage hsmrg
root@hsm1mds-node1:~# clresourcegroup online -emM hsmrg
root@hsm1mds-node1:~#
```

3. Make sure that the HA-HSM resource group is online. Use the Solaris Cluster **clresourcegroup status** command.

   In the example, the **hsmrg** resource group is **online** on the primary node, **hsm1mds-node1**:

```
root@hsm1mds-node1:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name  Node Name       Suspended   Status
----------  -------------   ---------   ------
hsmrg       hsm1mds-node1   No          Online
            hsm1mds-node2   No          Offline
root@hsm1mds-node1:~#
```

4. Next, make sure that the resource group fails over correctly. Move the resource group to the secondary node. Use the Solaris Cluster command **clresourcegroup switch -n** *node2 groupname*, where *node2* is the name of the secondary node and *groupname* is the name that you have chosen for the HA-HSM resource group. Then use **clresourcegroup status** to check the result.

   In the example, we move the **hsmrg** resource group to **hsm1mds-node2** and confirm that the resource group comes online on the specified node:

```
root@hsm1mds-node1:~# clresourcegroup switch -n hsm1mds-node2 hsmrg
root@hsm1mds-node1:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name  Node Name       Suspended   Status
----------  -------------   ---------   ------
hsmrg       hsm1mds-node1   No          Offline
            hsm1mds-node2   No          Online
root@hsm1mds-node1:~#
```

5. Move the resource group back to the primary node. Use the Solaris Cluster command **clresourcegroup switch -n** *node1 groupname*, where *node1* is the name of the primary node and *groupname* is the name that you have chosen for the HA-HSM resource group. Then use **clresourcegroup status** to check the result.

In the example, we successfully move the **hsmrg** resource group back to **hsm1mds-node1**:

```
root@hsm1mds-node1:~# clresourcegroup switch -n hsm1mds-node1 hsmrg
root@hsm1mds-node1:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name  Node Name       Suspended   Status
----------  -------------   ---------   ------
hsmrg       hsm1mds-node1   No          Online
            hsm1mds-node2   No          Offline
root@hsm1mds-node1:~#
```

6. If you need to share the HA-QFS file system with HA-NFS or HA-SAMBA, go to "Sharing HA-HSM or HA-QFS Configurations with HA-NFS or HA-SAMBA" on page 9-63.
7. If you plan on using the sideband database feature, go to "Configuring the Reporting Database" on page 10-1.
8. Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

High-Availability QFS Shared File Systems and Oracle RAC

In the Solaris Cluster-Oracle Real Application Cluster (SC-RAC) configuration, Solaris Cluster software manages a QFS shared file system as a **SUNW.qfs** resource mounted on nodes that also host Oracle Database and Oracle Real Application Cluster (RAC) software. All nodes are configured as QFS servers, with one the active metadata server and the others potential metadata servers. If the active metadata server node fails, Solaris Cluster software automatically activates a potential metadata server on a healthy node and initiates failover. I/O is coordinated through Oracle RAC, and the QFS file system is shared and already mounted on all nodes. So access to the data remains uninterrupted.

In the SC-RAC configuration, the 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 cluster nodes. Since file-system integrity is assured under RAC, the QFS potential metadata servers can perform I/O as clients of the shared file system. For additional information, see the Oracle Solaris Cluster Data Service documentation for Oracle Real Application Clusters in *Oracle Solaris Cluster Online Documentation Library*.

To configure a SC-RAC file system, carry out the tasks below:

- Create QFS hosts files on all SC-RAC cluster nodes.
- Configure the active QFS metadata server on the primary SC-RAC cluster node.
- Configure a potential QFS metadata server on the remaining SC-RAC cluster nodes.
- Configure failover of the SC-RAC metadata server.

Note that the SC-RAC configuration does *not* support file sharing via the Network File System (NFS) or the Server Message Block/Common Internet File System (SMB/CIFS) and does *not* support HA-NFS or HA-SAMBA.

Create QFS Hosts Files on All SC-RAC Cluster Nodes

In a QFS shared file system, you must configure a hosts file on the metadata servers, so that all hosts can access the metadata for the 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. Log in to the primary cluster node of the SC-RAC cluster as **root**.

   In the example, the primary node is **qfs1rac-node1**:

   ```
   root@qfs1rac-node1:~#
   ```

2. Display the cluster configuration. Use the command **/usr/global/bin/cluster show**. In the output, locate the record for each **Node Name**, and then note the **privatehostname**, the **Transport Adapter** name, and the **ip_address** property of each network adapter.

   In the examples, each node has two network interfaces, **qfe3** and **hme0**:

   - The **hme0** adapters have IP addresses on the private network that the cluster uses for internal communication between nodes. The Solaris Cluster software assigns a **privatehostname** corresponding to each private address.

     By default, the private hostname of the primary node is **clusternode1-priv**, and the private hostname of the secondary node is **clusternode2-priv**.

   - The **qfe3** adapters have public IP addresses and public hostnames—**qfs1rac-node1** and **qfs1rac-node2**—that the cluster uses for data transport.

   Note that the display has been abbreviated using ellipsis (**...**) marks:

   ```
   root@qfs1rac-node1:~# cluster show
   ...
     === Cluster Nodes ===
     Node Name:                                  qfs1rac-node1...
       privatehostname:                             clusternode1-priv...
       Transport Adapter List:                      qfe3, hme0...
       Transport Adapter:                        qfe3...
         Adapter Property(ip_address):              172.16.0.12...
       Transport Adapter:                        hme0...
         Adapter Property(ip_address):              10.0.0.129...
     Node Name:                                  qfs1rac-node2...
       privatehostname:                             clusternode2-priv...
       Transport Adapter List:                      qfe3, hme0...
         Adapter Property(ip_address):              172.16.0.13...
       Transport Adapter:                        hme0
         Adapter Property(ip_address):              10.0.0.122...
     Node Name:                                  qfs1rac-node3...
       privatehostname:                             clusternod3-priv...
       Transport Adapter List:                      qfe3, hme0...
         Adapter Property(ip_address):              172.16.0.33...
       Transport Adapter:                        hme0
         Adapter Property(ip_address):              10.0.0.092
   ```

3. Using a text editor, create the file **/etc/opt/SUNWsamfs/hosts.***family-set-name*, where *family-set-name* is the family-set name that the **/etc/opt/SUNWsamfs/mcf** file assigns to the file-system equipment.

   In the example, we create the file **hosts.qfs1rac** using the **vi** text editor. We add optional column headings, starting each line with a hash sign (**#**) to indicate a comment:

   ```
   root@qfs1rac-node1:~# vi /etc/opt/SUNWsamfs/hosts.qfs1rac
   # /etc/opt/SUNWsamfs/hosts.qfs1rac
   #                                               Server   On/  Additional
   ```

```
#Host Name     Network Interface                Ordinal  Off  Parameters
#------------  -------------------------------  -------  ---  ----------
```

4. In the first column of the table, enter the hostnames of the primary and secondary metadata server nodes followed by some spaces. Place each entry on a separate line.

   In a hosts file, the lines are rows (records) and spaces are column (field) separators. In the example, the Host Name column of the first two rows lists the hostnames of the cluster nodes **qfs1rac-node1**, **qfs1rac-node2**, and **qfs1rac-node3**.

```
#                                               Server   On/  Additional
#Host Name     Network Interface                Ordinal  Off  Parameters
#------------  -------------------------------  -------  ---  ----------
qfs1rac-node1
qfs1rac-node2
qfs1rac-node3
```

5. In the second column of each line, start supplying **Network Interface** information for host **Host Name**. Enter each SC-RAC cluster node's Solaris Cluster private hostname or private network address followed by a comma.

   The SC-RAC server nodes use the private hostnames for server-to-server communications within the high-availability cluster. In the example, we use the private hostnames **clusternode1-priv**, **clusternode2-priv**, and **clusternode3-priv**, which are the default names assigned by the Solaris Cluster software:

```
#                                               Server   On/  Additional
#Host Name     Network Interface                Ordinal  Off  Parameters
#------------  -------------------------------  -------  ---  ----------
qfs1rac-node1  clusternode1-priv,
qfs1rac-node2  clusternode2-priv,
qfs1rac-node3  clusternode3-priv,
```

6. Following the comma in the second column of each line, enter the public hostname for the active metadata server followed by spaces.

   The SC-RAC server nodes use the public data network to communicate with the clients, all of which reside outside the cluster. Since the IP address and hostname of the active metadata server changes during failover (from **qfs1rac-node1** to **qfs1rac-node2**, for example), we represent the active server with a logical host name, **qfs1rac-mds**. Later, we will configure the Solaris Cluster software to always route requests for **qfs1rac-mds** to the node that currently hosts the active metadata server:

```
#                                               Server   On/  Additional
#Host Name     Network Interface                Ordinal  Off  Parameters
#------------  -------------------------------  -------  ---  ----------
qfs1rac-node1  clusternode1-priv,qfs1rac-mds
qfs1rac-node2  clusternode2-priv,qfs1rac-mds
qfs1rac-node3  clusternode3-priv,qfs1rac-mds
```

7. In the third column of each line, enter the ordinal number of the server (**1** for the active metadata server, and **2** for the potential metadata server), followed by spaces.

   In the example, the primary node, **qfs1rac-node1**, is the active metadata server. So it is ordinal **1**. The second node, **qfs1rac-node2**, is ordinal **2**, and so on:

```
#                                               Server   On/  Additional
```

```
#Host Name     Network Interface                Ordinal  Off  Parameters
#------------  -------------------------------  -------  ---  ----------
qfs1rac-node1  clusternode1-priv,qfs1rac-mds    1
qfs1rac-node2  clusternode2-priv,qfs1rac-mds    2
qfs1rac-node3  clusternode3-priv,qfs1rac-mds    3
```

8. In the fourth column of each line, enter **0** (zero), followed by spaces.

   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 **samsharefs** man page).

```
#                                               Server   On/  Additional
#Host Name     Network Interface                Ordinal  Off  Parameters
#------------  -------------------------------  -------  ---  ----------
qfs1rac-node1  clusternode1-priv,qfs1rac-mds    1        0
qfs1rac-node2  clusternode2-priv,qfs1rac-mds    2        0
qfs1rac-node3  clusternode3-priv,qfs1rac-mds    3        0
```

9. In the fifth column of the line for the primary node, enter the keyword **server**. Save the file and close the editor.

   The server keyword identifies the default, active metadata server:

```
#                                               Server   On/  Additional
#Host Name     Network Interface                Ordinal  Off  Parameters
#------------  -------------------------------  -------  ---  ----------
qfs1rac-node1  clusternode1-priv,qfs1rac-mds    1        0    server
qfs1rac-node2  clusternode2-priv,qfs1rac-mds    2        0
qfs1rac-node3  clusternode3-priv,qfs1rac-mds    2        0
:wq
root@qfs1rac-node1:~#
```

10. Place a copy of the global **/etc/opt/SUNWsamfs/hosts.***family-set-name* file on each node in the SC-RAC cluster.

11. Now, configure an active QFS metadata server on the primary SC-RAC cluster node.

Configure the Active QFS Metadata Server on the Primary SC-RAC Cluster Node

- Configure the primary SC-RAC cluster node to use hardware RAID storage.
- Configure the primary SC-RAC cluster node to use software RAID storage.

Configure a Primary SC-RAC Cluster Node to Use Hardware RAID Storage

1. Select the cluster node that will serve as both the primary node for the SC-RAC cluster and the active metadata server for the QFS shared file system. Log in as **root**.

   In the example, the primary node is **qfs1rac-node1**:

```
root@qfs1rac-node1:~#
```

2. Select the global storage devices that will be used for the QFS file system. Use the command **/usr/global/bin/cldevice list -v**.

   Solaris Cluster software assigns unique Device Identifiers (DIDs) to all devices that attach to the cluster nodes. *Global* devices are accessible from all nodes in the cluster, while *local* devices are accessible only from the hosts that mount them. Global devices remain accessible following failover. Local devices do not.

In the example, note that devices **d1**, **d2**, **d6**, **d7**, and **d8** are not accessible from all nodes. So we select from devices **d3**, **d4**, and **d5** when configuring the high-availability QFS shared file system:

```
root@qfs1rac-node1:~# cldevice list -v
DID Device          Full Device Path
----------          ----------------
d1                  qfs1rac-node1:/dev/rdsk/c0t0d0
d2                  qfs1rac-node1:/dev/rdsk/c0t6d0
d3                  qfs1rac-node1:/dev/rdsk/c1t1d0
d3                  qfs1rac-node2:/dev/rdsk/c1t1d0
d3                  qfs1rac-node3:/dev/rdsk/c1t1d0
d4                  qfs1rac-node1:/dev/rdsk/c1t2d0
d4                  qfs1rac-node2:/dev/rdsk/c1t2d0
d4                  qfs1rac-node3:/dev/rdsk/c1t2d0
d5                  qfs1rac-node1:/dev/rdsk/c1t3d0
d5                  qfs1rac-node2:/dev/rdsk/c1t3d0
d5                  qfs1rac-node3:/dev/rdsk/c1t3d0
d6                  qfs1rac-node2:/dev/rdsk/c0t0d0
d7                  qfs1rac-node2:/dev/rdsk/c0t1d0
d8                  qfs1rac-node3:/dev/rdsk/c0t1d0
```

3. Create a shared, high-performance **ma** file system that uses **mr** data devices. In a text editor, open the **/etc/opt/SUNWsamfs/mcf** file.

   In the example, we configure the file system **qfs1rac**. We configure device **d3** as the metadata device (equipment type **mm**), and use **d4** and **d5** as data devices (equipment type **mr**):

```
root@qfs1rac-node1:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment           Equipment  Equipment  Family   Device   Additional
# Identifier          Ordinal    Type       Set      State    Parameters
#------------------   ---------  ---------  -------  ------   --------------
qfs1rac               100        ma         qfs1rac  -
/dev/did/dsk/d3s0     101        mm         qfs1rac  -
/dev/did/dsk/d4s0     102        mr         qfs1rac  -
/dev/did/dsk/d5s0     103        mr         qfs1rac  -
...
```

4. In the **/etc/opt/SUNWsamfs/mcf** file, enter the **shared** parameter in the **Additional Parameters** column of the file system entry. Save the file.

```
root@qfs1rac-node1:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment           Equipment  Equipment  Family   Device   Additional
# Identifier          Ordinal    Type       Set      State    Parameters
#------------------   ---------  ---------  -------  ------   --------------
qfs1rac               100        ma         qfs1rac  -        shared
/dev/did/dsk/d3s0     101        mm         qfs1rac  -
/dev/did/dsk/d4s0     102        mr         qfs1rac  -
/dev/did/dsk/d5s0     103        mr         qfs1rac  -
...
:wq
root@qfs1rac-node1:~#
```

5. Check the **mcf** file for errors. Use the command **/opt/SUNWsamfs/sbin/sam-fsd**, and correct any errors found.

The **sam-fsd** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error. In the example, we check the **mcf** file on host **qfs1rac-node1**:

```
root@qfs1rac-node1:~# sam-fsd
...
Would start sam-archiverd()
Would start sam-stagealld()
Would start sam-stagerd()
Would start sam-amld()
root@qfs1rac-node1:~#
```

6. Create the file system. Use the command **/opt/SUNWsamfs/sbin/sammkfs -S** *family-set-name*, where *family-set-name* is the equipment identifier for the file-system.

   The **sammkfs** command reads the **hosts.***family-set-name* and **mcf** files and creates a shared file system with the specified properties.

```
root@qfs1rac-node1:~# sammkfs -S qfs1rac
Building 'qfs1rac' will destroy the contents of devices:
  ...
Do you wish to continue? [y/N]yes ...
root@qfs1rac-node1:~#
```

7. Open the operating system's **/etc/vfstab** file in a text editor, and start a line for the new file system. Enter the file system name in the first column, spaces, a hyphen in the second column, and more spaces.

   In the example, we use the **vi** text editor. We start a line for the **qfs1rac** file system. The hyphen keeps the operating system from attempting to check file system integrity using UFS tools:

```
root@qfs1rac-node1:~# 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       -
...
qfs1rac    -
```

8. In the third column of the **/etc/vfstab** file, enter the mount point of the file system relative to the cluster. Specify a subdirectory that is not directly beneath the system root directory.

   Mounting a shared QFS file system immediately under root can cause failover issues when using the **SUNW.qfs** resource type. In the example, the mount point for the **qfs1rac** file system is **/global/sc-rac/qfs1rac**:

```
#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       -
...
qfs1rac    -        /global/sc-rac/qfs1rac
```

9. Enter the file-system type, **samfs**, in the fourth column, **-** (*hyphen*) in the fifth column, and **no** in the sixth column.

   ```
   #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       -
   ...
   qfs1rac    -        /global/sc-rac/qfs1rac    samfs   -     no
   :wq
   root@qfs1rac-node1:~#
   ```

10. In the seventh column of the **/etc/vfstab** file, enter the mount options listed below. Then save the file, and close the editor.

    The following mount options are recommended for the SC-RAC cluster configuration. They can be specified here, in **/etc/vfstab**, or in the file **/etc/opt/SUNWsamfs/samfs.cmd**, if more convenient:

    - **shared**
    - **stripe=1**
    - **sync_meta=1**
    - **mh_write**
    - **qwrite**
    - **forcedirectio**
    - **notrace**
    - **rdlease=300**
    - **wrlease=300**
    - **aplease=300**

    In the example, the list has been abbreviated to fit the page layout:

    ```
    #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      -
    ...
    qfs1rac   -       /global/sc-rac/qfs1rac samfs  -    no      shared,...=300
    :wq
    root@qfs1rac-node1:~#
    ```

11. Create the mount point for the high-availability shared file system.

    ```
    root@qfs1rac-node1:~# mkdir -p /global/sc-rac/qfs1rac
    root@qfs1rac-node1:~#
    ```

12. Mount the high-availability shared file system on the primary node.

    ```
    root@qfs1rac-node1:~# mount /global/sc-rac/qfs1rac
    root@qfs1rac-node1:~#
    ```

13. Go to "Configure a Potential QFS Metadata Server on the Remaining SC-RAC Cluster Nodes" on page 9-58.

Configure the Primary SC-RAC Cluster Node to Use Software RAID Storage

A high-availability file system must store data and metadata on redundant primary storage devices. Redundant disk array hardware can provide this redundancy using RAID-1 or RAID-10 for metadata and RAID-5 for data. But if you need to use plain, dual-port, SCSI disk devices or a JBOD (*just a bunch of disks*) array as primary storage, you need to provide the required redundancy in software.

For this reason, the SC-RAC configuration supports software RAID configurations based on Oracle Solaris Volume Manager (SVM) multi-owner disk sets. This section outlines the basic steps that you need to take when setting up this variant of the SC-RAC file-system configuration.

Note that you should use Solaris Volume Manager purely for managing the redundant storage array. Do not concatenate storage on separate devices. Doing so distributes I/O to the component devices inefficiently and degrades QFS file-system performance.

Carry out the following tasks:

- Install Solaris Volume Manager on Solaris 11+.
- Create Solaris Volume Manager multi-owner disk groups.
- Create mirrored volumes for the QFS data and metadata.
- Create a QFS shared file system on the SC-RAC cluster using mirrored volumes.

Install Solaris Volume Manager on Solaris 11+

Solaris Volume Manager (SVM) is no longer included with the Solaris operating system starting with Solaris 11. But the Solaris Cluster 4 software continues to support Solaris Volume Manager. So, to use the software, you must download and install the required packages. For each node in the cluster, proceed as follows:

1. Log in to the node as **`root`**.

   In the examples below, we configure cluster node **`qfs2rac-node1`** using the Solaris Image Packaging System (IPS):

   ```
   root@qfs2rac-node1:~#
   ```

2. Check for locally available Solaris Volume Manager (SVM) packages. Use the command the **`pkg info svm`**.

   ```
   root@qfs2rac-node1:~# pkg info svm
   pkg: info: no packages matching the following patterns you specified are
   installed on the system.  Try specifying -r to query remotely:
           svm
   root@qfs2rac-node1:~#
   ```

3. If no packages are found locally, check the Solaris Image Packaging System (IPS) repository. Use the command **`pkg -r svm`**.

   ```
   root@qfs2rac-node1:~# pkg -r svm
             Name: storage/svm
          Summary: Solaris Volume Manager
      Description: Solaris Volume Manager commands
         Category: System/Core
            State: Not installed
        Publisher: solaris
          Version: 0.5.11
    Build Release: 5.11
           Branch: 0.175.0.0.0.2.1
   Packaging Date: October 19, 2011 06:42:14 AM
   ```

```
          Size: 3.48 MB
          FMRI:
pkg://solaris/storage/svm@0.5.11,5.11-0.175.0.0.0.2.1:20111019T064214Z
root@qfs2rac-node1:~#
```

4. Install the package. Use the command **pkg install storage/svm**:

```
root@qfs2rac-node1:~# pkg install storage/svm
           Packages to install:   1
       Create boot environment:  No
Create backup boot environment: Yes
            Services to change:   1
DOWNLOAD      PKGS       FILES    XFER (MB)
Completed      1/1     104/104      1.6/1.6
PHASE            ACTIONS
Install Phase    168/168
PHASEITEMS
Package State Update Phase         1/1
Image State Update Phase           2/2
root@qfs2rac-node1:~#
```

5. When the installation finishes, check the location of **metadb**. Use the command the **which metadb**.

```
root@qfs2rac-node1:~# which metadb
/usr/sbin/metadb
root@qfs2rac-node1:~#
```

6. Check the installation. Use the command **metadb**.

```
root@qfs2rac-node1:~# metadb
root@qfs2rac-node1:~#
```

7. If **metadb** returns an error, see if the **kernel/drv/md.conf** file exists.

```
root@qfs2rac-node1:~# metadb
metadb: <HOST>: /dev/md/admin: No such file or directory
root@qfs2rac-node1:~# ls -l /kernel/drv/md.conf
-rw-r--r--   1 root     sys          295 Apr 26 15:07 /kernel/drv/md.conf
root@qfs2rac-node1:~#
```

8. If the **kernel/drv/md.conf** file does not exist, create it. Make **root** the file's owner, and make **sys** the group owner. Set permissions to **644**.

   In the example, we create the file with the **vi** editor. The content of the file should look like this:

```
root@qfs2rac-node1:~# vi kernel/drv/md.conf
###################################################
#pragma ident   "@(#)md.conf    2.1   00/07/07 SMI"
#
# Copyright (c) 1992-1999 by Sun Microsystems, Inc.
# All rights reserved.
#
name="md" parent="pseudo" nmd=128 md_nsets=4;
####################################################
:wq
root@qfs2rac-node1:~# chown root:sys kernel/drv/md.conf
root@qfs2rac-node1:~# chmod 644
root@qfs2rac-node1:~#
```

9. Dynamically rescan the **md.conf** file and make sure that the device tree is updated. Use the command **update_drv -f md**:

   In the example, the device tree is updated. So Solaris Volume Manager is installed:

```
root@qfs2rac-node1:~# update_drv -f md
root@qfs2rac-node1:~# ls -l  /dev/md/admin
lrwxrwxrwx   1 root root 31 Apr 20 10:12 /dev/md/admin ->
../../devices/pseudo/md@0:admin
root@qfs2rac-node1:~#
```

10. Next, create Solaris Volume Manager multi-owner disk groups.

Create Solaris Volume Manager Multi-Owner Disk Groups 1.Log in to all nodes in the SC-RAC configuration as **root**.

In the example, we log in to node **qfs2rac-node1**. We then open new terminal windows and use **ssh** to log in to nodes **qfs2rac-node2** and **qfs2rac-node3**:

```
root@qfs2rac-node1:~#

root@qfs2rac-node1:~# ssh root@qfs2rac-node2
Password:
root@qfs2rac-node2:~#

root@qfs2rac-node1:~# ssh root@qfs2rac-node3
Password:
root@qfs2rac-node3:~#
```

2. If you are using Oracle Solaris Cluster 4.*x* on Solaris 11.*x* or later and have not already done so, install Solaris Volume Manager on each node before proceeding further.

   Starting with Solaris 11, Solaris Volume Manager is not installed by default.

3. On each node, attach a new state database device and create three state database replicas. Use the command **metadb -a -f -c3** *device-name*, where *device-name* is a physical device name of the form **c***X***t***Y***d***Y***s***Z*.

   Do not use Solaris Cluster Device Identifiers (DIDs). Use the physical device name. In the example, we create state database devices on all three cluster nodes:

```
root@qfs2rac-node1:~# metadb -a -f -c3 /dev/rdsk/c0t0d0

root@qfs2rac-node2:~# metadb -a -f -c3 /dev/rdsk/c0t6d0

root@qfs2rac-node3:~# metadb -a -f -c3 /dev/rdsk/c0t4d0
```

4. Create a Solaris Volume Manager multi-owner disk group on one node. Use the command **metaset -s***diskset* **-M -a -h** *host-list*, where *host-list* is a space-delimited list of owners.

   Solaris Volume Manager supports up to four hosts per disk set. In the example, we create the disk group **datadisks** on **qfs2rac-node1** and specify the three nodes **qfs2rac-node1**, **qfs2rac-node2**, and **qfs2rac-node3** as owners:

```
root@qfs2rac-node1:~# metaset -s datadisks -M -a -h qfs2rac-node1 qfs2rac-node2
qfs2rac-node3
```

5. List the devices on one of the nodes. Use the Solaris Cluster command **cldevice list -n -v**.

```
root@qfs2rac-node1:~# cldevice list -n -v
```

```
DID Device  Full Device Path
----------  ----------------
d13         qfs2rac-node1:/dev/rdsk/c6t600C0FF00000000000332B62CF3A6B00d0
d14         qfs2rac-node1:/dev/rdsk/c6t600C0FF0000000000876E950F1FD9600d0
d15         qfs2rac-node1:/dev/rdsk/c6t600C0FF0000000000876E9124FAF9C00d0
...
root@qfs2rac-node1:~#
```

6. In the output of the **cldevice list -n -v** command, select the devices that will be mirrored.

   In the example, we select four pairs of devices for four mirrors: **d21** and **d13**, **d14** and **d17**, **d23** and **d16**, and **d15** and **d19**.

```
root@qfs2rac-node1:~# cldevice list -n -v
DID Device  Full Device Path
----------  ----------------
d13         qfs2rac-node1:/dev/rdsk/c6t600C0FF00000000000332B62CF3A6B00d0
d14         qfs2rac-node1:/dev/rdsk/c6t600C0FF0000000000876E950F1FD9600d0
d15         qfs2rac-node1:/dev/rdsk/c6t600C0FF0000000000876E9124FAF9C00d0
d16         qfs2rac-node1:/dev/rdsk/c6t600C0FF00000000000332B28488B5700d0
d17         qfs2rac-node1:/dev/rdsk/c6t600C0FF000000000086DB474EC5DE900d0
d18         qfs2rac-node1:/dev/rdsk/c6t600C0FF0000000000876E975EDA6A000d0
d19         qfs2rac-node1:/dev/rdsk/c6t600C0FF000000000086DB47E331ACF00d0
d20         qfs2rac-node1:/dev/rdsk/c6t600C0FF0000000000876E9780ECA8100d0
d21         qfs2rac-node1:/dev/rdsk/c6t600C0FF000000000004CAD5B68A7A100d0
d22         qfs2rac-node1:/dev/rdsk/c6t600C0FF000000000086DB43CF85DA800d0
d23         qfs2rac-node1:/dev/rdsk/c6t600C0FF000000000004CAD7CC3CDE500d0
d24         qfs2rac-node1:/dev/rdsk/c6t600C0FF000000000086DB4259B272300d0
....
root@qfs2rac-node1:~#
```

7. Add the selected devices to the disk set on the same node. Use the command **metaset -a** *devicelist*, where *devicelist* is a space-delimited list of one or more cluster device identifiers.

   In the example, we add the listed disks to multi-owner disk set **dataset1**:

```
root@qfs2rac-node1:~# metaset -s dataset1 -M -a -h /dev/did/rdsk/d21
/dev/did/rdsk/d13 /dev/did/rdsk/d14 /dev/did/rdsk/d17 /dev/did/rdsk/d23
/dev/did/rdsk/d16 /dev/did/rdsk/d15 /dev/did/rdsk/d19
root@qfs2rac-node1:~#
```

8. Next, create mirrored volumes for the QFS data and metadata.

Create Mirrored Volumes for the QFS Data and Metadata

1. To keep the relationships between components clear, decide on a naming scheme for the RAID-0 logical volumes and RAID-1 mirrors that you will create.

   Commonly, RAID-1 mirrors are named **d***n*, where *n* is an integer. The RAID-0 volumes that make up the RAID-1 mirrors are named **d***nX*, where *X* is an integer representing the device's position within the mirror (usually **0** or **1** for a two-way mirror).

   In the examples throughout this procedure, we create two-way RAID-1 mirrors from pairs of RAID-0 logical volumes. So we name the mirrors **d1**, **d2**, **d3**, **d4**, and so on. Then we name each pair of RAID-0 volumes for the RAID-1 mirror that includes it: **d10** and **d11**, **d20** and **d21**, **d30** and **d31**, **d40** and **d41**, etc.

2. Log in to the node where you created the multi-owner disk set. Log in as **root**.

In the examples above, we created the disk set on **qfs2rac-node1**:

```
root@qfs2rac-node1:~#
```

3. Create the first RAID-0 logical volume. Use the command **metainit -s** *diskset-name device-name number-of-stripes components-per-stripe component-names*, where:

   - *diskset-name* is the name that you have chosen for the disk set.
   - *device-name* is the name that you have chosen for the RAID-0 logical volume.
   - *number-of-stripes* is **1**.
   - *components-per-stripe* is **1**.
   - *component-name* is the device name of the disk set component to use in the RAID-0 volume.

   In the example, we use the cluster (DID) device **/dev/did/dsk/d21s0** in multi-owner disk set **dataset1** to create RAID-0 logical volume **d10**:

   ```
   root@qfs2rac-node1:~# metainit -s dataset1 d10 1 1 /dev/did/dsk/d21s0
   root@qfs2rac-node1:~#
   ```

4. Create the remaining RAID-0 logical volumes.

   ```
   root@qfs2rac-node1:~# metainit -s dataset1 d11 1 1 /dev/did/dsk/d13s0
   root@qfs2rac-node1:~# metainit -s dataset1 d20 1 1 /dev/did/dsk/d14s0
   root@qfs2rac-node1:~# metainit -s dataset1 d21 1 1 /dev/did/dsk/d17s0
   root@qfs2rac-node1:~# metainit -s dataset1 d30 1 1 /dev/did/dsk/d23s0
   root@qfs2rac-node1:~# metainit -s dataset1 d31 1 1 /dev/did/dsk/d16s0
   root@qfs2rac-node1:~# metainit -s dataset1 d40 1 1 /dev/did/dsk/d15s0
   root@qfs2rac-node1:~# metainit -s dataset1 d41 1 1 /dev/did/dsk/d19s0
   ...
   root@qfs2rac-node1:~#
   ```

5. Create the first RAID-1 mirror. Use the command **metainit -s** *diskset-name RAID-1-mirrorname* **-m** *RAID-0-volume0*, where:

   - *diskset-name* is the name of the multi-owner disk set.
   - *RAID-1-mirrorname* is the name of the RAID-1 mirrored volume.
   - *RAID-0-volume0* is the first RAID-0 logical volume that you are adding to the mirror.

   In the example, we create mirror **d1** and add the first RAID-0 volume in the mirror, **d10**:

   ```
   root@qfs2rac-node1:~# metainit -s dataset1 d1 -m d10
   root@qfs2rac-node1:~#
   ```

6. Add the remaining RAID-0 volumes to the first RAID-1 mirror. Use the command **metattach -s** *diskset-name RAID-1-mirrorname RAID-0-volume*, where:

   - *diskset-name* is the name of the multi-owner disk set
   - *RAID-1-mirrorname* is the name of the RAID-1 mirrored volume
   - *RAID-0-volume* is the RAID-0 logical volume that you are adding to the mirror.

   In the example, **d1** is a two-way mirror, so we add a single RAID-0 volume, **d11**:

   ```
   root@qfs2rac-node1:~# metattach -s dataset1 d11 d1
   root@qfs2rac-node1:~#
   ```

7. Create the remaining mirrors.

   In the example, we create mirrors, **d2**, **d3**, **d4**, etc.

   ```
   root@qfs2rac-node1:~# metainit -s dataset1 d2 -m d20
   root@qfs2rac-node1:~# metattach -s dataset1 d21 d2
   root@qfs2rac-node1:~# metainit -s dataset2 d3 -m d30
   root@qfs2rac-node1:~# metattach -s dataset2 d31 d3
   root@qfs2rac-node1:~# metainit -s dataset2 d4 -m d40
   root@qfs2rac-node1:~# metattach -s dataset2 d41 d4
   ...
   root@qfs2rac-node1:~#
   ```

8. Select the mirrors that will hold the QFS file-system metadata.

   For the examples below, we choose mirrors **d1** and **d2**.

9. In the selected mirrors, create soft partitions to hold the QFS metadata. For each mirror, use the command **metainit -s** *diskset-name partition-name* **-p** *RAID-1-mirrorname size*, where:

   - *diskset-name* is the name of the multi-owner disk set.
   - *partition-name* is the name of the new partition.
   - *RAID-1-mirrorname* is the name of the mirror.
   - *size* is the size of the partition.

   In the example, we create two 500-gigabyte partitions: **d53** on mirror **d1** and **d63** on mirror **d2**:

   ```
   root@qfs2rac-node1:~# metainit -s dataset1 d53 -p d1 500g
   root@qfs2rac-node1:~# metainit -s dataset1 d63 -p d2 500g
   ```

10. Next, Create a QFS shared file system on the SC-RAC cluster using mirrored volumes.

Create a QFS Shared File System on the SC-RAC Cluster Using Mirrored Volumes

1. If you have not already done so, carry out the procedure "Create QFS Hosts Files on All SC-RAC Cluster Nodes" on page 9-43. When finished, return here.

2. Select the cluster node that will serve as both the primary node for the SC-RAC cluster and the active metadata server for the QFS shared file system. Log in as **root**.

   In the example, we select node **qfs2rac-node1**:

   ```
   root@qfs2rac-node1:~#
   ```

3. On the primary node, create a shared, high-performance, **ma** file system. Use Solaris Volume Manager mirrored-disk volumes as **mm** metadata devices and **mr** data devices. In a text editor, open the **/etc/opt/SUNWsamfs/mcf** file, make the required edits, and save the file.

   In the example, we use the **vi** text editor to create the file system **qfs2rac**. Partitions on mirrored volumes **d1** and **d2** serve as the file system's two **mm** metadata devices, **110** and **120**. Mirrored volumes **d3** and **d4** serve as the file system's two **mr** data devices, **130** and **140**.

   ```
   root@qfs2rac-node1:~# vi /etc/opt/SUNWsamfs/mcf
   # /etc/opt/SUNWsamfs/mcf file:
   #
   ```

```
# Equipment              Equipment Equipment Family   Device  Additional
# Identifier             Ordinal   Type      Set      State   Parameters
# ---------------------- --------- --------  -------  ------  ----------
qfs2rac                  100       ma        qfs2rac  on      shared
/dev/md/dataset1/dsk/d53  110       mm        qfs2rac  on
/dev/md/dataset1/dsk/d63  120       mm        qfs2rac  on
/dev/md/dataset1/dsk/d3   130       mr        qfs2rac  on
/dev/md/dataset1/dsk/d4   140       mr        qfs2rac  on
:wq
root@qfs2rac-node1:~#
```

4. Check the **mcf** file for errors. Use the command **/opt/SUNWsamfs/sbin/sam-fsd**, and correct any errors found.

   The **sam-fsd** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error. In the example, we check the **mcf** file on host **qfs2rac-node1**:

```
root@qfs2rac-node1:~# sam-fsd
...
Would start sam-archiverd()
Would start sam-stagealld()
Would start sam-stagerd()
Would start sam-amld()
root@qfs2rac-node1:~#
```

5. Create the file system. Use the command **/opt/SUNWsamfs/sbin/sammkfs -S** *family-set-name*, where *family-set-name* is the equipment identifier for the file-system.

   The **sammkfs** command reads the **hosts.***family-set-name* and **mcf** files and creates a shared file system with the specified properties.

```
root@qfs2rac-node1:~# sammkfs -S qfs2rac
Building 'qfs2rac' will destroy the contents of devices:
  ...
Do you wish to continue? [y/N]yes ...
root@qfs2rac-node1:~#
```

6. Open the operating system's **/etc/vfstab** file in a text editor, and start a line for the new file system. Enter the file system name in the first column, spaces, a hyphen in the second column, and more spaces.

   In the example, use the **vi** text editor. We start a line for the **qfs2rac** file system. The hyphen keeps the operating system from attempting to check file system integrity using UFS tools:

```
root@qfs2rac-node1:~# 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      -
...
qfs2rac   -
```

7. In the third column of the **/etc/vfstab** file, enter the mount point of the file system relative to the cluster. Specify a mount-point subdirectory that is not directly beneath the system root directory.

Mounting a shared QFS file system immediately under root can cause failover issues when using the **SUNW.qfs** resource type. In the example, the mount point for the **qfs2rac** file system is **/global/sc-rac/qfs2rac**:

```
#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      -
...
qfs2rac   -       /global/sc-rac/qfs2rac samfs -    no
```

8. In the fourth column of the **/etc/vfstab** file, enter the file system type (**samfs**).

```
#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      -
...
qfs2rac   -       /global/sc-rac/qfs2rac samfs
```

9. In the fifth column of the **/etc/vfstab** file, enter the **fsck** pass option (**-**).

```
#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      -
...
qfs2rac   -       /global/sc-rac/qfs2rac samfs -
```

10. In the sixth column of the **/etc/vfstab** file, enter the mount-at-boot option (**no**).

```
#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      -
...
qfs2rac   -       /global/sc-rac/qfs2rac samfs -    no
```

11. In the seventh column of the **/etc/vfstab** file, enter the **sw_raid** mount option and the recommended mount options for the SC-RAC configuration. Then save the file, and close the editor.

    The following mount options are recommended. They can be specified here, in **/etc/vfstab**, or in the file **/etc/opt/SUNWsamfs/samfs.cmd**, if more convenient:

    - **shared**
    - **stripe=1**
    - **sync_meta=1**
    - **mh_write**
    - **qwrite**
    - **forcedirectio**

- **notrace**
- **rdlease=300**
- **wrlease=300**
- **aplease=300**

In the example, the mount options list has been abbreviated to fit the page layout:

```
#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      -
...
qfs2rac   -       /global/sc-rac/qfs2rac samfs  -    no      shared,...sw_raid
:wq
root@qfs2rac-node1:~#
```

12. Create the mount point for the high-availability shared file system.

```
root@qfs2rac-node1:~# mkdir -p /global/sc-rac/qfs2rac
root@qfs2rac-node1:~#
```

13. Mount the high-availability shared file system on the primary node.

```
root@qfs2rac-node1:~# mount /global/sc-rac/qfs2rac
root@qfs2rac-node1:~#
```

14. Next, configure a potential QFS metadata server on the remaining SC-RAC cluster nodes.

Configure a Potential QFS Metadata Server on the Remaining SC-RAC Cluster Nodes

The remaining nodes of the cluster serve as potential metadata servers. A potential metadata server is a host that can access the metadata devices and assume the duties of a metadata server. So, if the active metadata server on the primary node fails, the Solaris Cluster software can failover to a secondary node and activate a potential metadata server.

For each remaining node in the SC-RAC cluster, proceed as follows:

1. Log in to the node as **root**.

   In the example, the current node is **qfs1rac-node2**:

```
root@qfs1rac-node2:~#
```

2. Copy the **/etc/opt/SUNWsamfs/mcf** file from the primary node to the current node.
3. Check the **mcf** file for errors. Run the command **/opt/SUNWsamfs/sbin/sam-fsd**, and correct any errors found.

   The **sam-fsd** command reads Oracle HSM configuration files and initializes file systems. It will stop if it encounters an error. In the example, we check the **mcf** file on host **qfs1rac-node2**:

```
root@qfs1rac-node2:~# sam-fsd
...
Would start sam-archiverd()
Would start sam-stagealld()
Would start sam-stagerd()
Would start sam-amld()
root@qfs1rac-node2:~#
```

4. Open the operating system's **/etc/vfstab** file in a text editor, and start a line for the new file system.

   In the example, we use the **vi** editor:

```
root@qfs1rac-node2:~# 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       -
...
qfs1rac    -        /global/sc-rac/qfs1rac samfs   -     no
```

5. In the seventh column of the **/etc/vfstab** file, enter the mount options listed below. Then save the file, and close the editor.

   The following mount options are recommended for the SC-RAC cluster configuration. They can be specified here, in **/etc/vfstab**, or in the file **/etc/opt/SUNWsamfs/samfs.cmd**, if more convenient:

   - **shared**
   - **stripe=1**
   - **sync_meta=1**
   - **mh_write**
   - **qwrite**
   - **forcedirectio**
   - **notrace**
   - **rdlease=300**
   - **wrlease=300**
   - **aplease=300**

   In the example, the list has been abbreviated to fit the page layout:

```
#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      -
...
qfs1rac   -       /global/sc-rac/qfs1rac samfs   -     no      shared,...=300
:wq
root@qfs1rac-node2:~#
```

6. Create the mount point for the high-availability shared file system on the secondary node.

```
root@qfs1rac-node2:~# mkdir -p /global/sc-rac/qfs1rac
root@qfs1rac-node2:~#
```

7. Mount the high-availability shared file system on the secondary node.

```
root@qfs1rac-node2:~# mount /global/sc-rac/qfs1rac
root@qfs1rac-node2:~#
```

8. Now configure failover of the SC-RAC metadata server.

Configure Failover of the SC-RAC Metadata Servers

When you host an Oracle HSM shared file system in a cluster managed by Solaris Cluster software, you configure failover of the metadata servers by creating a **SUNW.qfs** cluster resource, a resource type defined by the Oracle HSM software (see the **SUNW.qfs** man page for details). To create and configure the resource for an SC-RAC configuration, proceed as follows:

1. Log in to the primary node in the SC-RAC cluster as **root**.

   In the example, the primary node is **qfs1rac-node1**:

   ```
   root@qfs1rac-node1:~#
   ```

2. See if the **SUNW.qfs** resource type has already been registered with the cluster. Use the command **clresourcetype show**.

   In the example, the **SUNW.qfs** has already been registered:

   ```
   root@qfs1rac-node1:~# clresourcetype show
   === Registered Resource Types ===
   ...
   Resource Type:                 SUNW.qfs:5
     RT_description:              SAM-QFS Agent on Solaris Cluster
   ...
   root@qfs1rac-node1:~#
   ```

3. If the **SUNW.qfs** resource type has not been registered, register it now. Use the command **clresourcetype registerSUNW.qfs**.

   ```
   root@qfs1rac-node1:~# clresourcetype register SUNW.qfs
   root@qfs1rac-node1:~#
   ```

4. If registration fails because the registration file cannot be found, place a symbolic link to the **/opt/SUNWsamfs/sc/etc/** directory in the directory where Solaris Cluster keeps resource-type registration files, **/opt/cluster/lib/rgm/rtreg/**.

   You did not install Oracle Solaris Cluster software before installing Oracle HSM software. Normally, Oracle HSM automatically provides the location of the **SUNW.qfs** registration file when it detects Solaris Cluster during installation. So you need to create a link manually.

   ```
   root@qfs1rac-node1:~# cd /opt/cluster/lib/rgm/rtreg/
   root@qfs1rac-node1:~# ln -s /opt/SUNWsamfs/sc/etc/SUNW.qfs SUNW.qfs
   root@qfs1rac-node1:~#
   ```

5. Create a resource group for the QFS metadata server. Use the Solaris Cluster command **clresourcegroup create -n** *node-list group-name*, where *node-list* is a comma-delimited list of the cluster nodes and *group-name* is the name that we want to use for the resource group.

   In the example, we create the resource group **qfsracrg** with the SC-RAC server nodes as members:

   ```
   root@qfs1rac-node1:~# clresourcegroup create -n qfs1rac-node1,qfs1rac-node2
   qfsracrg
   root@qfs1rac-node1:~#
   ```

6. In the new resource group, set up a logical host name for the active metadata server. Use the Solaris Cluster command **clreslogicalhostname create -g** *group-name*, where *group-name* is the name of the QFS resource group and *virtualMDS* is the logical host name.

   Use the same logical host name that you used in the hosts files for the shared file system. In the example, we create the virtual host **qfs1rac-mds** in the **qfsracrg** resource group:

   ```
   root@qfs1rac-node1:~# clreslogicalhostname create -g qfsracrg qfs1rac-mds
   root@qfs1rac-node1:~#
   ```

7. Add the QFS file-system resource to the resource group. Use the command **clresource create -g** *group-name* **-t SUNW.qfs -x QFSFileSystem=***mount-point resource-name*, where:

   - *group-name* is the name of the QFS resource group.
   - *mount-point* is the mount point for the file system in the cluster, a subdirectory that is not directly beneath the system root directory.

     Mounting a shared QFS file system immediately under root can cause failover issues when using the **SUNW.qfs** resource type.
   - *resource-name* is the name that you want to give to the QFS file-system resource.

   In the example, we create a resource named **scrac** of type **SUNW.qfs** in the resource group **qfsracrg**. We set the **SUNW.qfs** extension property **QFSFileSystem** to the **/global/sc-rac/qfs1rac** mount point:

   ```
   root@qfs1rac-node1:~# create -g qfsracrg -t SUNW.qfs
   -x QFSFileSystem=/global/sc-rac/qfs1rac scrac
   root@qfs1rac-node1:~#
   ```

8. The cluster should not make the QFS file system available if users and applications cannot reach the active metadata server. So make the **SUNW.qfs** resource depend on the logical host name. Use the Solaris Cluster command **clresource set -p Resource_dependencies=***virtualMDS resource-name*, where:

   - *virtualMDS* is the cluster's logical host name for the active file-system metadata server.

     The logical host name always points to the public network interface on the active metadata server, so that user and client access to the file system does not depend on the availability of a specific physical network interface.
   - *resource-name* is the name of the **SUNW.qfs** resource.

   In the example, the logical host name that we created when we set up the **SUNW.qfs** resource is **qfs1rac-mds**. The resource is named **haqfs**:

   ```
   root@qfs1rac1mds-node1:~# clresource set -p Resource_dependencies=qfs1rac-mds
   haqfs
   root@qfs1racmds-node1:~#
   ```

9. Bring the resource group online. Use the Solaris Cluster commands **clresourcegroup manage** *group-name* and **clresourcegroup online -emM** *group-name*, where *group-name* is the name of the QFS resource group.

   In the example, we bring the **qfsracrg** resource group online:

   ```
   root@qfs1rac-node1:~# clresourcegroup manage qfsracrg
   ```

```
root@qfs1rac-node1:~# clresourcegroup online -emM qfsracrg
root@qfs1rac-node1:~#
```

10. Make sure that the QFS resource group is online. Use the Solaris Cluster command **clresourcegroup status**.

    In the example, the **qfsracrg** resource group is **online** on the primary node, **qfs1rac-node1**:

```
root@qfs1rac-node1:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name  Node Name       Suspended   Status
----------  -------------   ---------   ------
qfsracrg    qfs1rac-node1   No          Online
            qfs1rac-node2   No          Offline
            qfs1rac-node3   No          Offline
root@qfs1rac-node1:~#
```

11. Make sure that the resource group fails over correctly. Move the resource group to the secondary node. Use the Solaris Cluster command **clresourcegroup switch -n** *node2 group-name*, where:

    - *node2* is the name of the secondary node.
    - *group-name* is the name that you have chosen for the HA-HSM resource group

    Then use **clresourcegroup status** to check the result.

    In the example, we move the **qfsracrg** resource group to **qfs1rac-node2** and **qfs1rac-node3**, confirming that the resource group comes online on the specified node:

```
root@qfs1rac-node1:~# clresourcegroup switch -n qfs1rac-node2 qfsracrg
root@qfs1rac-node1:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name  Node Name       Suspended   Status
----------  -------------   ---------   ------
qfsracrg    qfs1rac-node1   No          Offline
            qfs1rac-node2   No          Online
            qfs1rac-node3   No          Offline
root@qfs1rac-node1:~# clresourcegroup switch -n qfs1rac-node3 qfsracrg
root@qfs1rac-node1:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name  Node Name       Suspended   Status
----------  -------------   ---------   ------
qfsracrg    qfs1rac-node1   No          Offline
            qfs1rac-node2   No          Offline
            qfs1rac-node3   No          Online
root@qfs1rac-node1:~#
```

12. Move the resource group back to the primary node. Use the Solaris Cluster command **clresourcegroup switch -n** *node1 group-name*, where *node1* is the name of the primary node and *group-name* is the name that you have chosen for the HA-HSM resource group. Then use **clresourcegroup status** to check the result.

    In the example, we successfully move the **qfsracrg** resource group back to **qfs1rac-node1**:

```
root@qfs1rac-node1:~# clresourcegroup switch -n qfs1rac-node1 qfsracrg
root@qfs1rac-node1:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name  Node Name       Suspended   Status
```

```
----------  -------------   ---------   ------
samr        qfs1rac-node1   No          Online
            qfs1rac-node2   No          Offline
            qfs1rac-node3   No          Offline
root@qfs1rac-node1:~#
```

13. If you need to share the HA-QFS file system with HA-NFS or HA-SAMBA, go to "Sharing HA-HSM or HA-QFS Configurations with HA-NFS or HA-SAMBA" on page 9-63.
14. If you plan on using the sideband database feature, go to "Configuring the Reporting Database" on page 10-1.
15. Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

Sharing HA-HSM or HA-QFS Configurations with HA-NFS or HA-SAMBA

The Oracle HSM HA-HSM and HA-QFS failover configurations support highly available network file sharing using the Solaris Cluster HA-NFS and/or HA-SAMBA data services. (*HA-COTC and SC-RAC configurations do not support HA-NFS or HA-SAMBA.*)

To add HA-NFS and/or the HA-SAMBA to an HA-HSM or HA-QFS configuration, install the data service software, configure the file sharing resources, and add the resources to the HA-HSM or HA-QFS resource group, so that no resource depends on any resource outside its own resource group. For full instructions, see the relevant section(s) below:

- "Configure HA-NFS" on page 9-63
- "Configure HA-SAMBA" on page 9-66.

Configure HA-NFS

The High Availability NFS (HA-NFS) data service makes exported NFS paths available via highly available virtual IP addresses. To configure HA-NFS, carry out the following tasks:

- Prepare local storage for the HA-NFS data service.
- Install and configure the HA-NFS data service.

Configure Highly Available Local Storage for the HA-NFS Data Service

The HA-NFS agent uses highly available local storage for application state and configuration data. Proceed as follows:

1. If you are adding HA-NFS support to an Oracle HSM HA-QFS configuration and have not yet created a highly available local file system, do so now. Use the procedures "Create a High-Availability Local File System for Software Configuration Files" on page 9-32and "Configure Failover for the High-Availability Local File System" on page 9-38.
2. If you are adding HA-NFS support to an Oracle HSM HA-HSM configuration and have already created a highly available local file system for Oracle HSM, do not create another now.

   Use the existing highly available local file system.
3. Make sure that the highly available local file system is mounted. Use the **`mount`** command, and look for the file system name in the output.

In the example, the highly available local file system, **/global/ha_config**, is mounted:

```
root@hsm1mds-node1:~# mount
/ on ...
/global/ha_config/ on ...
root@hsm1mds-node1:~#
```

4. If you are adding HA-NFS support to an HA-HSM configuration, make sure that the Oracle HSM archive's highly available QFS file system is mounted.

   In the example, the highly available QFS file system, **/global/ha_hsmfs1**, is mounted:

```
root@hsm1mds-node1:~# mount
/ on ...
/global/ha_config/ on ...
/global/ha_hsmfs/ on ...
root@hsm1mds-node1:~#
```

5. If a required file system is not mounted, mount it now.

6. Create an **nfs/** subdirectory in the highly available local file system. Use the command **mkdir /global/***mountpoint***/nfs/**, where *mountpoint* is the subdirectory where the highly available local file system attaches to the cluster's **/global/** file system.

   This subdirectory will hold state and configuration data for the HA-NFS agent. In the example, we create the subdirectory **/global/ha_config/nfs/**, because **ha_config/** is the mount point directory of the highly available local file system that we created when we configured Oracle HSM for failover:

```
root@hsm1mds-node1:~# ls /global/
ha_config/
root@hsm1mds-node1:~# ls /global/ha_config/
hsm/
root@hsm1mds-node1:~# mkdir /global/ha_config/nfs/
root@hsm1mds-node1:~# ls /global/ha_config/
hsm/        nfs/
root@hsm1mds-node1:~#
```

7. In the **nfs/** subdirectory, create an **admin/** subdirectory. Use the command **mkdir /global/***mountpoint***/nfs/admin/**, where *mountpoint* is the subdirectory where the highly available local file system attaches to the cluster's **/global/** file system.

   In the example, we create **/global/ha_config/nfs/admin/**:

```
root@hsm1mds-node1:~# mkdir /global/ha_config/nfs/admin/
root@hsm1mds-node1:~# ls /global/ha_config/nfs/
admin/
root@hsm1mds-node1:~#
```

8. In the **nfs/admin/** subdirectory, create a **SUNW.nfs/** subdirectory. Use the command **mkdir /global/***mountpoint***/nfs/admin/SUNW.nfs/**, where *mountpoint* is the subdirectory where the highly available local file system attaches to the cluster's **/global/** file system.

   In the example, we create **/global/ha_config/nfs/admin/SUNW.nfs/**:

```
root@hsm1mds-node1:~# mkdir /global/ha_config/nfs/admin/SUNW.nfs/
root@hsm1mds-node1:~# ls /global/ha_config/nfs/admin/
SUNW.nfs/
root@hsm1mds-node1:~#
```

9. In the **nfs/admin/SUNW.nfs/** subdirectory, create a **statmon/** subdirectory. Use the command **mkdir /global/***mountpoint***/nfs/admin/SUNW.nfs/statmon/**, where *mountpoint* is the subdirectory where the highly available local file system attaches to the cluster's **/global/** file system.

   In the example, we create **/global/ha_config/nfs/admin/SUNW.nfs/statmon/**:

```
root@hsm1mds-node1:~# mkdir /global/ha_config/nfs/admin/SUNW.nfs/statmon/
root@hsm1mds-node1:~# ls /global/ha_config/nfs/admin/SUNW.nfs/
statmon/
root@hsm1mds-node1:~#
```

10. In the **nfs/admin/SUNW.nfs/** subdirectory, create a **v4_oldstate/** subdirectory. Use the command **mkdir /global/***mountpoint***/nfs/admin/SUNW.nfs/v4_oldstate/**, where *mountpoint* is the subdirectory where the highly available local file system attaches to the cluster's **/global/** file system.

    We create **/global/ha_config/nfs/admin/SUNW.nfs/v4_oldstate/** in the example:

```
root@hsm1mds-node1:~# mkdir /global/ha_config/nfs/admin/SUNW.nfs/v4_oldstate/
root@hsm1mds-node1:~# ls /global/ha_config/nfs/admin/SUNW.nfs/
statmon/    v4_oldstate/
root@hsm1mds-node1:~#
```

11. In the **nfs/admin/SUNW.nfs/** subdirectory, create a **v4_state/** subdirectory. Use the command **mkdir /global/***mountpoint***/nfs/admin/SUNW.nfs/v4_state/**, where *mountpoint* is the subdirectory where the highly available local file system attaches to the cluster's **/global/** file system.

    In the example, we create **/global/ha_config/nfs/admin/SUNW.nfs/v4_state/**:

```
root@hsm1mds-node1:~# mkdir /global/ha_config/nfs/admin/SUNW.nfs/v4_state/
root@hsm1mds-node1:~# ls /global/ha_config/nfs/admin/SUNW.nfs/
statmon/    v4_oldstate/    v4_state/
root@hsm1mds-node1:~#
```

12. Set permissions on the **nfs/** subdirectory recursively, so that the user/owner has read, write, and traverse permissions to all subdirectories, while others have only read and traverse. Use the command **chmod -R 755 /global/***mountpoint***/nfs/**, where *mountpoint* is the subdirectory where the highly available local file system attaches to the cluster's **/global/** file system.

    In the example, *mountpoint* is **ha_config**:

```
root@hsm1mds-node1:~# chmod -R 755 /global/ha_config/nfs/
root@hsm1mds-node1:~#
```

13. Next, install and configure the HA-NFS data service.

Install and Configure the HA-NFS Data Service

1. Install and configure HA-NFS using the procedures in the *Oracle Solaris Cluster Data Service for Network File System (NFS) Guide*.

   This document is available in *Oracle Solaris Cluster* online documentation library.

2. Set the **Pathprefix=** property of the HA-NFS resource to the path for the HA-NFS **admin/** directory. Use the command **clresource set-p Pathprefix=/global/***mountpoint***/nfs/admin/** *groupname*, where:

- *mountpoint* is the directory where the highly available local file system attaches to the cluster's **/global/** file system.
- *groupname* is the name that you have chosen for the HA-HSM or HA-QFS resource group.

```
root@hsm1mds-node1:~# clresourcegroup set -p Pathprefix=/global/ha_
config/nfs/admin/ hsmrg
```

3. The HA-NFS agent should not start unless the high-availability local file system is available. So make the **SUNW.nfs** resource depend upon the **SUNW.HAStoragePlus** resource. Use the Solaris Cluster command **clresource set -p Resource_dependencies=***dependency resource-name*, where:
   - *dependency* is name of the **SUNW.HAStoragePlus** resource.
   - *resource-name* is the name of the **SUNW.nfs** resource.

   In the example, we make the **SUNW.nfs** resource depend upon the **SUNW.HAStoragePlus** resource, **halocal**:

```
root@hsm1mds-node1:~# clresource set -p Resource_dependencies=halocal hanfs
root@hsm1mds-node1:~#
```

4. When finished, configure HA-SAMBA, if required.
5. Otherwise, if you plan on using the sideband database feature, go to "Configuring the Reporting Database" on page 10-1.
6. If not, go to "Configuring Notifications and Logging" on page 11-1.

Configure HA-SAMBA

SAMBA is an implementation of the Server Message Block/Common Internet File System networking protocol (SMB/CIFS) that lets hosts running Solaris and Linux operating systems to share files with hosts running Microsoft Windows. To configure SAMBA as a high availability service, carry out the following tasks:

- Configure highly available local storage for HA-SAMBA configuration files.
- Install the HA-SAMBA data service software.
- Configure SAMBA services.

Configure Highly Available Local Storage for HA-SAMBA

1. If you are adding HA-SAMBA support to an Oracle HSM HA-QFS configuration and have not yet created a highly available local file system for application state and configuration data, do so now. Use the procedures "Create a High-Availability Local File System for Software Configuration Files" on page 9-32 and "Configure Failover for the High-Availability Local File System" on page 9-38.
2. If you are adding HA-SAMBA support to an Oracle HSM HA-HSM configuration or if you have already created a highly available local file system for HA-NFS, do not create another now.

   Use the existing highly available local file system.
3. Make sure that the highly available local file system is mounted. Use the **mount** command, and look for the file system name in the output.

   In the example, the highly available local file system, **/global/ha_config**, is mounted:

```
root@hsm1mds-node1:~# mount
/ on ...
/global/ha_config/ on ...
root@hsm1mds-node1:~#
```

4. If you are adding HA-SAMBA support to an HA-HSM configuration, make sure that the Oracle HSM archive's highly available QFS file system is mounted.

   In the example, the highly available QFS file system, **/global/ha_hsmfs1**, is mounted:

```
root@hsm1mds-node1:~# mount
/ on ...
/global/ha_config/ on ...
/global/ha_hsmfs/ on ...
root@hsm1mds-node1:~#
```

5. If a required file system is not mounted, mount it now.
6. Create a **samba/** directory in the highly available local file system. Use the command **mkdir /global/***ha_localfs***/samba/**, where *ha_localfs* is the name of highly available local file system.

   This directory will hold state and configuration data for the HA-SAMBA agent. In the example, we created the highly available local file system **/global/ha_config** when we configured Oracle HSM for failover. So we create the subdirectory **/global/ha_config/samba/**:

```
root@hsm1mds-node1:~# mkdir /global/ha_config/samba/
root@hsm1mds-node1:~# ls /global/ha_config/
samba/
root@hsm1mds-node1:~#
```

7. In the **samba/** directory, create a directory named for the logical host name of the cluster. Use the command **mkdir /global/***ha_localfs***/samba/***logical-hostname*, where:

   - *ha_localfs* is the name of highly available local file system.
   - *logical-hostname* is the name of the highly available, virtual host that lets users and applications connect with the cluster.

   In the example, we created the highly available local file system **/global/ha_config** when we configured Oracle HSM for failover. We named the virtual host **hsm1mds** when we configured Solaris Cluster. So we create the subdirectory **/global/ha_config/samba/hsm1mds**:

```
root@hsm1mds-node1:~# mkdir /global/ha_config/samba/hsm1mds
root@hsm1mds-node1:~# ls /global/ha_config/samba/
hsm1mds/
root@hsm1mds-node1:~#
```

8. Next, install the HA-SAMBA data service.

Install the HA-SAMBA Data Service and Carry Out Initial Configuration

1. Install the HA-SAMBA data service software and carry out initial configuration using the procedures in the *Oracle Solaris Cluster Data Service for Samba Guide.*

   This document is available in *Oracle Solaris Cluster* online documentation library.

2. Then, when you are ready to edit the **/opt/SUNWscsmb/util/samba_config** file, configure the **smbd** and **nmbd** services as described below.

Configure SAMBA Services

To configure SAMBA services, you enter values for the required configuration parameters in a file, **/opt/SUNWscsmb/util/samba_config**, and then run **/opt/SUNWscsmb/util/samba_register**, a script that takes these values as input. To configure the services, proceed as follows:

- Edit **samba_config** and execute **samba_register** once to configure and register the **smbd** and**nmbd** services.
- If you are using Microsoft Active Directory, re-edit **samba_config** and re-execute **samba_register** to configure and register the **winbindd** service.
- Define HA-SAMBA resource dependencies.
- Finally, enable the HA-SAMBA resources.

Configure the smbd and nmbd Services

The **smbd** and **nmbd** daemons provide core SAMBA file-sharing and NetBIOS services. To configure them, proceed as follows:

1. Change to the **/opt/SUNWscsmb/util/** directory.

```
root@hsm1mds-node1:~# cd /opt/SUNWscsmb/util/
root@hsm1mds-node1:~#
```

2. Back up the **/opt/SUNWscsmb/util/samba_config** file.

```
root@hsm1mds-node1:~# cp samba_config samba_config.original
root@hsm1mds-node1:~#
```

3. Open the **/opt/SUNWscsmb/util/samba_config** file in a text editor and scroll down past the comments to the **Resource Specific Parameters** section.

```
root@hsm1mds-node1:~# vi /opt/SUNWscsmb/util/samba_config
#!/usr/bin/ksh
# Copyright (c) 2006, 2013, Oracle and/or its affiliates. All rights reserved.
#ident  "@(#)samba_config.ksh   1.9     13/04/25"
...
#+++ Resource Specific Parameters +++
RS=
```

4. In the **Resource Specific Parameters** section of the file, set the value of the **RS** (*resource*) parameter to the name that you want to assign to the highly available SAMBA resource.

   In the example, we name the resource **hasmb**:

```
...
#+++ Resource Specific Parameters +++
RS=hasmb
```

5. Set the value of the **RG** (*resource group*) parameter to the name of the resource group that you created for HA-HSM or HA-QFS.

   In the example, the resource group is named **hsmrg**:

```
...
#+++ Resource Specific Parameters +++
RS=hasmb
RG=hsmrg
```

6. Set the value of the **RS_LH** (*resource logical host*) parameter to the logical host name that you created for HA-HSM or HA-QFS.

   In the example, the logical host name is **hsm1mds**:

   ```
   ...
   #+++ Resource Specific Parameters +++
   RS=hasmb
   RG=hsmrg
   RS_LH=hsm1mds
   ```

7. Set the value of the **RS_HAS** (*resource HAStoragePlus*) parameter to the name that you created for the **HAStoragePlus** resource that provides failover for the highly available local file system.

   In the example, the **HAStoragePlus** resource is named **halocal**, which is the resource we configured when configuring HA-HSM:

   ```
   ...
   #+++ Resource Specific Parameters +++
   RS=hasmb
   RG=hsmrg
   RS_LH=hsm1mds
   RS_HAS=halocal
   ```

8. Set the value of the **SERVICES** parameter to **"smbd,nmbd"** (SMB and NetBIOS name services).

   ```
   ...
   #+++ Resource Specific Parameters +++
   RS=hasmb
   RG=hsmrg
   RS_LH=hsm1mds
   RS_HAS=halocal
   SERVICES="smbd,nmbd"
   ```

9. In the **Common Parameters** section of the file, set the value of the **BINDIR** parameter to the path to the UNIX user binaries directory.

   In the example, this path is **/usr/bin**:

   ```
   ...
   #+++ Common Parameters +++
   BINDIR=/usr/bin
   ```

10. Set the value of the **SBINDIR** parameter to the path to the UNIX system binaries directory.

    In the example, this path is **/usr/sbin**.

    ```
    ...
    #+++ Common Parameters +++
    BINDIR=/usr/bin
    SBINDIR=/usr/sbin
    ```

11. Set the value of the **CFGDIR** parameter to the path to the SAMBA configuration directory for the logical host name.

    You created this directory when you set up highly available local storage for HA-SAMBA. In the example, this path is **/global/ha_config/samba/hsm1mds**, where **hsm1mds** is the logical host name:

    ```
    ...
    ```

```
#+++ Common Parameters +++
BINDIR=/usr/bin
SBINDIR=/usr/sbin
CFGDIR=/global/ha_config/samba/hsm1mds
```

12. Set the value of the **LDPATH** parameter to the path to the shared libraries required by user and system binaries.

    In the example, this path is **/usr/lib**:

```
...
#+++ Common Parameters +++
BINDIR=/usr/bin
SBINDIR=/usr/sbin
CFGDIR=/global/ha_config/samba/hsm1mds
LDPATH=/usr/lib
```

13. Set the value of the **FMUSER** parameter to the name of the HA-SAMBA Fault Monitor user.

    You created Fault Monitor user during initial configuration of HA-SAMBA as documented in the *Oracle Solaris Cluster Data Service for Samba Guide*. In the example, the Fault Monitor user is named **smbmonitor**:

```
...
#+++ Common Parameters +++
BINDIR=/usr/bin
SBINDIR=/usr/sbin
CFGDIR=/global/ha_config/samba/hsm1mds
LDPATH=/usr/lib
FMUSER=smbmonitor
```

14. In the **SMBD & NMBD Specific Parameters** section of the **samba_config** file, set the value of the **SAMBA_LOGDIR** parameter to the path to the directory where the logs for this logical host should be kept.

    You created this directory when you set up highly available local storage. In the example, the path is **/global/ha_config/samba/hsm1mds/logs**:

```
...
#+++ SMBD & NMBD Specific Parameters +++
SAMBA_LOGDIR=/global/ha_config/samba/hsm1mds/logs
```

15. Set the value of the **SAMBA_FMPASS** parameter to the Fault Monitor user's password.

    You set the Fault Monitor user's password during initial configuration of HA-SAMBA as documented in the *Oracle Solaris Cluster Data Service for Samba Guide* and in the comments in the **samba_config** file. For the purposes of this example, the Fault Monitor user's password is the unencrypted string **SMBfm0n_PW**:

```
...
#+++ SMBD & NMBD Specific Parameters +++
SAMBA_LOGDIR=/global/ha_config/samba/hsm1mds/logs
SAMBA_FMPASS=SMBfm0n_PW
```

16. Leave the value of the **SAMBA_FMDOMAIN** parameter blank.

```
...
#+++ SMBD & NMBD Specific Parameters +++
SAMBA_LOGDIR=/global/ha_config/samba/hsm1mds/logs
SAMBA_FMPASS=SMBfm0n_PW
SAMBA_FMDOMAIN=
```

17. Save the **samba_config** file, and close the editor.

```
...
#+++ SMBD & NMBD Specific Parameters +++
SAMBA_LOGDIR=/global/ha_config/samba/hsm1mds/logs
SAMBA_FMPASS=SMBfm0n_PW
SAMBA_FMDOMAIN=
:wq
root@hsm1mds-node1:~#
```

18. Register the **smbd**/**nmbd** resource. Execute the configuration script **/opt/SUNWscsmb/util/samba_register**.

```
root@hsm1mds-node1:~# ./samba_register
root@hsm1mds-node1:~#
```

19. If you need to use the SAMBA host In a Microsoft Windows Active Directory domain, configure the **winbindd** service now.
20. Otherwise, define resource dependencies for the SAMBA service.

Configure the winbindd Service

The **winbindd** daemon provides the name-resolution and authentication mechanisms that are necessary when using non-Windows hosts with Microsoft Windows Active Directory. To configure **winbindd**, proceed as follows:

1. If you are not currently in the **/opt/SUNWscsmb/util/** directory, change to it now.

```
root@hsm1mds-node1:~# cd /opt/SUNWscsmb/util/
root@hsm1mds-node1:~#
```

2. Save the **smbd**/**nmbd** configuration by copying the edited **/opt/SUNWscsmb/util/samba_config** file and giving it a new name.

```
root@hsm1mds-node1:~# cp samba_config samba_config.smbdnmbd-hsm
root@hsm1mds-node1:~#
```

3. Open the **/opt/SUNWscsmb/util/samba_config** file in a text editor and scroll down past the comments to the **Resource Specific Parameters** section.

```
root@hsm1mds-node1:~# vi /opt/SUNWscsmb/util/samba_config
#!/usr/bin/ksh
# Copyright (c) 2006, 2013, Oracle and/or its affiliates. All rights reserved.
#ident  "@(#)samba_config.ksh   1.9     13/04/25"
...
#+++ Resource Specific Parameters +++
RS=
```

4. In the **Resource Specific Parameters** section of the file, set the value of the **RS** (*resource*) parameter to the name that you want to assign to the **windbind** resource.

   In the example, we name the resource **hawinb**:

```
...
#+++ Resource Specific Parameters +++
RS=hawinb
```

5. Leave the value of the **RG** (*resource group*) parameter set to the name of the resource group that you created for HA-HSM or HA-QFS.

   In the example, the resource group is named **hsmrg**:

```
...
```

```
#+++ Resource Specific Parameters +++
RS=hawinb
RG=hsmrg
```

6. Leave the value of the **RS_LH** (*resource logical host*) parameter set to the logical host name that you created for HA-HSM or HA-QFS.

   In the example, the logical host name is **hsm1mds**:

```
...
#+++ Resource Specific Parameters +++
RS=hawinb
RG=hsmrg
RS_LH=hsm1mds
```

7. Leave the value of the **RS_HAS** (*resource HAStoragePlus*) parameter set to the name that you created for the **HAStoragePlus** resource that provides failover for the highly available local file system.

   In the example, the **HAStoragePlus** resource is named **halocal**, which is the resource we configured when configuring HA-HSM:

```
#ident  "@(#)samba_config.ksh   1.9     13/04/25"
...
#+++ Resource Specific Parameters +++
RS=hawinb
RG=hsmrg
RS_LH=hsm1mds
RS_HAS=halocal
```

8. Set the value of the **SERVICES** parameter to **"winbindd"**.

```
...
#+++ Resource Specific Parameters +++
RS=hawinb
RG=hsmrg
RS_LH=hsm1mds
RS_HAS=halocal
SERVICES="winbindd"
```

9. In the **Common Parameters** section of the file, leave the value of the **BINDIR** parameter set to the path to the UNIX user binaries directory.

   In the example, this path is **/usr/bin**:

```
...
#+++ Common Parameters +++
BINDIR=/usr/bin
```

10. Leave the value of the **SBINDIR** parameter set to the path to the UNIX system binaries directory.

    In the example, this path is **/usr/sbin**.

```
...
#+++ Common Parameters +++
BINDIR=/usr/bin
SBINDIR=/usr/sbin
```

11. Leave the value of the **CFGDIR** parameter set to the path to the SAMBA configuration directory for the logical host name.

You created this directory when you set up highly available local storage for HA-SAMBA. In the example, this path is **/global/ha_config/samba/hsm1mds**, where **hsm1mds** is the logical host name:

```
...
#+++ Common Parameters +++
BINDIR=/usr/bin
SBINDIR=/usr/sbin
CFGDIR=/global/ha_config/samba/hsm1mds
```

12. Leave the value of the **LDPATH** parameter set to the path to the shared libraries required by user and system binaries.

    In the example, this path is **/usr/lib**:

```
...
#+++ Common Parameters +++
BINDIR=/usr/bin
SBINDIR=/usr/sbin
CFGDIR=/global/ha_config/samba/hsm1mds
LDPATH=/usr/lib
```

13. Leave the value of the **FMUSER** parameter set to the name of the HA-SAMBA Fault Monitor user.

    You created Fault Monitor user during initial configuration of HA-SAMBA as documented in the *Oracle Solaris Cluster Data Service for Samba Guide*. In the example, the Fault Monitor user is named **smbmonitor**:

```
...
#+++ Common Parameters +++
BINDIR=/usr/bin
SBINDIR=/usr/sbin
CFGDIR=/global/ha_config/samba/hsm1mds
LDPATH=/usr/lib
FMUSER=smbmonitor
```

14. In the **SMBD & NMBD Specific Parameters** section of the **samba_config** file, leave the value of the **SAMBA_LOGDIR** parameter set to the path to the directory where the logs for the virtual host should be kept.

    You created this directory when you set up highly available local storage. In the example, the path is **/global/ha_config/samba/hsm1mds/logs**:

```
...
#+++ SMBD & NMBD Specific Parameters +++
SAMBA_LOGDIR=/global/ha_config/samba/hsm1mds/logs
```

15. Leave the value of the **SAMBA_FMPASS** parameter set to the Fault Monitor user's password.

    You set the Fault Monitor user's password during initial configuration of HA-SAMBA as documented in the *Oracle Solaris Cluster Data Service for Samba Guide*. In the example, the Fault Monitor user's password is **SMBfm0n_PW**:

```
...
#+++ SMBD & NMBD Specific Parameters +++
SAMBA_LOGDIR=/global/ha_config/samba/hsm1mds/logs
SAMBA_FMPASS=SMBfm0n_PW
```

16. Set the value of the **SAMBA_FMDOMAIN** parameter to the name of the Samba NT domain where the Fault Monitor user was created.

In the example, we set the domain to **domain.example.com**:

```
...
#+++ SMBD & NMBD Specific Parameters +++
SAMBA_LOGDIR=/global/ha_config/samba/hsm1mds/logs
SAMBA_FMPASS=SMBfm0n_PW
SAMBA_FMDOMAIN=domain.example.com
```

17. Save the **samba_config** file, and close the editor.

```
...
#+++ SMBD & NMBD Specific Parameters +++
SAMBA_LOGDIR=/global/ha_config/samba/hsm1mds/logs
SAMBA_FMPASS=SMBfm0n_PW
SAMBA_FMDOMAIN=domain.example.com
:wq
root@hsm1mds-node1:~#
```

18. Save the **winbindd** configuration by copying the edited **/opt/SUNWscsmb/util/samba_config** file and giving it a new name.

```
root@hsm1mds-node1:~# cp samba_config samba_config.winbindd-hsm
root@hsm1mds-node1:~#
```

19. Register the **winbindd** resource. Execute the configuration script **/opt/SUNWscsmb/util/samba_register**.

```
root@hsm1mds-node1:~# ./samba_register
root@hsm1mds-node1:~#
```

20. Now, define resource dependencies for the SAMBA services.

Define HA-SAMBA Resource Dependencies

1. The highly available **smbd**/**nmbd** resource should not start unless the highly available local file system is available. So make the **smbd**/**nmbd** resource depend upon the **SUNW.HAStoragePlus** resource. Use the Solaris Cluster command **clresource set -p Resource_dependencies=***dependency resource-name*, where:

   - *dependency* is name of the **SUNW.HAStoragePlus** resource.
   - *resource-name* is the name of the **smbd**/**nmbd** resource.

   In the example, the **smbd**/**nmbd** resource is named **hasmb** (the value of the **RS** parameter for the **"smbd,nmbd"** services in the **samba_config** file). So we make the **hasmb** resource depend upon the **SUNW.HAStoragePlus** resource **halocal**:

```
root@hsm1mds-node1:~# clresource set -p Resource_dependencies=halocal hasmb
root@hsm1mds-node1:~#
```

2. The cluster should not make **smbd**/**nmbd** resources available if users and applications cannot reach the active metadata server. So make the **smbd**/**nmbd** resource depend upon the logical host name. Use the Solaris Cluster command **clresource set -p Resource_dependencies=***dependency resource-name*, where:

   - *dependency* is the logical host name.
   - *resource-name* is the name of the **smbd**/**nmbd** resource.

   In the example, we make the **hasmb** resource depend upon the **hsm1mds**, the logical host name that we created when we configured HA-HSM:

```
root@hsm1mds-node1:~# clresource set -p Resource_dependencies=hsm1mds hasmb
root@hsm1mds-node1:~#
```

3. If you need to use Microsoft Active Directory and have configured **winbindd**, make sure that the highly available **smbd**/**nmbd** resource does not start unless the **winbindd** resource is available. Use the command **clresource set -p Resource_ dependencies=***winbind-resource***{local_node}** *samba-resource*, where:

   - *winbind-resource* is the name of the **winbindd** resource (the value of the **RS** parameter for the **winbindd** service in the **samba_config** file).
   - *samba-resource* is the name of the **smbd**/**nmbd** resource (the value of the **RS** parameter for the **smbd** and **nmbd** services in the **samba_config** file).

   In the example, **hawinb** is the name of the **windbind** resource, and **hasmb** is the name of the **smbd**/**nmbd** resource:

   ```
   root@hsm1mds-node1:~# clresource set -p Resource_dependencies=hawinb{local_
   node} hasmb
   root@hsm1mds-node1:~#
   ```

4. Next, enable the HA-SAMBA resources.

Enable the HA-SAMBA Resources

1. First, enable the **windbind** resource. Use the command **clresource enable** *winbind-resource*, where *winbind-resource* is the name of the resource (the value of the **RS** parameter for the **winbindd** service in the **samba_config** file).

   In the example, **hawinb** is the name of the **windbind** resource:

   ```
   root@hsm1mds-node1:~# clresource enable hawinb
   ```

2. Then, enable the **smbd**/**nmbd** resource. Use the command **clresource enable** *smbd/nmbd-resource*, where *smbd/nmbd-resource* is the name of the resource (the value of the **RS** parameter for the **smbd** and **nmbd** services in the **samba_config** file).

   In the example, **hasmb** is the name of the **smbd**/**nmbd** resource:

   ```
   root@hsm1mds-node1:~# clresource enable hasmb
   ```

3. Test the HA-SAMBA configuration using the procedures described in the *Oracle Solaris Cluster Data Service for Samba Guide*.
4. When finished, if you plan on using the sideband database feature, go to "Configuring the Reporting Database" on page 10-1.
5. Otherwise, go to "Configuring Notifications and Logging" on page 11-1.

10

Configuring the Reporting Database

Oracle HSM supports an optional reporting database that stores current metadata information for every file in a specified file system. This *sideband database* can be invaluable for managing and reporting on files and file system activity.

Implementing the Oracle HSM sideband database is straightforward. You use the **samdb** command to create and configure a MySQL database using the supplied database schema (or a custom alternative) and a recovery point file generated by the **samfsdump** command. The Oracle HSM daemon processes then update the database automatically as the corresponding file system changes. Additional **samdb** commands let you query and manage the database. For full details on commands and options, see the **samdb** and **samdb.conf** man pages.

To use the sideband database feature, carry out the following tasks:

- Install and configure the MySQL server software.
- Create a database load file.
- Create the sideband database.

Install and Configure the MySQL Server Software

To enable **samdb** reporting features, you must install and configure a MySQL database. Proceed as follows.

1. Download the *MySQL Reference Manual* from **http://dev.mysql.com/doc/**.

   Use the procedure below to identify the MySQL tasks that are required when enabling **samdb** reporting. But note that the steps below are not meant to be complete or authoritative. Use them as a guide when consulting the *MySQL Reference Manual*.

2. Log in to the system that will host the MySQL server as **root**.

   You can install the MySQL server on the Oracle HSM metadata server host or on an independent Solaris or Linux host.

   In the example, we install MySQL on the Solaris host **samsql**:

   ```
   root@samsql:~#
   ```

3. Download and install the MySQL server software, as directed in the *MySQL Reference Manual*. Enable automatic startup.

4. Connect to the MySQL server with the **mysql** client and the **root** user account. Use the command **mysql --user=root -p**. When prompted, enter the password that you assigned to the **root** user during installation.

The **mysql** command shell starts:

```
root@samsql:~# mysql --user=root -p
Enter Password:
mysql>
```

5. Create the Oracle HSM MySQL user. Use the command **CREATE USER '***user_name***'@'***host_name***' IDENTIFIED BY '***user-password***'**, where:

   - *user_name* is the name of the Oracle HSM MySQL user
   - *host_name* is **localhost** when MySQL is installed on the Oracle HSM metadata server host. otherwise, it is the hostname or IP address of the metadata server.
   - *user-password* is the password that you assign to the Oracle HSM MySQL user.

   In the example, we create the user **samsql** on the Oracle HSM metadata server **samqfs1mds**. We set the user password **samsqluserpassw0rd** for demonstration purposes (it would not be a secure choice for production database use):

```
root@samsql:~# mysql --user=root
Enter Password:
mysql> CREATE USER 'samsql'@'samqfs1mds' IDENTIFIED BY 'samsqluserpassw0rd'
mysql>
```

6. Grant the Oracle HSM user the necessary privileges. Use the command **GRANT CREATE,DROP,INDEX,SELECT,INSERT,UPDATE,DELETE ON** *host_name* **TO '***user_name***'@'***host_name***'**.

   In the example, we grant privileges to the user **samsql** on metadata server **samqfs1mds**:

```
root@samsql:~# mysql --user=root -p
Enter Password:
mysql> CREATE USER 'samsql'@'host_name' IDENTIFIED BY 'samsqluserpassw0rd'
mysql> CREATE,DROP,INDEX,SELECT,INSERT,UPDATE,DELETE ON samqfs1mds TO
'samsql'@'samqfs1mds'
mysql>
```

7. Close the MySQL command interface, and return to the operating system command shell. Use the MySQL command **QUIT**.

```
root@samsql:~# mysql --user=root -p
Enter Password:
mysql> CREATE USER 'samsql'@'host_name' IDENTIFIED BY 'samsqluserpassw0rd'
mysql> GRANT CREATE,DROP,INDEX,SELECT,INSERT,UPDATE,DELETE ON samqfs1mds TO
'samsql'@'samqfs1mds'
mysql> QUIT
Bye
root@solaris:~#
```

8. Next, create a database load file.

Create a Database Load File

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

   In the example, we login to the host **samqfs1mds**:

```
root@samqfs1mds:~#
```

2. If you already have a current recovery point file, generate the database load file from the contents of the recovery point file. Use the command **samfsrestore -SZ** *output-path-name* **-f** *recoverypoint-file*, where:

   - **-f** specifies *recoverypoint-file* as the path and file name of the input file.
   - **-SZ** causes the command to scan a recovery point file and output a database load file with the path and file name specified by *output-path-name*.

   See the **samfsdump** man page for additional details.

   In the example, we use the daily recovery-point file, **/zfs1/hsmqfs1_recovery/140129**, that we scheduled when we configured the **samqfs1** file system (see "Configure File System Protection" on page 6-50). We send the output to the database load file **/root/hsmqfs1dataload**:

```
root@samqfs1mds:~# samfsrestore -SZ /root/hsmqfs1dataload -f /zfs1/hsmqfs1_
recovery/140129
...
root@samqfs1mds:~#
```

3. If you do not have a current recovery point file, create a database load file now. Change to the Oracle HSM file system's root directory. Then use the command **samfsdump -SZ** *output-path-name*.

   See the **samfsdump** man page for additional details. In the example, we change to the **/hsmqfs1** directory. We send the output to the database load file **/root/hsmqfs1dataload**:

```
root@samqfs1mds:~# cd /hsmqfs1
root@samqfs1mds:~# samfsdump -SZ /root/hsmqfs1dataload
```

4. Next, create the sideband database.

Create the Sideband Database

1. Log in to the MySQL server host as **root**.

   In the example, the MySQL server is hosted on Solaris host **samqfs1mds**:

```
root@samqfs1mds:~#
```

2. In a text editor, open the file **/etc/opt/SUNWsamfs/samdb.conf**.

   In the example, we use the **vi** editor. We start by adding a heading row as a comment:

```
root@samqfs1mds:~# vi /etc/opt/SUNWsamfs/samdb.conf
#FS_NAME:HOST:USER:PASSWORD:NAME:PORT:CLIENT_FLAG:MOUNT_POINT
```

3. In the first column of the **samdb.conf** file, enter the family-set name for the file system, followed by a colon (**:**) as a column separator.

   In the example, we enter the family-set name **samqfs1**:

```
root@samqfs1mds:~# vi /etc/opt/SUNWsamfs/samdb.conf
# /etc/opt/SUNWsamfs/samdb.conf
#FS_NAME:HOST:USER:PASSWORD:NAME:PORT:CLIENT_FLAG:MOUNT_POINT
samqfs1:
```

4. In the second column, enter the hostname for the MySQL database server, followed by a colon (**:**) as a column separator.

In the example, we are co-hosting the database server on the Oracle HSM metadata server host, **samqfs1mds**. So we enter the hostname **localhost**:

```
root@samqfs1mds:~# vi /etc/opt/SUNWsamfs/samdb.conf
# /etc/opt/SUNWsamfs/samdb.conf
#FS_NAME:HOST:USER:PASSWORD:NAME:PORT:CLIENT_FLAG:MOUNT_POINT
samqfs1:localhost:
```

5. In the third column, enter the user name that the Oracle HSM software uses when accessing the MySQL database, followed by a colon (**:**) as a column separator.

   In the example, we have created the user **samqfs** for the purpose of logging in to the database:

```
root@samqfs1mds:~# vi /etc/opt/SUNWsamfs/samdb.conf
# /etc/opt/SUNWsamfs/samdb.conf
#FS_NAME:HOST:USER:PASSWORD:NAME:PORT:CLIENT_FLAG:MOUNT_POINT
samqfs1:localhost:samqfs:
```

6. In the fourth column, enter the password that the Oracle HSM software uses when accessing the MySQL database, followed by a colon (**:**) as a column separator.

   In the example, we use a dummy password, **P^ssw0rd**:

```
root@samqfs1mds:~# vi /etc/opt/SUNWsamfs/samdb.conf
# /etc/opt/SUNWsamfs/samdb.conf
#FS_NAME:HOST:USER:PASSWORD:NAME:PORT:CLIENT_FLAG:MOUNT_POINT
samqfs1:localhost:samqfs:P^ssw0rd:
```

7. In the fifth column, enter the name of the MySQL database, followed by a colon (**:**) as a column separator.

   In the example, we name the database **samqfs1db**:

```
root@samqfs1mds:~# vi /etc/opt/SUNWsamfs/samdb.conf
# /etc/opt/SUNWsamfs/samdb.conf
#FS_NAME:HOST:USER:PASSWORD:NAME:PORT:CLIENT_FLAG:MOUNT_POINT
samqfs1:localhost:samqfs:P^ssw0rd:samqfs1db:
```

8. In the sixth column, enter the TCP/IP port of the database server, followed by a colon (**:**) as a column separator.

   In the example, we enter **0** (zero). If we were using a remote server, a zero (or blank) value would specify the default port, **3306**. But, since we are using **localhost**, the zero serves merely as a place holder:

```
root@samqfs1mds:~# vi /etc/opt/SUNWsamfs/samdb.conf
# /etc/opt/SUNWsamfs/samdb.conf
#FS_NAME:HOST:USER:PASSWORD:NAME:PORT:CLIENT_FLAG:MOUNT_POINT
samqfs1:localhost:samqfs:P^ssw0rd:samqfs1db:0:
```

9. In the seventh column, enter a MySQL client flag, followed by a colon (**:**) as a column separator.

   The MySQL client flag is usually set to **0** (zero). But various combinations of values can be set to enable particular MySQL features. For details, see the MySQL documentation for the **mysql_real_connect()** function.

   In the example, we enter **0** (zero):

```
root@samqfs1mds:~# vi /etc/opt/SUNWsamfs/samdb.conf
# /etc/opt/SUNWsamfs/samdb.conf
#FS_NAME:HOST:USER:PASSWORD:NAME:PORT:CLIENT_FLAG:MOUNT_POINT
samqfs1:localhost:samqfs:P^ssw0rd:samqfs1db:0:0:
```

10. In the eighth and last column, enter the mount point of the Oracle HSM file system. Save the file and close the editor.

    In the example, the file system is mounted at **`/hsmqfs/hsmqfs1`**:

    ```
    root@samqfs1mds:~# vi /etc/opt/SUNWsamfs/samdb.conf
    # /etc/opt/SUNWsamfs/samdb.conf
    #FS_NAME:HOST:USER:PASSWORD:NAME:PORT:CLIENT_FLAG:MOUNT_POINT
    samqfs1:localhost:samqfs:P^ssw0rd:samqfs1db:0:0:/hsmqfs/hsmqfs1
    :wq
    root@samqfs1mds:~#
    ```

11. Create a new database and associated tables. Use the command **`samdb create`** *`family_set`*, where *`family_set`* is the family-set name specified for the Oracle HSM file system in the **`/etc/opt/SUNWsamfs/mcf`** file.

    The default database schema is **`/opt/SUNWsamfs/etc/samdb.schema`**. You may specify an alternative by entering the command as **`samdb create`** *`family_set`* **`-s`** *`schema`*, where *`schema`* is the path and name of a schema file.

    In the example, we use the default schema to create a database for file-system family set **`samqfs1`**.

    ```
    root@samqfs1mds:~# samdb create samqfs1
    ```

12. Populate the database with the data contained in the database load file that you created in the preceding procedure. Use the command **`samdb load`** *`family_set input_file`*, where *`family_set`* is the family-set name specified for the file system in the **`/etc/opt/SUNWsamfs/mcf`** file and *`input_file`* is the path and name of the database load file.

    In the example, we load the database for file-system family set **`samqfs1`** using the database load file **`/root/hsmqfs1dataload`**.

    ```
    root@samqfs1mds:~# samdb load samqfs1 /root/hsmqfs1dataload
    ```

13. Check the database for consistency. Use the command **`samdb check`** *`family_set`*, where *`family_set`* is the family-set name specified for the file system in the **`/etc/opt/SUNWsamfs/mcf`** file.

    The **`samdb check`** command compares the database entries with the current file system metadata. It notes and, where possible, corrects inconsistencies that may have arisen during the load process.

    In the example, we load the database for file-system family set **`samqfs1`** using the database load file **`/root/hsmqfs1dataload`**.

    ```
    root@samqfs1mds:~# samdb check samqfs1
    ```

14. Next, mount the Oracle HSM file system with database support enabled.

Mount the Oracle HSM File System with Database Support Enabled

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

    ```
    root@samqfs1mds:~#
    ```

2. Back up the **`/etc/vfstab`** file.

    ```
    root@samqfs1mds:~# cp /etc/vfstab /etc/vfstab.backup
    ```

3. Open the **/etc/vfstab** file in a text editor, and scroll down to the entry for the file system for which you created the database.

   In the example, we use the **vi** editor. We scroll down to the entry for the **samqfs1** file system:

```
root@samqfs1mds:~# 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       -
...
samqfs1    -        /hsmqfs1  samfs   -     yes      ... ,partial=64
```

4. In the last column of the **/etc/vfstab** file, add **sam_db** to the mount options list for the file-system. Then save the file and close the editor.

   In the example, enable the sideband database on the **samqfs1** file system:

```
root@samqfs1mds:~# 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       -
...
samqfs1    -        /hsmqfs1  samfs   -     yes      ... ,partial=64,sam_db
:wq
root@solaris:~#
```

5. Mount the Oracle HSM archiving file system.

   When a file system is mounted with the **sam_db** option, the Oracle HSM software starts the processes that update the sideband database.

   In the example, we mount the file system **/hsmqfs1**:

```
root@samqfs1mds:~# mount /hsmqfs1
```

6. Next, go to "Configuring Notifications and Logging" on page 11-1

11 Configuring Notifications and Logging

Oracle HSM file systems support automated, remote notification using the Simple Network Management Protocol (SNMP) and provide comprehensive, configurable logging facilities. This chapter outlines the following topics:

- Configuring Simple Network Management Protocol (SNMP).
- Enabling Oracle HSM logging.
- Configuring device logging.
- Configuring log rotation.
- Enabling email alerts.

Configuring Simple Network Management Protocol (SNMP)

Network management applications can monitor Oracle HSM file systems using the Simple Network Management Protocol (SNMP). You can configure the SNMP agent to automatically send *traps* that alert network management stations to faults and configuration changes.

The Oracle HSM Management Information Base (MIB) defines the types of information that SNMP traps provide. These include configuration errors, SCSI **`tapealert`** events, and various kinds of atypical system activity. For more information, see the Management Information Base (MIB) file, **`/var/snmp/mib/SUN-SAM-MIB.mib`**.

When an Oracle HSM trap event occurs, the Solaris kernel system-event notification daemon, **`syseventd`**, calls the script **`/etc/opt/SUNWsamfs/scripts/sendtrap`**. The script then sends the trap to either the local host or to a management station that you specify. The script supports version 2c of the SNMP standard, which is backward compatible with earlier versions of the standard. Note that version 2c exchanges authentication credentials—**`community strings`**—and management data in clear text. See the **`sendtrap`** man page for additional details.

To configure SNMP notification, carry out the following tasks:

- Make sure that all SNMP management stations are listed in the **`/etc/hosts`** file.
- Enable support for SNMP.
- Designate management stations as trap recipients and configure authentication.

This section also includes instructions should you at any point need to disable support for SNMP.

Make Sure that All SNMP Management Stations are Listed in the `/etc/hosts` file

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

   In the example, the Oracle HSM server host is **`samqfs1mds`**:

   ```
   root@samqfs1mds:~#
   ```

2. Open the **`/etc/hosts`** file in a text editor. Make sure that it contains an entry for each host that you intend to use as an SNMP management station.

   In the example, we use the **`vi`** editor. One of the intended management stations, **`management1`**, is listed. But the other, **`management2`**, is not:

   ```
   root@samqfs1mds:~# vi /etc/hosts
   # Internet host table
   ::1 localhost
   127.0.0.1 localhost loghost
   10.0.0.10 server1
   10.0.0.20 management1
   ```

3. If the **`/etc/hosts`** file does not contain entries for some or all of your intended SNMP management station hosts, add the required entries and save the file.

   In the example, we use the **`vi`** editor to add the missing management station, **`management2`**:

   ```
   root@samqfs1mds:~# vi /etc/hosts
   # Internet host table
   ::1 localhost
   127.0.0.1 localhost loghost
   10.0.0.10 server1
   10.0.0.20 management1
   10.0.0.30 management2
   ```

4. When the **`/etc/hosts`** file contains entries for all of your intended SNMP management station hosts, close the editor.

   ```
   root@samqfs1mds:~# vi /etc/hosts
   ...
   10.0.0.20 management1
   10.0.0.30 management2
   :wq
   root@samqfs1mds:~#
   ```

5. Next, enable support for SNMP.

Enable Support for SNMP

By default, support for SNMP notifications is enabled, so no action is needed unless SNMP support has been disabled at some point. If you need to re-enable SNMP support, proceed as follows:

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

   In the example, the Oracle HSM server host is **`samqfs1mds`**:

   ```
   root@samqfs1mds:~#
   ```

2. Open the file **`/etc/opt/SUNWsamfs/defaults.conf`** in a text editor. Locate the line **`alerts = off`**.

The directive **alerts = off** disables SNMP support. In the example, we open the file in the **vi** editor and locate the line:

```
root@samqfs1mds:~# 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.
...
alerts = off
```

3. To enable support for SNMP notifications, change the value of the **alerts** directive to **on**. Then save the file and close the editor.

```
root@samqfs1mds:~# 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.
...
alerts = on
:wq
root@samqfs1mds:~#
```

4. Tell the Oracle HSM service to reread the **defaults.conf** file and reconfigure itself accordingly. Use the command **samd config**.

```
root@samqfs1mds:~# samd config
root@samqfs1mds:~#
```

5. Next, designate management stations as trap recipients and configure authentication.

Designate Management Stations as Trap Recipients and Configure Authentication

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

   In the example, the Oracle HSM server host is **samqfs1mds**:

```
root@samqfs1mds:~#
```

2. Open the **/etc/opt/SUNWsamfs/scripts/sendtrap** file in a text editor, and locate the line starting **TRAP_DESTINATION=**.

   The **sendtrap** file is a configurable shell script. In the example, we open the file in the **vi** editor:

```
root@samqfs1mds:~# vi /etc/opt/SUNWsamfs/scripts/sendtrap
# /etc/opt/SUNWsamfs/scripts/sendtrap
#!/usr/bin/sh
# sendtrap:
#  This script gets invoked by the sysevent configuration file.
#  This is not expected to be run as a stand-alone program
...
# CONFIGURATION PARAMETERS:
TRAP_DESTINATION=`hostname`
```

3. In the line **TRAP_DESTINATION=`**, replace the text within the single quotation marks with a space-delimited list of one or more trap recipients, each of the form *hostname*:*port*, where *hostname* is the hostname for a management station, as listed in **/etc/hosts**, and *port* is the port on which the host listens for traps.

   By default, traps are sent to UDP port **161** of the **localhost**. In the example, we add hosts **management1** and **management2** to the default, **localhost**. The

localhost and **management1** use the default port, while **management2** uses a custom port, **1161**:

```
...
# CONFIGURATION PARAMETERS:
TRAP_DESTINATION=`localhost:161 management1:161 management1:1161`
```

4. Scroll down to the line that sets the community string, **COMMUNITY="public"**.

   The community string is the shared, plain-text password that authenticates agents and management stations in SNMP version 2c. The default value is the SNMP standard, **public**.

```
...
# CONFIGURATION PARAMETERS:
TRAP_DESTINATION=`localhost:161 management1:161 management1:1161`
...
COMMUNITY="public"
```

5. Set the **COMMUNITY=""** directive to the value used by your management stations. Then save the file and close the editor.

   Do not edit anything else in the file. **COMMUNITY=""** and **TRAP_DESTINATION=``** are the only editable parameters.

   Note that the default SNMP community string, **public**, is insecure. So your network administrator may mandate a more secure choice. SNMP version 2c allows up to 32 alphanumeric characters. In the example, we set the community string to **Iv0wQh2th74bVVt8of16t1m3s8it4wa9**.

```
...
# CONFIGURATION PARAMETERS:
TRAP_DESTINATION=`localhost:161 management1:163 management1:1162`
...
COMMUNITY="Iv0wQh2th74bVVt8of16t1m3s8it4wa9"
:wq
root@samqfs1mds:~#
```

6. Next, enable Oracle HSM application logging.

Disable Support for SNMP

The remote notification facility is enabled by default. If you want to disable remote notification, proceed as follows:

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

   In the example, the Oracle HSM server host is **samqfs1mds**:

```
root@samqfs1mds:~#
```

2. Open the **/etc/opt/SUNWsamfs/defaults.conf** file in a text editor. Locate the line **#alerts = on**.

   In the example, we use the **vi** editor:

```
root@samqfs1mds:~# 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.
...
#alerts = on
root@samqfs1mds:~#
```

3. To disable support for SNMP notifications, delete the hash character (**#**) to uncomment the line and change the value of **alerts** to **off**. Then save the file and close the editor.

   ```
   root@samqfs1mds:~# 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.
   ...
   alerts = off
   :wq
   root@samqfs1mds:~#
   ```

4. Tell the Oracle HSM service to reread the **defaults.conf** file and reconfigure itself accordingly. Use the command **samd config**.

   ```
   root@samqfs1mds:~# samd config
   root@samqfs1mds:~#
   ```

5. Stop here. SNMP support is disabled.

Enabling Oracle HSM Logging

The **/var/adm/sam-log** file records status and error information for the Oracle HSM application and its component daemons and processes. To set up the logging process, proceed as follows:

Enable Oracle HSM Application Logging

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

   In the example, the Oracle HSM server host is **samqfs1mds**:

   ```
   root@samqfs1mds:~#
   ```

2. Open the **/etc/syslog.conf** file in a text editor.

   In the example, we open the file in the **vi** editor:

   ```
   root@samqfs1mds:~# vi /etc/syslog.conf
   # syslog configuration file ...
   *.err;kern.notice;auth.notice                   /dev/sysmsg
   *.err;kern.debug;daemon.notice;mail.crit        /var/adm/messages
   *.alert;kern.err;daemon.err                     operator
   *.alert                                         root
   ...
   ```

3. In the **/etc/syslog.conf** file, add a line consisting of the string **local7.debug**, one or more Tab characters, and the path string **/var/adm/sam-log**. Then save the file and close the editor.

   In the example, we also add a comment:

   ```
   root@samqfs1mds:~# vi /etc/syslog.conf
   # syslog configuration file ...
   *.err;kern.notice;auth.notice                   /dev/sysmsg
   *.err;kern.debug;daemon.notice;mail.crit        /var/adm/messages
   *.alert;kern.err;daemon.err                     operator
   *.alert                                         root
   ...
   ```

```
# Oracle HSM logging
local7.debug    /var/adm/sam-log
:wq
root@samqfs1mds:~#
```

4. Create the log file **/var/adm/sam-log**. Use the command **touch /var/adm/sam-log**.

```
root@samqfs1mds:~# touch /var/adm/sam-log
root@samqfs1mds:~#
```

5. Tell the Solaris **syslogd** daemon to re-read its configuration files and start Oracle HSM logging. Use the command **pkill -HUP syslogd**.

   Whenever it receives a HUP signal, the **syslogd** logging service re-reads the **/etc/syslog.conf** configuration file, closes all open log files, and then opens the log files that are listed in **syslog.conf**. When the command runs, Oracle HSM logging is enabled:

```
root@samqfs1mds:~# pkill -HUP syslogd
root@samqfs1mds:~#
```

6. Go to configuring device logging.

Configuring Device Logging

The device logging facility provides error information specific to individual hardware devices (it does not collect soft media errors). Each device has its own log file, named with the corresponding equipment ordinal number and stored in the **/var/opt/SUNWsamfs/devlog/** directory.

Device logs can grow rapidly. So, by default, the system logs a limited set of event data: **err**, **retry**, **syserr**, and **date**. Later, if problems arise, you can use the **samset** command to log additional events on a per-device basis (see the **devlog** section of the **samset** man page for details).

Enable the Device Log in the `defaults.conf` File

To enable basic device logging, proceed as follows:

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

   In the example, the Oracle HSM server host is **samqfs1mds**:

```
root@samqfs1mds:~#
```

2. Open the **/etc/opt/SUNWsamfs/defaults.conf** in a text editor.

   In the example, we open the file in the **vi** editor:

```
root@samqfs1mds:~# 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.
...
```

3. In the **defaults.conf** file, add a line that defines the default level of device logging that you require. Enter the directive **devlog** *equipment-number loggable-events*, where:

- *equipment-number* is either the keyword **all**, for all equipment defined in the **/etc/opt/SUNWsamfs/mcf** file, or the equipment ordinal that identifies a specific piece of equipment defined in the **mcf**.
- *loggable-events* is a space-delimited list of the defaults: **err retry syserr date**

  See the **devlog** section of the **samset** man page for a comprehensive list of event types. But, to minimize log sizes, use the default selections. For diagnostic purposes, the **samset** command can selectively enable additional events as needed.

In the example, we enable device logging for **all** devices using the default logging level:

```
[samfs-mds1]root@solaris:~# 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.
...
devlog all err retry syserr date
```

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

```
[samfs-mds1]root@solaris:~# 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.
...
devlog all err retry syserr date
:wq
[samfs-mds1]root@solaris:~#
```

5. Tell the Oracle HSM service to reread the **defaults.conf** file and reconfigure itself accordingly. Use the command **samd config**.

```
[samfs-mds1]root@solaris:~# samd config
[samfs-mds1]root@solaris:~#
```

6. Next, set up automatic rotation for Oracle HSM log files.

Configuring Log Rotation

Log files can grow rapidly, consuming large amounts of space and making the logs hard to use. So you should configure automatic log rotation for Oracle HSM logs. The software includes a script for this purpose, **log_rotate.sh**, that you can run from the Solaris **crontab** file.

For each log that you intend to rotate, you create two **crontab** entries. The first one runs the **log_rotate.sh** script at the desired time. If the target log file has reached a specified minimum size (the default is **100000** bytes), the script renames it and deletes the oldest existing copy (seven are kept at any one time). The second **crontab** entry tells the Solaris logging daemon, **syslogd**, to restart logging with a new log file.

Set Up Automatic Rotation for Oracle HSM Log Files

Consider rotating the following logs:

- The Oracle HSM log file, **sam-log**, located as specified in the **/etc/syslog.conf** file.

- The device log files located in the **/var/opt/SUNWsamfs/devlog/** directory.
- The stager log files specified in the **/etc/opt/SUNWsamfs/stager.cmd** file.
- The releaser log files specified in the **/etc/opt/SUNWsamfs/releaser.cmd** file.
- The recycler log files specified in the **/etc/opt/SUNWsamfs/recycler.cmd** file.

Archiver log files should not be rotated! The log information is valuable for analytics and file-system recovery. For proper handling of archiver logs, see "Configure File System Protection" on page 6-50.

Once you have decided on the logs that should be rotated, proceed as follows for each log:

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

   In the example, the Oracle HSM server host is **samqfs1mds**:

   ```
   root@samqfs1mds:~#
   ```

2. Copy the script file **log_rotate.sh** from **/opt/SUNWsamfs/examples/** (the uninstalled location) to **/etc/opt/SUNWsamfs/scripts/**.

   ```
   root@samqfs1mds:~# cp /opt/SUNWsamfs/examples/log_rotate.sh
   /etc/opt/SUNWsamfs/scripts/
   root@samqfs1mds:~#
   ```

3. Open the **root** user's **crontab** file for editing. Use the command **crontab -e**.

   The **crontab** command opens an editable copy of the **root** user's **crontab** file in the text editor specified by the **EDITOR** environment variable (for full details, see the Solaris **crontab** man page). In the example, we use the **vi** editor:

   ```
   [samfs-mds1]root@solaris:~# crontab -e
   #ident"%Z%%M%%I%%E% SMI"
   # Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
   # Use is subject to license terms.
   # The root crontab should be used to perform accounting data collection.
   10 3 * * * /usr/sbin/logadm
   ...
   30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh
   ```

4. On a new line, specify the time of day when the log file will be rotated by entering *minutes hour* *** *** *day-of-the-week*, where:

   - *minutes* is an integer in the range [**0-59**] that species the minute when the job starts.
   - *hour* is an integer in the range [**0-23**] that species the hour when the job starts.
   - ***** (asterisk) specifies unused values.

     For a task that runs daily, the values for day of the month [**1-31**] and month [**1-12**] are unused.
   - *day-of-the-week* is an integer in the range [**0-6**], starting from Sunday (**0**).
   - Spaces separate the fields in the time specification.

   In the example, we schedule log rotation to begin at 3:10 AM every Sunday.

   ```
   [samfs-mds1]root@solaris:~# crontab -e
   #ident"%Z%%M%%I%%E% SMI"
   # Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
   # Use is subject to license terms.
   ```

```
# The root crontab should be used to perform accounting data collection.
10 3 * * * /usr/sbin/logadm
...
30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh
10 3 * * 0
```

5. Continuing on the same line, enter the path and name of the shell script file that rotates Oracle HSM logs, **`/etc/opt/SUNWsamfs/scripts/log_rotate.sh`**, followed by a space.

```
[samfs-mds1]root@solaris:~# crontab -e
#ident"%Z%%M%%I%%E% SMI"
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
# The root crontab should be used to perform accounting data collection.
10 3 * * * /usr/sbin/logadm
...
30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh
10 3 * * 0 /etc/opt/SUNWsamfs/scripts/log_rotate.sh
```

6. Continuing on the same line, enter the name of the log that you need to rotate and the minimum file size file to rotate. Enter the text *`samfslog`* [*`minimum-size`*] where *`samfslog`* is the path to an Oracle HSM log file and [*`minimum-size`*] is an optional integer that specifies the smallest file-size in bytes that the script rotates (the default is **`100000`**).

   In the example, we need to rotate the **`/var/adm/sam-log`**. We accept the default minimum size:

```
[samfs-mds1]root@solaris:~# crontab -e
#ident"%Z%%M%%I%%E% SMI"
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
# The root crontab should be used to perform accounting data collection.
10 3 * * * /usr/sbin/logadm
...
30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh
10 3 * * 0 /etc/opt/SUNWsamfs/scripts/log_rotate.sh /var/adm/sam-log
```

7. Start a new line. Create a **`crontab`** entry that starts 10 minutes after the **`log_rotate.sh`** script. This entry tells the Solaris **`syslogd`** daemon to close its old log file and resume logging to a new file. Enter the line *`minutes hour`* **`* *`** *`day-of-the-week`* **`/bin/kill -HUP `/bin/cat /etc/syslog.pid``**, where *`minutes hour`* **`* *`** *`day-of-the-week`* specifies a time 10 minutes later than the time specified in the previous step.

   In the example, the entry restarts Oracle HSM logging at 3:20 AM every Sunday:

```
[samfs-mds1]root@solaris:~# crontab -e
#ident"%Z%%M%%I%%E% SMI"
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
# The root crontab should be used to perform accounting data collection.
10 3 * * * /usr/sbin/logadm
...
30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh
10 3 * * 0 /etc/opt/SUNWsamfs/scripts/log_rotate.sh /var/adm/sam-log
20 3 * * 0  /bin/kill -HUP `/bin/cat /etc/syslog.pid`
```

8. Save the file, and close the editor.

```
[samfs-mds1]root@solaris:~# crontab -e
#ident"%Z%%M%%I%%E% SMI"
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
# The root crontab should be used to perform accounting data collection.
10 3 * * * /usr/sbin/logadm
...
30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh
10 3 * * 0 /etc/opt/SUNWsamfs/scripts/log_rotate.sh /var/adm/sam-log
20 3 * * 0  /bin/kill -HUP `/bin/cat /etc/syslog.pid`
:wq
[samfs-mds1]root@solaris:~#
```

9. Repeat this procedure until you have configured log rotation for all required logs.
10. Now, enable email alerts, if required.
11. Otherwise, go to "Backing Up the Oracle HSM Configuration" on page 13-1.

Enabling Email Alerts

Email alerts are best set up from the Oracle HSM Manager graphical user interface. See the online help for details.

If you must configure email alerts from the commandline interface, see the **`defaults.conf`**, **`archiver.sh`**, **`dev_down.sh`**, **`load_notify.sh`**, **`recycler.sh`**, **`archiver.cmd`**, **`recycler.cmd`**, and **`notify.cmd`** man pages.

Your Oracle HSM system is now configured. But before you begin to use it, protect your work. For instructions, see "Backing Up the Oracle HSM Configuration" on page 13-1.

12 Tuning I/O Characteristics for Special Needs

The basic file-system configuration steps described in the preceding chapters provide optimal, balanced performance in most situations. So if you are at all uncertain of how your application behaves, you are usually better off leaving the settings in this section at their default values. However, if your application makes unusually consistent or unusually large I/O requests, overall performance may benefit from tuning or changing the way in which the file system handles physical I/O.

Physical I/O is most efficient when all or most reads and writes begin and end exactly on the 512-byte boundary of a disk sector. Disk I/O can only occur in sector-sized chunks. So, when an I/O request straddles a sector boundary, the system must perform additional operations to separate the application data from unrelated data in the same sector. It must insure that the latter is not corrupted in the process. In the worst case, when writing across sectors, the file system has to read the sector, modify the sector data in memory, and then write the sector back to disk. The additional mechanical activity alone makes such read-modify-write operations extremely costly in performance terms.

Unfortunately, most applications need to read and write data in varied sizes that are not well aligned on sector boundaries. For this reason, like many file systems, Oracle HSM uses *paged I/O* by default. The file system handles immediate I/O requests from the application by reading from or writing to a data cache in Solaris paged memory. The file system asynchronously updates the cache with more efficiently sized and better aligned physical reads and writes. Whenever it reads data from disk, it can make the most of the physical I/O by anticipating upcoming reads and loading the corresponding data into cache in the same operation. Most I/O requests are thus met using data cached in virtual memory pages, with no additional physical disk activity. Paged I/O uses memory and imposes some additional load on the system CPU, but, in most cases, these costs are more than offset by greater efficiency of physical I/O.

In a few cases, however, the extra overhead associated with paged I/O is not offset by its advantages. Applications that always perform well-aligned I/O and applications that can be tuned to do so gain nothing from page caching. Applications that perform extremely large I/Os may also gain little from page caching, because only the first and last sectors are misaligned, and because the large I/Os may, in any case, be too large to be retained in cache. Finally, applications that stream telemetry data, surveillance video, or other types of real-time information may risk loss of irrecoverable data if writes are not immediately committed to non-volatile storage. In these cases, it may be better to use *direct I/O*. When direct I/O is specified, the file system transfers data between application memory and the disk device directly, bypassing the page cache.

Oracle HSM gives you considerable latitude when it comes to selecting and tuning I/O caching behavior. Once you understand the I/O characteristics of your application and have carried out the tasks described in "Tune Solaris System and Driver Parameters for Anticipated File System I/O" on page 2-2, select your approach as follows:

- If your application consistently makes small, variably sized, and/or misaligned I/O requests, accept the Oracle HSM default settings. Do not make any of the changes in this section.
- If your application makes variably sized but larger than average, misaligned I/O requests, optimize paged I/O for larger data transfers.
- If your application makes a mix of well-aligned or very large I/O requests and small, misaligned requests, enable switching between paged and direct I/O.
- If your application *consistently* makes well-aligned or very large I/O requests, configure the file system to use direct I/O exclusively.
- If applications running on shared file-system clients consistently open large numbers of files, increase the directory name lookup cache size.

Optimize Paged I/O for Larger Data Transfers

Paged I/O can be tuned to better match application and hardware characteristics. Reads to cache and writes from cache should be large enough to transfer the average amount of data that the application transfers or the maximum amount of data that the physical storage can transfer, whichever is larger. If we fail to tune page caching behavior for either, the cache will be under utilized, application I/O requests will require more physical I/O, and overall system performance will suffer.

For example, consider the difference between an **md** data device that is implemented on a single disk volume and an **md** device implemented on a 3+1 RAID 5 volume group. If we were to handle each write request from the application by writing a single 64 kilobyte disk allocation unit (DAU) from cache to the latter, ignoring the additional bandwidth possible with the multiple-disk device, the RAID device would have to split the I/O into three smaller, still less efficient 21- and 22-kilobyte fragments before writing data out to the three data disks in the RAID group. Fulfilling 64-kilobyte I/O requests from the application would thus require significantly more work using this configuration than it would have required had we used the page cache to assemble the requests into a single, 3-DAU, 192-kilobyte I/O. If the application could—or could be tuned to—make I/O requests in even multiples of the device bandwidth—192-, 384-, or 576-kilobytes—then we could cache even more data and transfer more with each physical I/O, further reducing overhead and boosting performance accordingly.

So, identify the I/O requirements of your application and understand the I/O properties of your hardware. Then proceed as follows.

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

   ```
   root@solaris:~#
   ```

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

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

3. Open the **/etc/vfstab** file in a text editor, and locate the row for the file system that needs tuning.

   In this example, the file system is named **qfsma**:

   ```
   root@solaris:~# 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       -
...
qfsma      -        /qfsma    samfs   -     yes      ...
```

4. In the **Mount Options** field for the file system, add the **writebehind=***n* mount option, where *n* is a multiple of **8** kilobytes. Use a comma (no spaces) to separate mount options. Save the file and close the editor.

   The **writebehind** option determines how much of a given file can queue up in the page cache before the cache is flushed to disk. Setting the parameter to a higher value improves performance, because a large queue consolidates multiple small application writes into fewer, larger, more efficient physical I/Os. Setting the parameter lower better protects data, because changes are written to non-volatile storage sooner.

   The default value is **512** kilobytes (eight 64-kilobyte DAUs), which generally favors large-block, sequential I/O. But in this example, the family set contains two **md** disk devices with striped file allocation. The stripe width is one 64-kilobyte DAU, for a write of 128 kilobytes to the two **md** devices. The **md** devices are 3+1 RAID 5 groups. So we want to write at least 128 kilobytes to each of the three data spindles, for a total write of at least **768** kilobytes (96 groups of 8 kilobytes each):

```
#File
#Device    Device   Mount     System  fsck  Mount    Mount
#to Mount  to fsck  Point     Type    Pass  at Boot  Options
#--------  -------  --------  ------  ----  -------  --------------------
/devices   -        /devices  devfs   -     no       -
...
qfsma      -        /qfsma    samfs   -     yes      ...,writebehind=768
:wq
root@solaris:~#
```

5. Test the I/O performance of the file system and adjust the **writebehind** setting as needed.

6. Re-open the **/etc/vfstab** file in a text editor. In the **Mount Options** field for the file system, add the **readahead=***n* mount option, where *n* is a multiple of 8 kilobytes. Use a comma (no spaces) to separate mount options. Save the file and close the editor.

   The **readahead** option determines the amount of data that is read into cache during a single physical read. When an application appears to be reading sequentially, the file system caches upcoming blocks of file data during each physical read. A series of application read requests can then be handled from cache memory, consolidating several application read requests into a single physical I/O request.

   The default value is **1024** kilobytes (sixteen 64-kilobyte DAUs), which generally favors large-block, sequential I/O. If a database or similar application performs its own readahead, set Oracle HSM **readahead** to **0** to avoid conflicts. Otherwise, **readahead** should generally be set to cache the maximum data that a single physical I/O can transfer. If the **readahead** setting is smaller than the amount of data that applications typically request and that devices can supply, fulfilling an application I/O request requires more physical I/Os than necessary. However, if **readahead** is set excessively high, it may consume enough memory to degrade overall system performance. In the example, we set **readahead** to **736** kilobytes (thirty-six 64-kilobyte DAUs).

```
#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       -
...
qfsma      -        /qfsma    samfs   -     yes      ...,readahead=736
:wq
root@solaris:~#
```

7. Test the I/O performance of the file system and adjust the **readahead** setting as needed.

   Increasing the size of the **readahead** parameter increases the performance of large file transfers, but only up to a point. So test the performance of the system after resetting the **readahead** size. Then adjust **readahead** size upwards until you see no more improvement in transfer rates.

Enable Switching Between Paged and Direct I/O

You can configure Oracle HSM file systems to switch between paged and direct I/O when doing so better suits the I/O behavior of your application. You specify the sector-alignment and minimum-size characteristics of reads and writes that might benefit from direct I/O and then set the number of qualifying reads and writes that should trigger the switch. Proceed as follows:

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

   ```
   root@solaris:~#
   ```

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

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

3. Open the **/etc/vfstab** file in a text editor, and locate the row for the file system that you want to configure.

   In this example, the file system is named **qfsma**:

   ```
   root@solaris:~# vi /etc/vfstab
   #File     Device                       Mount
   #Device   to     Mount    System fsck at    Mount
   #to Mount fsck   Point    Type   Pass Boot  Options
   #-------- ------ -------- ------ ---- ----- ----------------------------
   /devices  -      /devices devfs  -    no    -
   /proc     -      /proc    proc   -    no    -
   ...
   qfsma     -      /qfsma   samfs  -    yes   stripe=1
   ```

4. To set a threshold size for starting direct I/O for read requests that align well with 512-byte sector boundaries, add the **dio_rd_form_min=***n* mount option to the **Mount Options** field for the file system, where *n* is a number of kilobytes. Use a comma (no spaces) to separate mount options.

   By default, **dio_rd_form_min=256** kilobytes. In the example, we know that our application does not produce consistently well-aligned reads until it requests a read of at least 512 kilobytes. So we change the threshold size for well-aligned direct reads to **512**:

   ```
   #File     Device                       Mount
   ```

```
#Device   to     Mount    System fsck at    Mount
#to Mount fsck   Point    Type   Pass Boot  Options
#-------- ------ -------- ------ ---- ----- -----------------------------
/devices  -      /devices devfs  -    no    -
/proc     -      /proc    proc   -    no    -
...
qfsma     -      /qfsma   samfs  -    yes   stripe=1,dio_rd_form_min=512
```

5. To set a threshold size for starting direct I/O for write requests that align well with 512-byte sector boundaries, add the **`dio_wr_form_min=`***n* mount option to the **`Mount Options`** field for the file system, where *n* is a number of kilobytes. Use a comma (no spaces) to separate mount options.

   By default, **`dio_wr_form_min=256`** kilobytes. In the example, we know that our application does not produce consistently well-aligned writes until it requests a write of at least a megabyte. So we change the threshold size for well-aligned direct writes to **`1024`** kilobytes:

```
#File     Device                      Moun
#Device   to     Mount    System fsck at    Mount
#to Mount fsck   Point    Type   Pass Boot  Options
#-------- ------ -------- ------ ---- ----- -----------------------------
/devices  -      /devices devfs  -    no    -
/proc     -      /proc    proc   -    no    -
...
qfsma     -      /qfsma   samfs  -    yes   ...,dio_wr_form_min=1024
```

6. To set a threshold size for starting direct I/O for read requests that do not align well with 512-byte sector boundaries, add the **`dio_rd_ill_min=`***n* mount option to the **`Mount Options`** field for the file system, where *n* is a number of kilobytes. Use a comma (no spaces) to separate mount options.

   By default, **`dio_rd_ill_min=0`** kilobytes, so direct I/O is not used for misaligned reads. In the example, we know that our application generally makes misaligned read requests for small chunks of data. Much of this data is subsequently reread. So page caching is likely to be beneficial for these reads. Switching to direct I/O would cause needless additional physical I/O and reduced performance. So we accept the default and make no changes to the **`vfstab`** file:

```
#File     Device                      Mount
#Device   to     Mount    System fsck at    Mount
#to Mount fsck   Point    Type   Pass Boot  Options
#-------- ------ -------- ------ ---- ----- -----------------------------
/devices  -      /devices devfs  -    no    -
/proc     -      /proc    proc   -    no    -
...
qfsma     -      /qfsma   samfs  -    yes   ...,dio_wr_form_min=1024
```

7. To set a threshold size for starting direct I/O for write requests that do not align well with 512-byte sector boundaries, add the **`dio_wr_ill_min=`***n* mount option to the **`Mount Options`** field for the file system, where *n* is a number of kilobytes. Use a comma (no spaces) to separate mount options.

   By default, **`dio_wr_ill_min=0`** kilobytes, so direct I/O is not used for misaligned writes. Misaligned writes can be particularly costly in performance terms, because the system has to read, modify, and write sectors. In the example, however, we know that our application occasionally makes large, single write requests that do not fall on sector boundaries. Since read-write-modify operations are limited to the beginning and end of a large block of sequential sectors, the benefits of direct I/O outweigh those of paged I/O. So we set **`dio_wr_ill_min=2048`** kilobytes:

In this example, we change the default threshold value for using direct I/O during writes with misaligned data to **`2048`** kilobytes:

```
#File     Device                       Mount
#Device   to     Mount    System fsck at    Mount
#to Mount fsck   Point    Type   Pass Boot  Options
#-------- ------ -------- ------ ---- ----- ----------------------------
/devices  -      /devices devfs  -    no    -
/proc     -      /proc    proc   -    no    -
...
qfsma     -      /qfsma   samfs  -    yes   ...,dio_wr_ill_min=2048
```

8. To enable direct I/O for reads, add the **`dio_rd_consec=`***n* mount option to the **`Mount Options`** field, where *n* is the number of consecutive I/O transfers that must meet the size and alignment requirements specified above in order to trigger the switch to direct I/O. Select a value that selects for application operations that benefit from direct I/O. Use a comma (no spaces) to separate mount options.

   By default, **`dio_rd_consec=0`**, so I/O switching is disabled.In the example, we know that, once our application requests three, successive, well-aligned reads of at least the minimum size specified by **`dio_rd_form_min`**, 512 kilobytes, it will continue to do so for long enough to make direct I/O worthwhile. The minimum size specified by **`dio_rd_form_min`** is the default, **`0`**, so enabling direct I/O will not affect misaligned read requests. So we set **`dio_rd_consec=3`**:

```
#File     Device                       Mount
#Device   to     Mount    System fsck at    Mount
#to Mount fsck   Point    Type   Pass Boot  Options
#-------- ------ -------- ------ ---- ----- ----------------------------
/devices  -      /devices devfs  -    no    -
/proc     -      /proc    proc   -    no    -
...
qfsma     -      /qfsma   samfs  -    yes   ...,dio_rd_consec=3
```

9. To enable direct I/O for writes, add the **`dio_wr_consec=`***n* mount option to the **`Mount Options`** field, where *n* is the number of consecutive I/O transfers that must meet the size and alignment requirements specified above in order to trigger the switch to direct I/O. Select a value that selects for application operations that benefit from direct I/O. Use a comma (no spaces) to separate mount options.

   By default, **`dio_wr_consec=0`**, so I/O switching is disabled. In the example, we know that, once our application requests two, successive, well-aligned writes of at least the minimum size specified by **`dio_wr_form_min`**, 1024 kilobytes, it will continue to do so for long enough to make direct I/O worthwhile. We also know that two successive, misaligned writes larger than **`dio_wr_form_min`**, 2048 kilobytes, will be large enough that the misalignment will matter relatively little. So we set **`dio_wr_consec=2`**:

```
#File     Device                       Mount
#Device   to     Mount    System fsck at    Mount
#to Mount fsck   Point    Type   Pass Boot  Options
#-------- ------ -------- ------ ---- ----- ----------------------------
/devices  -      /devices devfs  -    no    -
/proc     -      /proc    proc   -    no    -
...
qfsma     -      /qfsma   samfs  -    yes   ...,dio_wr_consec=2
```

10. Save the **`vfstab`** file, and close the editor.

```
#File     Device                       Mount
#Device   to     Mount    System fsck at    Mount
```

```
#to Mount fsck   Point    Type   Pass Boot  Options
#-------- ------ -------- -----  ---- ----- ----------------------------
/devices  -      /devices devfs  -    no    -
/proc     -      /proc    proc   -    no    -
...
qfsma     -      /qfsma   samfs  -    yes   ...,dio_wr_consec=2
:wq
root@solaris:~#
```

11. Mount the modified file system:

```
root@solaris:~# mount /qfsms
root@solaris:~#
```

Configure the File System to Use Direct I/O Exclusively

When the I/O characteristics of applications make exclusive use of direct I/O desirable, you can mount entire file systems using the **forcedirectio** mount option (for information on how to specify direct I/O for individual files or directories, see the Oracle HSM **setfa** man page).

To mount a file system to use direct I/O exclusively, proceed as follows:

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

```
root@solaris:~#
```

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

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

3. Open the **/etc/vfstab** file in a text editor, and locate the row for the file system where you want to use direct I/O.

   In this example, the file system is named **qfsma**:

```
root@solaris:~# 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       -
...
qfsma      -        /qfsma    samfs   -     yes      stripe=1
```

4. In the **Mount Options** field for the file system, add the **forcedirectio** mount option. Use a comma (no spaces) to separate mount options. Save the file, and close the editor.

```
#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       -
...
qfsma      -        /qfsma    samfs   -     yes      stripe=1,forcedirectio
:wq
root@solaris:~#
```

5. Mount the modified file system:

```
root@solaris:~# mount /qfsms
root@solaris:~#
```

Increase the Directory Name Lookup Cache Size

The default size of the Oracle Solaris directory name lookup cache (DNLC) on the metadata server may prove inadequate when the clients of a shared file system open many files at the same time. The metadata server looks up file names on behalf of all clients, so file system performance may suffer under these conditions.

If you anticipate this kind of work load, change the value of directory name lookup cache-size parameter, **`ncsize`**, to double or triple the default size. For instructions, see the *Oracle Solaris Tunable Parameters Reference Manual*, available in the *Oracle Solaris Information Library* (see the "Available Documentation" section of the preface).

13 Backing Up the Oracle HSM Configuration

Once you have completed Oracle Hierarchical Storage Manager and StorageTek QFS Software configuration, protect your investment by backing up configuration files and related information. Carry out the following tasks:

- Create a backup location for your Oracle HSM configuration.
- Run **samexplorer** and safely store the report.
- Manually backup the Oracle HSM configuration.

Create a Backup Location for Your Oracle HSM Configuration

Proceed as follows:

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

   In the example, the host name is **samqfs1mds**:

   ```
   root@samqfs1mds:~#
   ```

2. Select a storage location for backup copies of your Oracle HSM configuration. Select an independent file system that can be mounted on the file system host.
3. Make sure that the selected file system does not share any physical devices with the archiving file system.

   Do not store recovery point files in the file system that they are meant to protect. Do not store recovery point files on logical devices, such as partitions or LUNs, that reside on physical devices that also host the archiving file-system.
4. In the selected file system, create a directory to hold the configuration information. Use the command **mkdir** *mount-point***/***path*, where *mount-point* is the mount point for the selected independent file system and *path* is the path and name of the chosen directory.

   In the example, we have created the directory **/zfs1/sam_config** on the independent file system **/zfs1**:

   ```
   root@samqfs1mds:~# mkdir /zfs1/sam_config
   ```

5. Next, run **samexplorer** and safely store the report.

Run samexplorer and Safely Store the Report

The **samexplorer** is a diagnostic tool that captures and reports comprehensive configuration and status information for the Oracle HSM software and file systems. Oracle support services personnel use the output when troubleshooting. So creating a baseline **samexplorer** report whenever you configure or reconfigure Oracle HSM software and file systems is a good idea.

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

   In the example, the hostname is **samqfs1mds**:

   ```
   root@samqfs1mds:~#
   ```

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 for the selected independent file system and *path* is the path and name of the chosen directory.

   In the example, we create the directory **/zfs1/sam_config/explorer**:

   ```
   root@samqfs1mds:~# mkdir /zfs1/sam_config/explorer
   root@samqfs1mds:~#
   ```

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 is named **/tmp/SAMreport.***hostname***.***YYYYMMDD***.***hhmmz***.tar.gz**.

   In the example, we create the file **samhost1.20140130.1659MST.tar.gz** in the directory **/zfs1/sam_config/explorer/** (note that the command below is entered as a single line—the line break is escaped by the backslash):

   ```
   root@samqfs1mds:~# samexplorer \
   /zfs1/sam_config/explorer/samhost1.20140130.1659MST.tar.gz

        Report name:     /zfs1/sam_
   config/explorer/samhost1.20140130.1659MST.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/samhost1.20140130.1659MST.tar.gz

   root@samqfs1mds:~#
   ```

4. Repeat this procedure whenever you significantly reconfigure your file systems.

5. Next, manually backup the Oracle HSM configuration.

Manually Backup the Oracle HSM Configuration

While the **samexplorer** utility captures much of your Oracle HSM configuration information, for full redundancy, you should carry out the following procedure after ever major configuration effort:

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

   In the example, the hostname is **samqfs1mds**:

```
root@samqfs1mds:~#
```

2. In the directory 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 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 **`/hsmqfs1`**. So we have created the directory **`/zfs1/sam_config/samconfig`**:

```
root@samqfs1mds:~# mkdir /zfs1/sam_config/samconfig
root@samqfs1mds:~#
```

3. In the chosen directory, 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 use the date to name the subdirectory:

```
root@samqfs1mds:~# mkdir /zfs1/sam_config/samconfig/20140127
root@samqfs1mds:~#
```

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
```

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`** (note that the command below is entered as a single line—the line break is escaped by the backslash):

```
root@samqfs1mds:~# dump_cat -V /var/opt/SUNWsamfs/catalog/library1cat \
> /zfs1/sam_config/20140513/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 **crontab** entries to manage creation of recovery points and log rotation, you save a copy now.

8. 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.

9. In the chosen subdirectory, save copies of downloaded Oracle HSM, Solaris, and Solaris Cluster packages so that you can restore the software quickly, should it become necessary.

10. Stop here. You have backed up your configuration, and your file systems are ready to use.

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
- Other Equipment and Media Types

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. See "Create a Cloud Library Parameter File" on page 6-13.
- *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.

Cloud libraries support AES-256 symmetric-key encryption.

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. For instructions on creating these files, see "Create a Cloud Library Parameter File" on page 6-13.

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 "Configure Oracle HSM Hosts to Use Disk Volumes" on page 6-11 and 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

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`**).

C
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
```

D

OpenStack Swift on Oracle HSM File Systems

This appendix explains how you configure Oracle HSM shared file systems as the backing store for an OpenStack Swift object storage cloud. Using an Oracle HSM file system that incorporates solid-state devices (SSDs), hard disks, and tape storage can simplify object replication, lower media costs, increase data redundancy, and simplify management. When you build an OpenStack Swift private storage cloud on top of an Oracle HSM archiving file system, the Oracle HSM disk cache serves as the cloud's object store, while the Oracle Hierarchical Storage Manager software supplies critical management functionality that a successful Swift object storage deployment cannot, on its own, deliver. As the OpenStack documentation notes:

You must address a range of storage management-related considerations in the design of a storage-focused OpenStack cloud. These considerations include, but are not limited to, backup strategy (and restore strategy, since a backup that cannot be restored is useless), data valuation-hierarchical storage management, retention strategy, data placement, and workflow automation.(**`https://docs.openstack.org/arch-design/design-storage/design-storage-arch.html`**)

Oracle HSM takes over object replication and automatically manages data integrity, backup, retention, and storage costs according to policies that you define for each file system. It can store objects on an optimized mix of solid-state, high-performance disk, high-capacity disk, and tape media, based on frequency and type of use.

To use an Oracle HSM file system as an OpenStack Swift Object-Storage service, carry out the following high-level tasks:

- Study the relevant OpenStack, Oracle Solaris, and Linux documentation.
- Prepare hosts, storage, and Oracle Hierarchical Storage Manager file systems (if you have not already done so).
- Configure a host as the OpenStack control and proxy node.
- Configure the Oracle HSM client hosts as OpenStack Swift nodes.

Study Relevant OpenStack and Operating System Documentation

This appendix is not a comprehensive guide to OpenStack installation and configuration. It does not address aspects of storage cloud implementation that do not relate directly to the use of Oracle HSM file systems. It cannot keep up with frequent OpenStack updates and changing levels of support for the software in Oracle Solaris and Linux distributions. Nor can it anticipate the requirements that drive your OpenStack cloud implementation. So use this appendix to supplement the documentation provided for your OpenStack release and your releases of Oracle Solaris, and/or Linux.

Prepare Hosts, Storage, and Oracle HSM File Systems

1. Configure at least one Oracle Solaris host as an Oracle HSM file-system metadata server (MDS).

   Only one metadata server can be active at a time. But you can configure one or more additional, potential metadata servers that can be activated in the event that the active server fails.

2. Configure Solaris 11.3+ and Linux machines as Oracle HSM file-system clients.

3. On the metadata server, create at least two high-performance (ma) Oracle HSM file systems for each client host that you plan to configure. Use the procedures outlined in "Configure a High-Performance `ma` File System" on page 6-5.

   In the example, we have created two ma file systems for each Oracle HSM client that will serve as an OpenStack Swift storage node. In each file system, we have placed the metadata (**`mm`**) devices on fast, solid-state devices (SSDs) and the data (**`mr`**) devices on more economical hard disk devices. In a cloud deployment, this arrangement maximizes overall file system performance while minimizing cost. The corresponding master configuration file (**`mcf`**) looks like this:

```
[root@hsm-lx-client1 ~]# cat /etc/opt/SUNWsamfs/mcf
# Oracle HSM file systems that are shared with OpenStack Swift storage nodes
# Equipment          Equipment Equipment Family        Device Additional
# Identifier         Ordinal   Type      Set           State  Parameters
#------------------ --------- --------- ------------- ------ ----------
fs100client1        100        ma       fs100client1  on
/dev/dsk/c0t...d0s0 101        mm       fs100client1  on
/dev/dsk/c0t...d0s0 102        mr       fs100client1  on
/dev/dsk/c0t...d0s0 103        mr       fs100client1  on
/dev/dsk/c0t...d0s0 104        mr       fs100client1  on
/dev/dsk/c0t...d0s0 105        mr       fs100client1  on
fs200client1        200        ma       fs200client1  on
/dev/dsk/c0t...d0s0 201        mm       fs200client1  on
/dev/dsk/c0t...d0s0 202        mr       fs200client1  on
/dev/dsk/c0t...d0s0 203        mr       fs200client1  on
/dev/dsk/c0t...d0s0 204        mr       fs200client1  on
/dev/dsk/c0t...d0s0 205        mr       fs200client1  on
fs300client2        300        ma       fs300client2  on
/dev/dsk/c0t...d0s0 301        mm       fs300client2  on
/dev/dsk/c0t...d0s0 302        mr       fs300client2  on
/dev/dsk/c0t...d0s0 303        mr       fs300client2  on
/dev/dsk/c0t...d0s0 304        mr       fs300client2  on
/dev/dsk/c0t...d0s0 305        mr       fs300client2  on
fs400client2        400        ma       fs400client2  on
/dev/dsk/c0t...d0s0 401        mm       fs400client2  on
/dev/dsk/c0t...d0s0 402        mr       fs400client2  on
/dev/dsk/c0t...d0s0 403        mr       fs400client2  on
/dev/dsk/c0t...d0s0 404        mr       fs400client2  on
/dev/dsk/c0t...d0s0 405        mr       fs400client2  on
fs500client3        500        ma       fs500client3  on
/dev/dsk/c0t...d0s0 501        mm       fs500client3  on
/dev/dsk/c0t...d0s0 502        mr       fs500client3  on
/dev/dsk/c0t...d0s0 503        mr       fs500client3  on
/dev/dsk/c0t...d0s0 504        mr       fs500client3  on
/dev/dsk/c0t...d0s0 505        mr       fs500client3  on
fs600client3        600        ma       fs600client3  on
/dev/dsk/c0t...d0s0 601        mm       fs600client3  on
/dev/dsk/c0t...d0s0 602        mr       fs600client3  on
/dev/dsk/c0t...d0s0 603        mr       fs600client3  on
/dev/dsk/c0t...d0s0 604        mr       fs600client3  on
```

```
/dev/dsk/c0t...d0s0 605        mr       fs600client3   on
```

4. On the metadata server, configure each high-performance file system for archiving, as described in "Configuring Oracle HSM Archiving File Systems" on page 6-11.

5. Share the file systems, as described in "Configuring an Oracle HSM Shared File System" on page 8-7.

   Share the same number of file systems with each client, allowing at least two file systems per client. In the example, three Oracle HSM Linux clients are configured as OpenStack Swift nodes. As the corresponding **/etc/fstab** files show, the metadata server shares file systems **fs100client1** and **fs200client1** with host **hsm-lx-client1**, file systems **fs300client2** and **fs400client2** with host **hsm-lx-client2**, and file systems **fs500client3** and **fs600client3** with host **hsm-lx-client3**:

```
[root@hsm-lx-client1 ~]# cat /etc/fstab
# /etc/fstab on OpenStack Swift storage node hsm-lx-client1
#File         Mount                         Mount
#System       Point                  Type   Options                Dump Fsck
#------------ ---------------------- ------ ---------------------- ---- ----
...
proc          /proc                  proc   defaults               0    0
/dev/sdb1     /srv/node/sdb1         xfs    noatime,...,logbufs=8  0    0
fs100client1  /srv/node/fs100client1 samfs  shared                 0    0
fs200client1  /srv/node/fs200client1 samfs  shared                 0    0

[root@hsm-lx-client2 ~]# cat /etc/fstab
#/etc/fstab on OpenStack Swift storage node hsm-lx-client2
#File         Mount                         Mount
#System       Point                  Type   Options                Dump Fsck
#------------ ---------------------- ------ ---------------------- ---- ----
...
proc          /proc                  proc   defaults               0    0
/dev/sdb1     /srv/node/sdb1         xfs    noatime,...,logbufs=8  0    0
fs300client2  /srv/node/fs300client2 samfs  shared                 0    0
fs400client2  /srv/node/fs400client2 samfs  shared                 0    0

[root@hsm-lx-client3 ~]# cat /etc/fstab
#/etc/fstab on OpenStack Swift storage node hsm-lx-client3]
#File         Mount                         Mount
#System       Point                  Type   Options                Dump Fsck
#------------ ---------------------- ------ ---------------------- ---- ----
...
proc          /proc                  proc   defaults               0    0
/dev/sdb1     /srv/node/sdb1         xfs    noatime,...,logbufs=8  0    0
fs500client3  /srv/node/fs500client3 samfs  shared                 0    0
fs600client3  /srv/node/fs600client3 samfs  shared                 0    0
```

6. Next, configure the OpenStack controller/proxy server.

Configure the OpenStack Controller/Proxy Server Node

An OpenStack controller node handles administration and authorization for the Swift object storage cluster. Proxy-server nodes manage requests from users and applications. In Oracle HSM-based Swift solutions, a single host usually handles both functions.

For instructions on configuring the controller/proxy-server host, consult the documentation for your releases of OpenStack and the operating system.

Now configure the Oracle HSM file-system clients as OpenStack Swift Object Storage Nodes.

Configure Oracle HSM Clients as OpenStack Swift Object Storage Nodes

OpenStack storage nodes provide the storage for the Swift object storage service. The basic steps for configuring storage nodes are listed below.

1. Install the OpenStack Swift storage-node packages for your operating system and software version.

   For instructions, consult the documentation for your releases of OpenStack and the host operating system.

2. Create the OpenStack Swift configuration directory, /etc/swift, and assign owner and group permissions to the swift service.

3. Make the Swift user and group the owner of the directory /srv/node and all of its contents.

   This directory contains the mount points for the shared Oracle HSM file systems.

4. Configure the Swift service and its component object, container, and account services.

5. Next, configure the OpenStack Swift rings.

Configure the OpenStack Swift Rings for Use With Oracle HSM

OpenStack Swift rings are consistent hash tables, data structures that let the Swift object storage service create, distribute, locate, and retrieve replicas of user data and system metadata across multiple devices, hosts, and locations. You create the rings on the controller/proxy-server node and store the ring files on each node of the OpenStack Swift cluster. Thereafter, the Swift service periodically rebalances its rings by redistributing object replicas across the available hard disks in the cluster.

In an Oracle HSM/Swift implementation, you configure the OpenStack rings as you normally would, with one key exception: you disable replication and rebalancing. The Oracle archiving and staging processes create, update, manage, retrieve, and manage files and archived copies, so Swift replication is unnecessary. Swift replication is also entirely disk-based, while Oracle HSM archiving typically uses a variety of media that includes tape and cloud storage as well as disk. Any rebalancing of the rings would thus force the Oracle HSM file system to mount and position multiple tapes, significantly reducing the performance of the file systems.

To configure the OpenStack rings and disable replication and rebalancing, proceed as follows:

1. Log in to the OpenStack proxy server host as **root**.

   ```
   [root@swift-proxy ~]#
   ```

2. Calculate the Swift part-power for the object rings based on the number of Oracle HSM file systems that you have configured.

   The part-power of the object rings is the power of two that best represents the total number of units of storage (partitions) that the solution will require for objects and their replicas. In the examples in this appendix, we have configured two Oracle

HSM file systems on each storage node, for a total of six. Six multiplied by 100 is 600, which falls between 512 (2^9) and 1024 (2^{10}). So we select **`10`** as our part-power.

3. On the proxy server, disable replication and periodic rebalancing when you create the three rings. Use the command **`swift-ring-builder ring.builder create replicas min_part_hours`**, where:

   - **`ring`** is one of the three rings, object, container, or account.
   - **`part-power`** is the object part-power calculated in the previous step.
   - **`replicas`** is 1 (one), the number of copies that are to be made of each object.

     The Oracle HSM archiver will handle object replication by archiving object files to tape. So should not create additional replicas.
   - **`min_part_hours`** is 1 (one) hour.

     The **`min_part_hours`** parameter is the time interval during which the Swift rebalancing process must finish moving one replica before it starts moving another. One hour is small compared to the time required by a move, so a value of 1 effectively disables rebalancing.

```
[root@swift-proxy ~]# swift-ring-builder object.builder create 10 1 1
...
[root@swift-proxy ~]# swift-ring-builder container.builder create 10 1 1
...
[root@swift-proxy ~]# swift-ring-builder account.builder create 10 1 1
```

4. Add each Oracle HSM file system to each of the three rings. Use the command **`swift-ring-builder ring.builder add r1z1-ip:6000Rip:port-number/hsm 100`**, where:

   - **`ring`** is one of the three rings, object, container, or account.
   - **`ip`** is the IP address of the storage node that mounts the file system that you are adding.
   - **`port-number`** is the default OpenStack Swift port number for the ring: **`6000`** for the **`object`** ring, **`6001`** for the **`container`** ring, or **`6002`** for the **`account`** ring.
   - **`R`** specifies the network address for object replication.

     Oracle HSM will handle replication, so we can use the same network, IP address, and port number for storing and replicating objects.
   - **`hsm`** is the name of the directory under **`/srv/node/`** where the Oracle HSM file system.

     Oracle HSM file systems are mounted on the storage node.
   - Set the **`weight`** parameter to **`100`**.

     The Swift weight is a number that determines how many partitions are put on the device relative to the rest of the devices in the cluster.

   In the example, we add **`fs100client1`** and **`fs200client1`** on node **`hsm-lx-client1 (10.80.28.8)`**, **`fs300client2`** and **`fs400client2`** on node **`hsm-lx-client2 (10.80.28.9)`**, and **`fs500client3`** and **`fs600client3`** on node **`hsm-lx-client3 (10.80.28.10)`** to the object ring:

```
[root@swift-proxy ~]# swift-ring-builder object.builder add
r1z1-10.80.28.8:6000R10.80.28.8:6000/fs100client1 100
...
[root@swift-proxy ~]# swift-ring-builder object.builder add
r1z1-10.80.28.8:6000R10.80.28.8:6000/fs200client1 100
```

```
...
[root@swift-proxy ~]# swift-ring-builder object.builder add
r1z1-10.80.28.9:6000R10.80.28.9:6000/fs300client2 100
...
[root@swift-proxy ~]# swift-ring-builder object.builder add
r1z1-10.80.28.9:6000R10.80.28.9:6000/fs400client2 100
...
[root@swift-proxy ~]# swift-ring-builder object.builder add
r1z1-10.80.28.10:6000R10.80.28.10:6000/fs500client3 100
...
[root@swift-proxy ~]# swift-ring-builder object.builder add
r1z1-10.80.28.10:6000R10.80.28.10:6000/fs600client3 100
...
[root@swift-proxy ~]#
```

5. Check the contents of each ring using the command **swift-ring-builder ring.builder**, where **ring** is one of the three rings: **object**, **container**, or **account**.

   The example shows representative commands and output for the **object** ring. Note that our planned configuration has been successfully implemented.

```
[root@swift-proxy]# swift-ring-builder object.builder
object.builder, build version ...
1024 partitions, 1.000000 replicas, 1 regions, 1 zones, 6 devices, 100.00 ...
The minimum number of hours before a partition can be reassigned is 1
The overload factor is 0.00% (0.000000)
Devices:
id region zone  ip address  port replication ... name         weight ...
0  1      1     10.80.28.8  6000 10.80.28.8  ... fs100client1 100.00 ...
1  1      1     10.80.28.8  6000 10.80.28.8  ... fs200client1 100.00 ...
0  1      1     10.80.28.9  6000 10.80.28.9  ... fs300client2 100.00 ...
1  1      1     10.80.28.9  6000 10.80.28.9  ... fs400client2 100.00 ...
0  1      1     10.80.28.10 6000 10.80.28.10 ... fs500client3 100.00 ...
1  1      1     10.80.28.10 6000 10.80.28.10 ... fs600client3 100.00 ...
...
[root@swift-proxy]#
```

6. If you see any problems, repeat the steps above until all rings are correctly configured.

7. Manually rebalance (redistribute) the Swift partitions across the rings, and generate the ring files. Use the command **swift-ring-builder ring.builder rebalance**, where **ring** is one of the three rings: **object**, **container**, or **account**.

```
[root@swift-proxy ~]# swift-ring-builder object.builder rebalance
...
[root@swift-proxy ~]# swift-ring-builder container.builder rebalance
...
[root@swift-proxy ~]# swift-ring-builder account.builder rebalance
...
[root@swift-proxy ~]#
```

8. Copy the files **/etc/swift/account.ring.gz, /etc/swift/container.ring.gz, and /etc/swift/object.ring.gz** from the proxy server to the **/etc/swift/** directory on each storage node.

9. Restart the proxy service.

10. Now start the OpenStack Swift object storage services.

Start the OpenStack Swift Object Storage Services

You can start the OpenStack Swift services as described in the OpenStack documentation—with one key exception: do not start the object auditor service! When the OpenStack Swift object auditor is running, it tries to audit files in the Oracle HSM file system, some of which are typically stored on removable media. Unnecessarily mounting media significantly reduces the performance of the file system and the storage cloud.

To audit cloud storage, use the Oracle HSM Data Integrity Validation (DIV) feature. DIV was designed to avoid tape-related performance issues. "Configure Archival Media Validation" on page 6-56.

Back Up the OpenStack Configuration

To insure that Swift objects remain recoverable in all circumstances, you must back up the configuration files specified in the documentation for your releases of OpenStack and the host operating system. Backing them up to the same Oracle HSM shared file systems that store the object files lets you automatically maintain multiple copies of complete, up-to-date recovery information on multiple kinds of media.

E
Examples

The **/opt/SUNWsamfs/examples/** subdirectory contains sample Oracle HSM configuration files, shell scripts, and **dtrace** programs that illustrate various features and solutions to various requirements. These include the following files:

```
01_example.archiver.cmd.simple.txt
01_example.mcf.simple.txt
01_example.vfstab.txt
02_example.archiver.cmd.disk.tape.txt
02_example.diskvols.conf.NFS.txt
02_example.mcf.shared.txt
02_example.vfstab.disk.archive.NFS.txt
03_example.archiver.cmd.dk.9840.9940.txt
03_example.diskvols.conf
03_example.mcf.dk.9840.9940.txt
03_example.stk.9840C_parms.txt
03_example.stk.9940B_parms.txt
03_example.vfstab.disk.archive.txt
04_example.archiver.cmd.9840.LTO.txt
04_example.mcf.ma.9840.LTO.txt
04_example.stk50c.txt
05_example.archiver.cmd.9840.9940.T10k.txt
05_example.mcf.veritas.9840.9940.T10K.txt
05_example.stk_params9840.txt
05_example.stk_params9940.txt
05_example.stk_paramsT10K.txt
05_example.vstab.txt
06_example.archiver.cmd.samremote.client.txt
06_example.local.copy.samremote.client.stk50.txt
06_example.mcf.samremote.client.txt
06_example.mcf.samremote.server.txt
06_example.samremote.client.setup.stk100.txt
06_example.samremote.client.vfstab.txt
06_example.samremote.configuration.samremote.server.txt
07_example.mcf.distio.client.txt
07_example.mcf.distio.mds.txt
08_example.archiver.cmd.cloud.txt
08_example.cloud_params.txt
08_example.mcf.cloud.txt
08_example.releaser.cmd.txt
08_example.vfstab.txt
09_example.archiver.cmd.dk.cloud.txt
09_example.cloud_params.txt
09_example.diskvols.conf.txt
09_example.mcf.cloud.txt
09_example.vfstab.txt
10_example.archiver.cmd.2cloud.txt
```

```
10_example.cloudx_params.txt
10_example.cloudy_params.txt
10_example.mcf.2cloud.txt
10_example.releaser.cmd.txt
10_example.vfstab.txt
11_example.archiver.cmd.cloud.txt
11_example.cloud_params.txt
11_example.mcf.cloud.txt
11_example.releaser.cmd.txt
11_example.vfstab.txt
12_example.cloud_params_dynamic_kmip.txt
12_example.cloud_params_dynamic_kms.txt
12_example.cloud_params_static_kms.txt
12_example.cloud_params_static_ksf.txt
archive_status.py
archiver.sh
cloud.examples.readme.txt
defaults.conf
dev_down.sh
dtrace
hosts.shsam1
hosts.shsam1.local.client
hosts.shsam1.local.server
inquiry.conf
load_notify.sh
log_rotate.sh
media.c
metadata_config_samfs.xml
migrationd.cmd
nrecycler.sh
preview.cmd
recover.sh
recycler.sh
restore.sh
samdb.conf
samfs.cmd
samst.conf
save_core.sh
sendtrap
ssi.sh
st.conf_changes
stageback.sh
syslog.conf_changes
tarback.sh
verifyd.cmd
```

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.