

**StorageTek Storage Archive Manager and
StorageTek QFS Software**

Installation and Configuration Guide

Release 5.4

E42062-02

January 2015

StorageTek Storage Archive Manager and StorageTek QFS Software Installation and Configuration Guide,
Release 5.4

E42062-02

Copyright © 2011, 2015, Oracle and/or its affiliates. All rights reserved.

Primary Author: Robert Craig Johnson

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface	xi
Documentation Accessibility	xi
Prerequisites for Using this Document	xi
Conventions	xi
Available Documentation	xii
1 Deploying SAM-QFS Solutions	
QFS File Systems	1-2
QFS Defaults and I/O Performance-Tuning Objectives	1-2
Disk Allocation Units and Logical Device Types	1-3
File Allocation Methods	1-3
Striped Allocation	1-3
Round-Robin Allocation	1-4
Storage Allocation and Integrated Volume Management	1-4
File System Types	1-4
General-Purpose ms File Systems	1-4
High-Performance ma File Systems	1-4
SAM-QFS Archiving File Systems	1-5
Archiving	1-6
Staging	1-7
Releasing	1-8
Recycling	1-8
2 Configuring Host Systems	
Configuring Oracle Solaris for SAM-QFS	2-1
Install the Latest Operating System Updates	2-1
Tune Solaris System and Driver Parameters for Anticipated File System I/O	2-1
Configuring Linux for SAM-QFS Clients	2-3
Disable Incompatible Operating System Features	2-3
Install Required Kernel Development and Utility Packages	2-4
3 Configuring Storage Hosts and Devices	
Configuring Primary Storage	3-1
Configure Devices for the Primary Cache	3-1
Configuring Archival Storage	3-2

Zone SAN-attached Devices	3-2
Configuring Archival Disk Storage	3-3
Configuring Archival Tape Storage	3-3
Determine the Order in Which Drives are Installed in the Library	3-4
Gather Drive Information for the Library and the Solaris Host	3-4
Map the Drives in a Direct-Attached Library to Solaris Device Names	3-4
Map the Drives in an ACSLS-Attached Library to Solaris Device Names	3-6
Configure a Direct-Attached Library	3-8
Create a Path-Oriented Alias for the sugen Driver	3-9
Configuring Storage for High-Availability File Systems	3-10

4 Installing SAM-QFS Software

Obtaining Software	4-1
Check Installation Requirements	4-1
Download Software Installation Packages	4-2
Installing Solaris Cluster Software (High-Availability Configurations Only)	4-3
Upgrading Shared SAM-QFS File Systems	4-3
Upgrade Any Significantly Older Releases of SAM-QFS	4-3
Carry Out the Rolling Upgrade	4-4
Installing, Upgrading, or Downgrading SAM-QFS Software on Hosts	4-6
Install, Upgrade, or Downgrade SAM-QFS Software on Oracle Solaris Hosts	4-6
Prepare the Host for Software Changes	4-6
Locate the Packages for Your Host Architecture	4-8
Install the Software Using the Image Packaging System (IPS)	4-9
Upgrade or Downgrade the Software Using the Image Packaging System (IPS)	4-11
Install the Software Using SVR4 <code>pkgm</code> and <code>pkgadd</code> Commands	4-13
Upgrade or Downgrade the Software Using SVR4 <code>pkgm</code> and <code>pkgadd</code> Commands	4-13
Install or Update SAM-QFS Client Software on Linux Hosts	4-14
Uninstalling SAM-QFS Software	4-16
Uninstall SAM-QFS on a Solaris Host	4-16
Uninstall the SAM-QFS Client on a Linux Host	4-17

5 Using the `samsetup` Configuration Wizard

6 Configuring the Basic File System

Configuring QFS File Systems	6-1
Prepare Disk Storage for a QFS File System	6-1
Configure a General-Purpose ms File System	6-2
Configure a High-Performance ma File System	6-7
Configuring SAM-QFS Archiving File Systems	6-11
Prepare Disk Storage for Archival Copies	6-11
Prepare Removable Media Libraries and Drives	6-13
Configure an Oracle StorageTek ACSLS Network-Attached Automated Library	6-13
Configure Labeling Behavior for Barcoded Removable Media	6-15
Set Drive Timing Values	6-17
Configure the Archiving File System	6-19

Mount the Archiving File System	6-22
Configure the Archiving Process	6-24
Configuring the Recycling Process	6-33
Configure Recycling by Archive Set	6-33
Configure Recycling by Library	6-35
Catalog Archival Media Stored in a Network-Attached Tape Library	6-36
Configure File System Protection	6-38
Create Locations for Storing Recovery Point Files and Copies of the Archiver Log	6-39
Automatically Create Recovery Points and Save Archiver Logs	6-40
Configure Archival Media Validation	6-44
Configure SAM-QFS to Support Data Integrity Validation	6-44
Configure SAM-QFS Periodic Media Verification	6-46
Enabling Support for Write Once Read Many (WORM) Files	6-51
Enable the WORM Support on a SAM-QFS File System	6-52
Enabling Support for the Linear Tape File System (LTFS)	6-54
Beyond the Basics	6-55

7 Accessing File Systems from Multiple Hosts

Accessing File Systems from Multiple Hosts Using SAM-QFS Software	7-1
Configuring a SAM-QFS Single-Writer, Multiple-Reader File System	7-2
Create the File System on the Writer	7-2
Configure the Readers	7-4
Configuring a SAM-QFS Shared File System	7-7
Configuring a File System Metadata Server for Sharing	7-7
Create a Hosts File on the Active Metadata Server	7-7
Create the Shared File System on the Active Server	7-9
Mount the Shared File System on the Active Server	7-10
Configuring File System Clients for Sharing	7-12
Create the Shared File System on the Solaris Clients	7-12
Mount the Shared File System on the Solaris Clients	7-15
Create the Shared File System on the Linux Clients	7-16
Mount the Shared File System on the Linux Clients	7-18
Use Local Hosts Files to Route Network Communications	7-19
Configuring Archival Storage for a Shared File System	7-21
Connect Tape Drives to Server and Datamover Hosts Using Persistent Bindings	7-21
Configure the Hosts of an Archiving File System to Use the Archival Storage	7-24
Distribute Tape I/O Across the Hosts of the Shared Archiving File System	7-26
Accessing File Systems from Multiple Hosts Using NFS and SMB/CIFS	7-29
Sharing SAM-QFS File Systems Using NFS	7-30
Disable Delegation Before Using NFS 4 to Share a SAM-QFS Shared File System	7-30
Configure NFS Servers and Clients to Share SAM-QFS WORM Files and Directories	7-31
Configure the NFS Server on the SAM-QFS Host	7-31
Share the SAM-QFS File System as an NFS Share	7-34
Mount the NFS-Shared SAM-QFS File System on the NFS Clients	7-34
Sharing SAM-QFS File Systems Using SMB/CIFS	7-37
Review Oracle Solaris SMB Configuration and Administration Documentation	7-38
Explicitly Map Windows Identities for the SMB Server (Optional)	7-38

Configure SAM-QFS File Systems for Sharing With SMB/CIFS.....	7-38
Convert a SAM-QFS Unshared File System that Uses POSIX-Style ACLs	7-38
Convert a SAM-QFS File Shared System that Uses POSIX-Style ACLs	7-39
Configure the SMB Server for Windows Active Directory Domains or Workgroups ..	7-41
Configure the SMB Server in Domain Mode	7-41
Configure the SMB Server in Workgroup Mode	7-42
Share the SAM-QFS File System as an SMB/CIFS Share.....	7-43

8 Configuring SAM-Remote

Make Sure that All SAM-Remote Hosts Use the Same Software	8-2
Stop SAM-QFS Processes	8-2
Configure the SAM-Remote Server	8-3
Define the Remotely Shared Archiving Equipment in the SAM-Remote Server's mcf File	8-3
Create the samremote Server Configuration File	8-5
Configure the SAM-Remote Clients	8-7
Define the Remote Archiving Equipment in the SAM-Remote Client's MCF File.....	8-7
Create the SAM-Remote Client Configuration File.....	8-9
Configure the archiver.cmd file on the SAM-Remote Client	8-10
Validate the Archiving Configuration on the SAM-Remote Server	8-11
Validate the Archiving Configuration on Each SAM-Remote Client	8-13
Configuring Recycling for SAM-Remote	8-14
Configure Recycling on the SAM-Remote Server	8-15
Configure Recycling on the SAM-Remote Client.....	8-17

9 Preparing High-Availability Solutions

Understanding the Supported SAM-QFS High-Availability Configurations	9-1
HA-QFS, a High-Availability QFS Unshared, Standalone File-System Configuration.....	9-1
HA-COTC, a QFS Shared File System with High-Availability Metadata Servers	9-2
HA-SAM, a High-Availability, Archiving, QFS Shared File-System Configuration	9-2
SC-RAC, a High-Availability QFS Shared File-System Configuration for Oracle RAC.....	9-2
High-Availability QFS Unshared File Systems	9-3
Create Unshared QFS File Systems on Both Cluster Nodes	9-3
Configure the High-Availability QFS File System	9-4
High-Availability QFS Shared File Systems, Clients Outside the Cluster	9-5
Create a QFS Shared File System Hosts File on Both HA-COTC Cluster Nodes	9-6
Create Local Hosts Files on the QFS Servers and Clients Outside the HA-COTC Cluster	9-9
Configure an Active QFS Metadata Server on the Primary HA-COTC Cluster Node	9-10
Create a High-Performance QFS File System on the Primary HA-COTC Node.....	9-10
Exclude Data Devices from Cluster Control.....	9-12
Disable Fencing for QFS Data Devices in the HA-COTC Cluster	9-12
Place Shared Data Devices in a Local-Only Device Group on the HA-COTC Cluster	9-12
Mount the QFS File System on the Primary HA-COTC Node.....	9-13
Configure a Potential QFS Metadata Server on the Secondary HA-COTC Cluster Node ..	9-14
Create a High-Performance QFS File System on the Secondary HA-COTC Node.....	9-14
Mount the QFS File System on the Secondary HA-COTC Node.....	9-15
Configure Failover of the HA-COTC Metadata Servers	9-15

Configure Hosts Outside the HA-COTC Cluster as QFS Shared File System Clients.....	9-17
High-Availability SAM-QFS Shared Archiving File Systems	9-20
Create a SAM-QFS Shared File System Hosts File on Both HA-SAM Cluster Nodes	9-21
Create Local Hosts Files on the HA-SAM Servers	9-23
Configure an Active QFS Metadata Server on the Primary HA-SAM Cluster Node	9-25
Configure a Potential QFS Metadata Server on the Secondary HA-SAM Cluster Node	9-27
Configure Failover of HA-SAM Local Storage	9-28
Configure Failover of the HA-SAM Metadata Servers.....	9-31
High-Availability QFS Shared File Systems and Oracle RAC	9-33
Create a QFS Shared File System Hosts File on All SC-RAC Cluster Nodes	9-34
Configure an Active QFS Metadata Server on the Primary SC-RAC Cluster Node	9-37
Configure a Potential QFS Metadata Server on the Remaining SC-RAC Cluster Nodes	9-40
Configure Failover of the SC-RAC Metadata Servers	9-41
Configure QFS Metadata Servers on SC-RAC Nodes Using Software RAID Storage.....	9-44
Install Solaris Volume Manager on Solaris 11	9-44
Create Solaris Volume Manager Multi-Owner Disk Groups	9-46
Create Mirrored Volumes for the QFS Data and Metadata	9-47
Create a QFS Shared File System on the SC-RAC Cluster Using Mirrored Volumes ...	9-49
10 Configuring the SAM-QFS Reporting Database	
Install and Configure the MySQL Server Software	10-1
Create a Database Load File	10-2
Create the Sideband Database	10-3
Mount the SAM-QFS File System with Database Support Enabled	10-5
11 Configuring Notifications and Logging	
Configuring Simple Network Management Protocol (SNMP)	11-1
Make Sure that All SNMP Management Stations are Listed in the <code>/etc/hosts</code> file.....	11-2
Enable Support for SNMP	11-2
Designate Management Stations as Trap Recipients and Configure Authentication.....	11-3
Disable Support for SNMP	11-4
Enabling SAM-QFS Application Logging	11-5
Enable SAM-QFS Logging	11-5
Configuring Device Logging	11-6
Enable the Device Log in the <code>defaults.conf</code> File.....	11-6
Configuring Log Rotation	11-7
Set Up Automatic Rotation for SAM-QFS Log Files	11-7
Enabling Email Alerts	11-10
12 Tuning I/O Characteristics for Special Needs	
Optimize Paged I/O for Larger Data Transfers	12-2
Enable Switching Between Paged and Direct I/O	12-4
Configure the File System to Use Direct I/O Exclusively	12-7
Increase the Directory Name Lookup Cache Size.....	12-8

13 Backing Up the SAM-QFS Configuration

Create a Backup Location for Your SAM-QFS Configuration	13-1
Run <code>samexplorer</code> and Safely Store the Report	13-1
Manually Backup the SAM-QFS Configuration.....	13-2

A Glossary of Equipment Types

Recommended Equipment and Media Types	A-1
Other Equipment and Media Types	A-2

B Mount Options in a Shared File System

Shared File System Mount Options	B-1
bg : Mounting in the Background	B-1
retry : Reattempting a File System Mount	B-1
shared : Declaring a SAM-QFS Shared File System.....	B-1
minallopsz and maxallopsz : Tuning Allocation Sizes	B-1
rdlease , wrlease , and aplease : Using Leases in a SAM-QFS Shared File System	B-2
mh_write : Enabling Multiple Host Reads and Writes	B-2
min_pool : Setting the Minimum Number of Concurrent Threads	B-3
meta_timeo : Retaining Cached Attributes.....	B-3
stripe : Specifying Striped Allocation	B-3
sync_meta : Specifying the Frequency With Which Metadata Is Written.....	B-3
worm_capable and def_retention : Enabling WORM Functionality	B-4

C Configuration Directives for Archiving

Archiving Directives.....	C-1
Global Archiving Directives	C-1
archivemeta : Controlling Whether Metadata Is Archived.....	C-2
archmax : Controlling the Size of Archive Files	C-2
bufsize : Setting the Archiver Buffer Size	C-2
drives : Controlling the Number of Drives Used for Archiving.....	C-3
examine : Controlling Archive Scans.....	C-3
interval : Specifying an Archive Interval	C-4
logfile : Specifying an Archiver Log File	C-4
notify : Renaming the Event Notification Script.....	C-4
ovflmin : Controlling Volume Overflow.....	C-5
scanlist_squash : Controlling Scanlist Consolidation.....	C-5
setarchdone : Controlling the Setting of the archdone Flag	C-5
wait : Delaying Archiver Startup	C-6
File System Directives.....	C-6
fs : Specifying a File System	C-6
copy-number [archive-age]: Specifying Multiple Copies of File System Metadata	C-6
interval , logfile , scanlist as File System Directives	C-7
archive-set-name : the Archive Set Assignment Directive	C-7
Archive Copy Directives.....	C-8
Copy Parameters	C-9
Volume Serial Number (VSN) Pools Directives	C-13

Volume Serial Number (VSN) Association Directives	C-13
Staging Directives	C-14
The stager.cmd File	C-14
drives : Specifying the Number of Drives for Staging	C-15
bufsize : Setting the Stage Buffer Size	C-15
logfile : Specifying a Staging Log File	C-15
maxactive : Specifying the Number of Stage Requests	C-17
copysel : Specifying the Copy Selection Order During Staging.....	C-17
Preview Request Directives	C-17
Global Directives	C-17
vsn_priority : Adjusting Volume Priorities	C-18
age_priority : Adjusting Priorities for Time Spent Waiting in the Queue	C-18
Global and/or File System-Specific Directives.....	C-18
hwm_priority : Adjusting Priorities When the Disk Cache is Nearly Full.....	C-18
lwm_priority : Adjusting Priorities When the Disk Cache is Nearly Empty	C-19
lhwm_priority : Adjusting Priorities as the Disk Cache Fills	C-19
hlwm_priority : Adjusting Priorities as the Disk Cache Empties	C-19
Sample preview.cmd File	C-19

D Examples

E Product Accessibility Features

Glossary

Preface

This document addresses the needs of system administrators, storage and network administrators, and service engineers who are tasked with installing and configuring SAM-QFS file systems and archiving solutions.

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 SAM-QFS Solaris operating system, storage, and network administration. Please refer to the Solaris documentation and man pages and to storage hardware documentation for information on relevant tasks, commands, and procedures.

Conventions

The following textual conventions are used in this document:

- *Italic* type represents book titles and emphasis.
- Monospace type represents commands and text displayed in a terminal window and the contents of configuration files, shell scripts, and source code files.
- **Monospace bold** type represents user inputs and significant changes to commandline output, terminal displays, or file contents. It may also be used to emphasize particularly relevant parts of a file or display.
- **Monospace bold oblique** type represents variable inputs and outputs in a terminal display or file.
- *Monospace oblique* type represents other variables in a terminal display or file.
- ... (three-dot ellipsis marks) represent file contents or command output that is not relevant to the example and has thus been omitted for brevity or clarity.

- \ (a backslash) at the end of a line in examples escapes the line break so that the following line is part of the same command.
- [-] (brackets surrounding values separated by a hyphen) delimit value ranges.
- [] (brackets) in command syntax descriptions indicate optional parameters.
- root@solaris:~# and [hostname]:root@solaris:~# represent Solaris command shell prompts.
- [root@linux ~]# represents Linux command shell prompts.

Available Documentation

The *StorageTek Storage Archive Manager and StorageTek QFS Software Installation and Configuration Guide* is part of the multivolume *SAM-QFS Customer Documentation Library*, available from docs.oracle.com.

For Solaris operating system documentation, see the *Oracle Solaris 11.1 Information Library*, also available from docs.oracle.com.

Deploying SAM-QFS Solutions

Deploying Oracle StorageTek Storage Archive Manager and StorageTek QFS Software (SAM-QFS) is basically a fairly 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, SAM-QFS offers a wide range of options and tuning parameters. These extra features can let you address almost any special need. But unneeded features also complicate deployment and make the resulting solution less than satisfactory.

Accordingly, this document is designed to guide you through a deployment of a SAM-QFS deployment that closely follows detailed solution requirements. It starts from the workings, installation, and configuration of basic QFS and SAM-QFS file systems that 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 you can then branch to procedures for configuring additional features that support particular environments and specialized business needs. You will

- configure hardware and operating system software to meet requirements
- configure the basic QFS and/or SAM-QFS file systems required, accepting defaults wherever possible
- configure any additional SAM-QFS 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 Storage Archive Manager 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, SAM-QFS archiving system implemented on a mix of disk arrays and tape libraries as if it were an ordinary UFS file system on a single, local disk. Once installed and configured, SAM-QFS software should automatically manage your data and storage resources in the most efficient and reliable way possible, with minimal human intervention. Over-complex implementations and overzealous micromanagement of file systems and storage resources thus undermine key goals of a SAM-QFS deployment, while often impairing performance, capacity utilization, and/or data protection.

The remainder of this introduction provides brief, descriptive overviews of [QFS File Systems](#) and [SAM-QFS 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 that allows a high degree of customization while remaining fully compatible with standard interfaces. QFS software organizes physical storage devices into custom logical devices whose characteristics can be tuned to meet almost any set of requirements. But 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-driver interfaces. This combination of standards compliance and tunability distinguishes QFS from other UNIX file systems.

The remainder of this section starts with a brief discussion of [QFS Defaults and I/O Performance-Tuning Objectives](#), 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 System Types](#) 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 the number of *seeks* (movements of the read/write heads on the disk devices) required to service 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.

SAM-QFS 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 using adjustable [Disk Allocation Units and Logical Device Types](#). You can optimize the way in which files are stored on devices by selecting from a range of [File Allocation Methods](#). You can select from [File System Types](#) 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 determine the average size of the data files that you need to access and store and then specify the DAU that provides the best match.

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]).

File systems that 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-specified 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.

For file systems that 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 sizes of the files, 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 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 to each 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 (disk head movement) 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 SAM-QFS Software](#)" on page 7-1 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 standard UNIX file system.

File System Types

There are two types of QFS file-system:

- [General-Purpose `ms` File Systems](#) are the simplest to implement and are well suited for most common purposes.
- [High-Performance `ma` File Systems](#) minimize I/O contention by segregating data and metadata I/O on dedicated devices, thus improving data transfer rates in demanding applications.

General-Purpose `ms` File Systems

The general-purpose `ms` file system stores file-system metadata with file data on the same devices and uses dual-allocation `md` disk devices only. This approach allows simplified hardware configuration that meets most needs.

High-Performance `ma` File Systems

High-performance `ma` file systems store metadata and data separately, on dedicated devices. This approach maximizes data input/output (I/O). Metadata updates do not contend with user and application I/O, and device configurations can be better

optimized. You can, for example, place the metadata on RAID-10 mirrored disks for high redundancy and fast reads while the data resides on a more space-efficient RAID-5 disk array.

An `ma` file system can store data on `md` disk devices, `mr` disk devices, or striped groups. But metadata is stored on dedicated `mm` devices. So each file system must contain at least one `mm` device and one `md`, `mr`, or striped-group data device (`md`, `mr`, and striped-group devices cannot be mixed within a single `ma` file system).

SAM-QFS Archiving File Systems

Archiving file systems combine one or more QFS `ma`- or `ms`-type file systems with archival storage and Oracle StorageTek Storage Archive Manager (SAM) software. The SAM software copies files from the file system's disk cache to secondary disk storage and/or removable media and manages the copies as an integral part of the file system. So SAM-QFS file systems offer 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 media.

A properly configured, SAM-QFS file system provides continuous data protection without separate backup applications. User-defined policies copy file data automatically, as the files are created or changed. Up to four copies can be maintained on a mix of disk and tape media, using both local and remote resources. The file system metadata records the location of the file and all of the copies. The software provides a range of tools for quickly locating copies. So lost or damaged files can be readily recovered from the archive. Yet backup copies are kept in standard, POSIX-compliant `tar` (tape archive) format that also let you recover data even if the SAM-QFS software is not available. SAM-QFS keeps file-system metadata consistent at all times by dynamically detecting and recovering from I/O errors. So you can bring a file system back up without time-consuming, file-system integrity checks—a major consideration when hundreds of thousands of files and petabytes of data are being stored. If the file-system metadata is stored on separate devices and only data-storage disks are involved, recovery is complete when replacement disks are configured into the file system. When users request files that resided on a failed disk, SAM-QFS automatically stages backup copies from tape to the replacement disk. If metadata is lost as well, an administrator can restore it from a `samfsdump` backup file using the `samfsrestore` command. Once the metadata has been restored, files can again be restored from tape as users request them. 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 optical media makes SAM-QFS 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 high-density 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, SAM-QFS file systems can be

WORM-enabled, with support for 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.

There are four, basic, SAM-QFS processes that 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 a SAM-QFS 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 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 SAM-QFS, 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.

Archive log files a continuous record of archival action. You can use the log file to locate earlier copies of files for traditional backup purposes. The `sam-arfind` and `sam-arcopy` processes use the `syslog` facility and `archiver.sh` to log warnings and informational messages in a log file that contains information about each archived or automatically unarchived file.

The log file is disabled by default. Use the `logfile=` directive in the `archiver.cmd` file to enable logging and to specify the name of the log 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 the entire file is staged, 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 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 them on a per-directory or per-file basis from the command line (see the `stage` and `stager.cmd` man pages for details). 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 could specify *associative* staging to stage a group of files that are used together whenever any one of them is staged.

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 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.

The 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 (see the `release` and `releaser.cmd` man pages for details). 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.

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 volume to other volumes, and then calls an editable script, `/etc/opt/SUNWsamfs/scripts/recycler.sh`, that relabels the volume, exports it from the library, or carries out some other, user-defined action.

By default, recycling process does not run automatically. But you can configure the Solaris `crontab` file it to run a convenient time or you can call it as needed from the command line using the command `/opt/SUNWsamfs/sbin/sam-recycler`. To modify default recycling parameters, you add recycling parameters to the

`/etc/opt/SUNWsamfs/archiver.cmd` or create a separate
`/etc/opt/SUNWsamfs/recycler.cmd` file. See the corresponding man pages for details.

Configuring Host Systems

Configure host operating systems for SAM-QFS before proceeding further with installation and configuration. The chapter outlines the following topics:

- [Configuring Oracle Solaris for SAM-QFS](#)
- [Configuring Linux for SAM-QFS Clients](#)

Configuring Oracle Solaris for SAM-QFS

Carry out the following tasks:

- [Install the Latest Operating System Updates](#)
- [Tune Solaris System and Driver Parameters for Anticipated File System I/O](#)

Install the Latest Operating System Updates

If possible, always install the latest patches and updates for the operating system. If you need to use the latest features available in SAM-QFS Release 5.4, you must have Oracle Solaris 11 operating system software installed on all SAM-QFS Solaris hosts.

For the minimum recommended operating system releases for use with SAM-QFS software, consult the release notes and My Oracle Support (support.oracle.com).

For Solaris operating system installation and update instructions, consult the installation and administration documents in the *Customer Documentation Library* for the chosen version of Solaris and the knowledge articles on the Oracle Technical Network (OTN) and support.oracle.com. If you are new to the Image Packaging System (IPS), the following 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 compatibly sized 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 SAM-QFS 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 = 0xvalue`, 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=0xvalue`, 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=0xvalue`, 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 are preparing a SAM-QFS solution that includes shared file systems with one or more Linux clients, go to [Configuring Linux for SAM-QFS Clients](#).
10. Otherwise, go to "[Configuring Storage Hosts and Devices](#)" on page 3-1.

Configuring Linux for SAM-QFS Clients

Before you install the SAM-QFS client software, you must the prepare the Linux operating system as follows:

- [Disable Incompatible Operating System Features](#)
- [Install Required Kernel Development and Utility Packages](#)

Disable Incompatible Operating System Features

1. Log in to the SAM-QFS 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.

SAM-QFS 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. SAM-QFS does not support AppArmor.

4. Next, [Install Required Kernel Development and Utility Packages](#).

Install Required Kernel Development and Utility Packages

Prior to installation of the SAM-QFS 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 SAM-QFS 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.e16.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.e16.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.e16.x86_64
libgcc-4.4.7-3.e16.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.e16.x86_64
libmake-3.81.20.e16.x86_64
[root@linux ~]#

```

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

If the SAM-QFS 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.e16.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.e16.x86_64
[root@linux ~]#
```

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

If the SAM-QFS 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.e16.x86_64
rpm-libs-4.8.0-27.e16.x86_64
rpm-python-4.8.0-27.e16.x86_64
[root@linux ~]#
```

12. Next, go to ["Configuring Storage Hosts and Devices"](#) on page 3-1.

Configuring Storage Hosts and Devices

Carry out the storage configuration tasks outlined in this chapter before proceeding further with SAM-QFS installation and configuration. The chapter outlines the following topics:

- [Configuring Primary Storage](#)
- [Configuring Archival Storage](#)
- [Configuring Storage for High-Availability File Systems](#)

Configuring Primary Storage

In a SAM-QFS 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. Determine the total capacity required to hold the data that you plan to store in each file system.
2. Allow an additional 10% of the storage capacity allocated for the file system to store file-system metadata.
3. If you are preparing for a high-performance ma-type file system, configure one, hardware-controlled, four-disk, RAID 10 (1+0) volume group for each mm metadata device in the file-system configuration. Consider using solid-state disks for maximum performance.

The characteristics of striped-mirror, RAID 10 arrays are ideal for storing SAM-QFS metadata. 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 following tasks:

- [Zone SAN-attached Devices](#)
- [Configuring Archival Disk Storage](#)
- [Configuring Archival Tape Storage](#)

Zone SAN-attached Devices

1. Zone the storage area network (SAN) to allow communication between the drive and the host bus adapter.
2. 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
```

3. 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.
4. 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
```

5. 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:
```

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

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

10. Next, go to [Configuring Archival Disk Storage](#).

Configuring Archival Disk Storage

You can use ZFS, UFS, QFS, or NFS 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 contention between archiving and staging jobs and between SAM-QFS and other applications. Observe the following guidelines:

1. Use dedicated file systems, so that SAM-QFS does not contend with other applications and users for access to the file system.
2. Configure one SAM-QFS archival disk volume per file system or ZFS data set and set a quota for the amount of storage space that the archival disk volume can occupy.

When the storage space for an archive volume is dynamically allocated from a pool of shared disk devices, make sure that the underlying physical storage is not oversubscribed. Quotas help to keep SAM-QFS archiving processes from trying to use more of the aggregate storage than it has available.

3. Size each file system at between 10 and 20 terabytes, if possible.
4. When the available disk resources allow, configure multiple file systems, so that individual SAM-QFS archiving and staging jobs do not contend with each other for access to the file system. Between fifteen and thirty archival file systems are optimum.
5. Configure each file system on dedicated devices, so that individual archiving and staging jobs do not contend with each other for access to the same underlying hardware.

Do not use the subdirectories of a single file system as separate archival volumes.

Do not configure two or more file systems on LUNs that reside on the same physical drive or RAID group.

6. Now go to [Configuring Archival Tape Storage](#).

Configuring Archival Tape Storage

Carry out the following tasks:

- [Determine the Order in Which Drives are Installed in the Library](#)
- [Configure a Direct-Attached Library.](#)

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 `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 SAM-QFS metadata server and datamover host, determine the drive order by carrying out the tasks listed below:

- [Gather Drive Information for the Library and the Solaris Host](#)
- Either [Map the Drives in a Direct-Attached Library to Solaris Device Names](#) or [Map the Drives in an ACSLS-Attached Library to Solaris Device Names](#), depending on the equipment you are using.

Gather Drive Information for the Library and the Solaris Host

1. Consult the library documentation. Note how drives and targets are identified. If there is a local operator panel, see how it can be used to determine drive order.
2. If the library has a local operator panel mounted on the library, use it to determine the order in which drives attach to the controller. Determine the SCSI target identifier or World Wide Name of each drive.
3. Log in to the Solaris host as `root`.

```
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 Drives in a Direct-Attached Library to Solaris Device Names](#) or [Map the Drives in an ACSLS-Attached Library to Solaris Device Names](#).

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 SAM-QFS server host, carry out the following procedure:

1. If you are not already logged in to the SAM-QFS 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 3-4, 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. If you are mapping the drives in a tape library, identify the Solaris `/dev/rmt/` entry that corresponds to the drive that mounts the tape. Until you identify the drive, run the command `mt -f /dev/rmt/number status` where `number` identifies the drive in `/dev/rmt/`.

In the example, the drive at `/dev/rmt/0` is empty, but the drive at `/dev/rmt/1` holds the tape. So the drive that the library identifies as drive 1 corresponds to Solaris `/dev/rmt/1`:

```

root@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. Open the device-mappings file that you created in the previous procedure file in a text editor.

In the example, we use the vi editor:

```

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:

```

7. 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

```

8. Unload the tape.
9. Repeat this procedure until the device-mappings file holds entries that map all devices that the library assigns to the SAM-QFS host to Solaris logical device names. 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:~#

```

10. Keep the mappings file. You will need the information for [Configuring the Basic File System \(Chapter 6\)](#), and you may wish to include it when [Backing Up the SAM-QFS Configuration \(Chapter 13\)](#).
11. Next, go to ["Configure a Direct-Attached Library"](#) on page 3-8.

Map the Drives in an ACSLS-Attached Library to Solaris Device Names

1. If you are not already logged in to the SAM-QFS 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 3-4, 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 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 Basic File System \(Chapter 6\)](#), and you may wish to include it when [Backing Up the SAM-QFS 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 3-10. Otherwise, go to ["Installing SAM-QFS Software"](#) on page 4-1.

Configure a Direct-Attached Library

By default, Solaris 10 update 6 and later versions of the operating system control robotic media libraries using the generic SCSI driver `sgen`. So, SAM-QFS Release 5.4 and later uses the default `sgen` driver in place of the legacy SAM-QFS `samst` driver.

1. Physically connect the library and drives to the SAM-QFS server host.
2. If you are installing SAM-QFS for the first time or upgrading a SAM-QFS configuration on Solaris 11, stop once the hardware has been physically connected.

The installation software will use the `sgen` driver automatically and update driver aliases and any existing `/etc/opt/SUNWsamfs/mcf` as necessary.

3. If you are installing SAM-QFS on a Solaris 10 system, log in to the server host as `root`, and find out which version of Solaris is installed. Use the commands `uname -a`.

```
root@solaris:~# cat /etc/release
                Oracle Solaris 10 9/10 s10s_u9wos_14a SPARC
                Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
                Assembled 11 August 2010

root@solaris:~#
```

4. See if one of the driver aliases in the list below is assigned to the `sgen` driver in your version of Solaris. Use the command `grep scs.*,08 /etc/driver_aliases`.

Depending on the version of Solaris 10, 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:~#
```

5. If the `grep` command returns `sgen "alias"`, where `alias` is an alias in the list above, stop here.

The `sgen` driver is installed and assigned to the alias.

6. 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](#).

- 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
```

- 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
```

- If the command `grep scs.*,08 /etc/driver_aliases` shows that the alias `scsiclass,08` is assigned to the `sgen` driver, stop here. The driver is properly configured.

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

The library has now been configured using the `sgen` driver.

- If you are configuring a high-availability file system, see [Configuring Storage for High-Availability File Systems](#).
- Otherwise, go to "[Installing SAM-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:

- Log in to the SAM-QFS server host as `root`.

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

- 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:~#
```

- 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). The command is a single line, but has been formatted as two to fit the page layout:

```
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, go to [Configuring Storage for High-Availability File Systems](#).
7. Otherwise, go to ["Installing SAM-QFS Software"](#) on page 4-1.

Configuring Storage for High-Availability File Systems

For optimal file system performance, the metadata and file data should be accessible through multiple interconnects and multiple disk controllers. In addition, plan to write file data to separate, redundant, highly available disk devices.

Plan to write your file system's metadata to RAID-10 disks. You can write file data to either RAID-10 or RAID-5 disks.

If you want to configure a QFS shared file system on a cluster, you must provide highly available, redundant data paths and storage. To insure redundant data paths, you must provide multiple host bus adapters (HBAs) that are configured from a single node, and you must configure Oracle Solaris I/O multipathing software (for more information, see the *Oracle Solaris SAN Configuration and Multipathing Guide* in the *Oracle Solaris 11.1 Information Library*, or see the `stmsboot` man page). You can provide redundant storage using either hardware or software RAID technology. You can configure hardware-controlled RAID arrays with RAID-10 mirrors and/or RAID 5 volume groups. You configure software-controlled RAID-1 mirrors by using the multi-owner diskset feature of Oracle Solaris Cluster. No other software volume management configurations are supported. See "[Configure QFS Metadata Servers on SC-RAC Nodes Using Software RAID Storage](#)" on page 9-44 for details.

To determine redundancy, consult the hardware documentation for your disk controllers and disk devices. You need to know whether the disk controller or disk devices that are reported by the `cldevice show` command are on redundant storage. For information, see the storage controller vendor's documentation set and view the current controller configuration.

Installing SAM-QFS Software

Starting with Release 5.4, SAM-QFS has adopted 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, and rolled back if necessary. Installations and updates can thus be applied safely to running, production systems.

To install the SAM-QFS software, carry out the following tasks:

- [Obtaining Software](#)
- [Installing Solaris Cluster Software \(High-Availability Configurations Only\)](#)
- [Upgrading Shared SAM-QFS File Systems](#) (if applicable)
- [Installing, Upgrading, or Downgrading SAM-QFS Software on Hosts](#)

The chapter concludes with brief notes on [Uninstalling SAM-QFS Software](#).

Obtaining Software

This section outlines the process for obtaining required installation software and software updates. See the following sections:

- [Check Installation Requirements](#)
- [Download Software Installation Packages](#)

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 SAM-QFS release notes, My Oracle Support at support.oracle.com, and the SAM-QFS wiki pages at wikis.oracle.com/display/SAMQFS/Home.

Download Software Installation Packages

Download the installation packages for Oracle software products from the Oracle Software Delivery Cloud. The basic procedure is similar for all Oracle products.

To download SAM-QFS Release 5.4 packages, proceed as follows:

1. Open `edelivery.oracle.com` in a web browser window.
2. Register, if you have not already used the site.
3. Sign in using your registration credentials.
4. Check the checkbox to acknowledge the applicable software license.
5. Check the checkbox to agree to the export restrictions that apply to the software.
6. On the Media Pack Search page, click the License List link.
7. When the License List appears, search for the product that you have licensed.
StorageTek Storage Archive Manager and StorageTek QFS Software are listed separately. But you do not need to select both.
8. From the Platform list, select the operating system and platform architecture that will host the SAM-QFS software.
9. Press the Go button.
10. When the results list appears, click the radio button that corresponds to the StorageTek Storage Archive Manager and StorageTek QFS Software media pack and press Continue.
11. When the StorageTek Storage Archive Manager and StorageTek QFS Software Media Pack for Oracle Solaris page appears, press the Readme button and read the download instructions.
12. While still on the StorageTek Storage Archive Manager and StorageTek QFS Software Media Pack for Oracle Solaris page, press the View Digest button and save the digest values.

Digests are checksums created by a cryptographic hash function. By comparing the published digest with the digest locally computed from a downloaded file, you can insure that the downloaded file is complete and intact. For instructions on calculating checksums from a file, see the Solaris `dgst` and `md5` man pages.
13. While still on the StorageTek Storage Archive Manager and StorageTek QFS Software Media Pack for Oracle Solaris page, press the Download button that corresponds to the product that you have licensed.

There are separate buttons for StorageTek Storage Archive Manager and StorageTek QFS Software. The former includes both the archiving software and the file system. The latter contains only the file system.
14. When prompted, save the ZIP archive to a local directory, as described on the Readme page.

The chosen directory should be accessible from all SAM-QFS hosts via the local network. For the examples in this chapter, we download the file to the `/samqfs` directory on host `sw_install`.
15. If you cannot download the required files after several tries, please contact Software Delivery Customer Service at `edelivery_ww@oracle.com` for assistance.
16. Once you have downloaded the ZIP file, unzip it in the local directory.

In the example, we unzip the file `STK_SAM-QFS_5.4.zip` in the `/samqfs` subdirectory and then list the contents:

```
[sw_install]root@solaris:~# cd /samqfs
[sw_install]root@solaris:~# unzip STK_SAM-QFS_5.4.zip
[sw_install]root@solaris:~# ls -l
./
../
README.txt
STK_SAM-QFS_5.4/
STK_SAM-QFS_5.4.zip
STK_SAM-QFS_5.4_linux.iso
iso.md5
```

17. If you are preparing a high-availability file-system, got to "[Installing Solaris Cluster Software \(High-Availability Configurations Only\)](#)" below.
18. If you are preparing a multi-host, shared file system, go to "[Upgrading Shared SAM-QFS File Systems](#)" below.
19. Otherwise, go directly to "[Installing, Upgrading, or Downgrading SAM-QFS Software on Hosts](#)" on page 4-6.

Installing Solaris Cluster Software (High-Availability Configurations Only)

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.

Upgrading Shared SAM-QFS 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 SAM-QFS metadata server available at all times, so that file-system data remain available.

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

- [Upgrade Any Significantly Older Releases of SAM-QFS](#)
- [Carry Out the Rolling Upgrade](#)

Upgrade Any Significantly Older Releases of SAM-QFS

At any given time, the SAM-QFS 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 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.

Proceed as follows:

1. If any client hosts are not running the same release of the SAM-QFS software as the metadata server, upgrade them to the release used on the server before proceeding.

2. If the 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 Upgrade](#), one release level at a time, until all hosts are fully up to date.
3. If the 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 ["Installing, Upgrading, or Downgrading SAM-QFS Software on Hosts"](#) on page 4-6.

Carry Out the Rolling Upgrade

1. Make sure that you [Upgrade Any Significantly Older Releases of SAM-QFS](#) 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, `first-mds`. Then, in a second terminal window, we use secure shell (`ssh`) to log in to the inactive, potential metadata server, `second-mds`:

```
[first-mds]root@solaris:~#

[first-mds]root@solaris:~# ssh root@second-mds
Password:
[second-mds]root@solaris:~#
```

3. Upgrade the currently inactive, second metadata server. Install the updated SAM-QFS software using the procedures in ["Installing, Upgrading, or Downgrading SAM-QFS Software on Hosts"](#) on page 4-6.
4. Once the upgrade steps are complete, prepare to activate the second server. If the first, active metadata server mounts a 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 *StorageTek Storage Archive Manager and QFS Software Maintenance and Administration Guide*.

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

5. On the second metadata server, load the SAM-QFS configuration files and start SAM-QFS processes. Use the command `samd config`.

```
[second-mds]root@solaris:~# samd config
[second-mds]root@solaris:~#
```

6. On the second metadata server, mount the SAM-QFS file system.

```
[second-mds]root@solaris:~# mount sharefs1
[second-mds]root@solaris:~#
```

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 SAM-QFS shared file system.

In the example, the potential metadata server is `second-mds` and the file system name is `sharefs1`:

```
[second-mds]root@solaris:~# samsharefs -s second-mds sharefs1
[second-mds]root@solaris:~#
```

8. Upgrade the now-inactive first metadata server. Install the updated SAM-QFS software using the procedures in ["Installing, Upgrading, or Downgrading SAM-QFS Software on Hosts"](#) on page 4-6.
9. Once the upgrade steps are complete, prepare to re-activate the first metadata server. If the currently active second metadata server mounts a 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.

```
[second-mds]root@solaris:~# samcmd aridle
[second-mds]root@solaris:~# samcmd stidle
...
[second-mds]root@solaris:~# samd stop
[second-mds]root@solaris:~#
```

10. On the first metadata server, load the SAM-QFS configuration files and start SAM-QFS processes. Use the command `samd config`.

```
[first-mds]root@solaris:~# samd config
[first-mds]root@solaris:~#
```

11. On the first metadata server, mount the SAM-QFS file system.

```
[first-mds]root@solaris:~# mount sharefs1
[first-mds]root@solaris:~#
```

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 SAM-QFS shared file system.

In the example, the potential metadata server is `first-mds` and the file system name is `sharefs1`:

```
[first-mds]root@solaris:~# samsharefs -s second-mds sharefs1
[first-mds]root@solaris:~#
```

13. Update the remaining clients. Install the updated SAM-QFS software using the procedures in ["Installing, Upgrading, or Downgrading SAM-QFS Software on](#)

[Hosts](#)" on page 4-6.

14. Stop here. The upgrade is complete.

Installing, Upgrading, or Downgrading SAM-QFS Software on Hosts

To install, upgrade, or downgrade SAM-QFS software on an individual host, carry out the following tasks:

- [Install, Upgrade, or Downgrade SAM-QFS Software on Oracle Solaris Hosts](#).
- [Install or Update SAM-QFS Client Software on Linux Hosts](#) (if any).

Install, Upgrade, or Downgrade SAM-QFS Software on Oracle Solaris Hosts

To install, upgrade, or downgrade SAM-QFS 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, [Install the Software Using the Image Packaging System \(IPS\) command `pkg install`](#).
- If you are upgrading or downgrading software that was installed using the IPS command `pkg install`, [Upgrade or Downgrade the Software Using the Image Packaging System \(IPS\) command `pkg update`](#).
- If you are installing new software on a Solaris 10 host, [Install the Software Using SVR4 `pkgrm` and `pkgadd` Commands](#).
- If you are upgrading software that was installed using the SVR4 command `pkgadd`, [Upgrade or Downgrade the Software Using SVR4 `pkgrm` and `pkgadd` Commands](#).

Prepare the Host for Software Changes

1. If SAM-QFS software is not currently installed on the host system, go to "[Locate the Packages for Your Host Architecture](#)".
2. Otherwise, log in to the SAM-QFS server as `root`.

```
[samqfs1host]root@solaris:~#
```

3. If SAM-QFS 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:

```
[samqfs1host]root@solaris:~# samcmd aridle  
[samqfs1host]root@solaris:~#
```

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:

```
[samqfs1host]root@solaris:~# samcmd stidle  
[samqfs1host]root@solaris:~#
```

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:

```
[samqfs1host]root@solaris:~# samcmd a
Archiver status samcmd      5.4 10:20:34 May 20 2014
samcmd on samqfs1host
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:

```
[samqfs1host]root@solaris:~# samcmd u
Staging queue samcmd      5.4 10:20:34 May 20 2014
samcmd on solaris.demo.lan
Staging queue by media type: all
sam-stagerd: Waiting for :strun
[samqfs1host]root@solaris:~#
```

7. 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:

```
[samqfs1host]root@solaris:~# samcmd 801 idle
[samqfs1host]root@solaris:~# samcmd 802 idle
[samqfs1host]root@solaris:~# samcmd 803 idle
[samqfs1host]root@solaris:~# samcmd 804 idle
[samqfs1host]root@solaris:~#
```

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.

```
[samqfs1host]root@solaris:~# samcmd r
Removable media samcmd      5.4 18:37:09 Feb 17 2014
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
[samqfs1host]root@solaris:~#
```

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`.

```
[samqfs1host]root@solaris:~# samd stop
[samqfs1host]root@solaris:~#
```

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 SAM-QFS file system.

In the first example, we stop NFS sharing of the SAM-QFS standalone file system `samqfs1`.

```
[samqfs1host]root@solaris:~# unshare /samqfs1
[samqfs1host]root@solaris:~#
```

In the second example, we stop NFS sharing of the SAM-QFS shared file system `samqfs2`:

```
[samqfs2server]root@solaris:~# unshare /samqfs2
[samqfs2server]root@solaris:~#
```

11. Unmount all SAM-QFS file systems.

In the first example, we unmount the unshared, standalone file system `samqfs1`:

```
[samqfs1host]root@solaris:~# umount samqfs1
```

In the second example, we unmount the shared file system `samqfs1`, first from the clients and then from the server, allowing 60 seconds for clients to unmount.

```
[samqfs2server]root@solaris:~# ssh root@samqfs2client1
Password:
[samqfs2client1]root@solaris:~# umount /samqfs2
[samqfs2client1]root@solaris:~# exit
[samqfs2server]root@solaris:~#
```

```
[samqfs2server]root@solaris:~# ssh root@samqfs2client1
Password:
[samqfs2client2]root@solaris:~# umount /samqfs2
[samqfs2client2]root@solaris:~# exit
[samqfs2server]root@solaris:~# 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:

```
[host1]root@solaris:~# yes | pkgrm SUNWsamfsu SUNWsamfsr
```

13. Next, [Locate the Packages for Your Host Architecture](#).

Locate the Packages for Your Host Architecture

1. Log in to the SAM-QFS host as `root`.

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

2. Change to the subdirectory of the directory where the SAM-QFS download file was unpacked, either `STK_SAM-QFS_release-number` or `STK_SAM-QFS_release-number-patch-number`.

The initially released packages are stored in the subdirectory `STK_SAM-QFS_release-number`, where *release-number* is a major and a minor release number, joined by a dot: `STK_SAM-QFS_5.4/`. Patch releases (if any) are

located in the subdirectory `STK_SAM-QFS_release-number-patch-number`, where *patch-number* is a two-digit patch sequence number: `STK_SAM-QFS_5.4-01/`.

In the example we change to the download directory for the initial release of the software, `STK_SAM-QFS_5.4/` and list the contents:

```
root@solaris:~# cd /net/sw-install/samqfs/STK_SAM-QFS_5.4/
root@solaris:~# ls -l
./
../
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@solaris:~# cd solaris_sparc/
root@solaris:~# ls -l
./
../
S10/
S11/
S11_ips/
fsmgr_5.4.zip
fsmgr_setup*
```

4. When Solaris 11 or later is installed on the host, you can install the software using the Image Packaging System. To use IPS, change to the subdirectory `S11_ips/` and [Install the Software Using the Image Packaging System \(IPS\)](#) or [Upgrade or Downgrade the Software Using the Image Packaging System \(IPS\)](#).

```
root@solaris:~# cd S11_ips/
```

5. Alternatively, when Solaris 11 or later is installed on the host, you can also install the software using the `pkgadd` method. In this case, change to the subdirectory `S11/` and either [Upgrade or Downgrade the Software Using SVR4 `pkgrm` and `pkgadd` Commands](#).

```
root@solaris:~# cd S11_ips/
```

6. When Solaris 10 is installed on the host, change to the subdirectory `S10/` and [Upgrade or Downgrade the Software Using SVR4 `pkgrm` and `pkgadd` Commands](#).

```
root@solaris:~# cd S10/
```

Install the Software Using the Image Packaging System (IPS)

Use Image Packaging System (IPS) commands to install, upgrade, or downgrade SAM-QFS 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 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 the first release of SAM-QFS 5.4, `STK_SAM-QFS_5.4/solaris_sparc/S11_ips/repo.samqfs/`:

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

- To install both the StorageTek Storage Archive 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 SAM-QFS Image Packaging System package names.

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

- 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 SAM-QFS Image Packaging System package names.

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

- When the packages finish installing, run the post-installation script, `SAM-QFS-post-install`. It is located in the `util/` subdirectory of the SAM-QFS installation directory (either `/opt/SUNWsamfs/` or `/opt/SUNWqfs/`).

In the example, we run `/opt/SUNWsamfs/util/SAM-QFS-post-install`:

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

6. Add the SAM-QFS directories `/opt/SUNWsamfs/bin` and `/opt/SUNWsamfs/sbin` (or `/opt/SUNWqfs/bin` and `/opt/SUNWqfs/sbin`) to the system `PATH` variable, if it is not already in path.
7. Add the SAM-QFS 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 SAM-QFS configuration includes additional Solaris hosts, repeat this procedure from the beginning until the software is installed on all hosts.
9. If the planned SAM-QFS configuration includes Linux hosts as shared file-system clients, go to [Install or Update SAM-QFS Client Software on Linux Hosts](#).
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 SAM-QFS 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 Packages for Your Host Architecture](#).
2. To upgrade the StorageTek Storage Archive 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 SAM-QFS Image Packaging System package names.

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

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 SAM-QFS Image Packaging System package names.

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

4. To downgrade the SAM-QFS 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 SAM-QFS package.

In the example, SAM-QFS version 5.4.9, the first patch release, is installed on the host:

```
root@solaris:~# samcmd l
Usage information samcmd      5.4.9 14:06:14 Dec  9 2014 ...
root@solaris:~#
```

We need to downgrade to version 5.4.6, the first version of the software. So we run the `pkg info` commands for `SUNWsamfs` and `SUNWsamqassy` in the IPS repository for the first release, `STK_SAM-QFS_5.4/solaris_sparc/S11_ips/repo.samqfs`:

```
root@solaris:~# pwd
/net/STK_SAM-QFS_5.4/solaris_sparc/S11_ips/repo.samqfs
root@solaris:~# 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://samqfs/SUNWsamfs@5.4,5.11:20140708T225656Z

root@solaris:~# 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://samqfs/SUNWsamqassy@5.4,5.11:20140926T172135Z
root@solaris:~#
```

- Then, to downgrade the SAM-QFS 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 FMRI of the 5.4.6 versions of the `SUNWsamfs` and `SUNWsamqassy` packages:

```
root@solaris:~# 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@solaris:~# pkg update -g . SUNWsamqassy@5.4,5.11:20140926T172135Z
...
root@solaris:~#
```

6. If the planned SAM-QFS configuration includes additional Solaris hosts, repeat this procedure from the beginning until the software has been updated on all hosts.
7. If the planned SAM-QFS configuration includes Linux hosts as shared file-system clients, go to [Install or Update SAM-QFS Client Software on Linux Hosts](#).

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

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

For each SAM-QFS 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 Packages for Your Host Architecture](#).
2. To install both the StorageTek Storage Archive 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, `STK_SAM-QFS_5.4/solaris_sparc/S10`. Then we pipe the reply `yes` into the command so that all questions are automatically answered:

```
root@solaris:~# pwd
/net/STK_SAM-QFS_5.4/solaris_sparc/s10
root@solaris:~# 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@solaris:~# yes | pkgadd -d . SUNWqfsr SUNWqfsu
```

4. If the planned SAM-QFS configuration includes Linux hosts as shared file-system clients, go to [Install or Update SAM-QFS Client Software on Linux Hosts](#).
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 installing SAM-QFS software on hosts that run Solaris 10 and when you are upgrading software that was originally installed using SVR4 commands.

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

1. If you are downgrading the SAM-QFS 5.4 software to version 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 SAM-QFS 5.4 configuration files for use with SAM-QFS 5.3:

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

2. Uninstall all SAM-QFS 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@solaris:~# yes | pkgrm SUNWsamfsu SUNWsamfsr
```

3. If you have not already done so, [Locate the Packages for Your Host Architecture](#).
4. To install both the StorageTek Storage Archive 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, `STK_SAM-QFS_5.4/solaris_sparc/S10`. Then we pipe the reply `yes` into the command so that all questions are automatically answered:

```
root@solaris:~# pwd
/net/STK_SAM-QFS_5.4/solaris_sparc/s10
root@solaris:~# 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@solaris:~# pwd
/net/STK_SAM-QFS_5.4/solaris_sparc/s10
root@solaris:~# yes | pkgadd -d . SUNWqfsr SUNWqfsu
```

6. If the planned SAM-QFS configuration includes Linux hosts as shared file-system clients, go to [Install or Update SAM-QFS Client Software on Linux Hosts](#).
7. Otherwise, go to [Chapter 5, "Using the samsetup Configuration Wizard"](#) or [Chapter 6, "Configuring the Basic File System"](#).

Install or Update SAM-QFS Client Software on Linux Hosts

For each Linux client of a SAM-QFS shared file system, proceed as follows:

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

```
[root@linux ~]#
```
2. Unmount all mounted SAM-QFS file systems.
3. Uninstall old SAM-QFS packages.

In the example, we use the command `rpm --erase SUNWsamfsu SUNWsamfsr`:

```
[root@linux ~]# rpm --erase SUNWsamfsu SUNWsamfsr
```

4. Locate the Linux client ISO image. The ISO image is in the directory where you downloaded the SAM-QFS installation software (see "Obtaining Software" on page 4-1).

In the example, the installation software is located on the host `sw_install` (IP address `192.168.0.2`) in the directory `/samqfs`:

```
[sw_install]root@solaris:~# ls -l
./
../
README.txt
STK_SAM-QFS_5.4/
STK_SAM-QFS_5.4.zip
STK_SAM-QFS_5.4_linux.iso
iso.md5
```

5. Mount the ISO image on the Linux host.

In the first example, we mount the image using NFS. We first NFS mount the remote directory that holds the ISO image using `192.168.0.2` as the IP address for the installation server `sw_install`. Then we NFS mount the ISO image itself on the mount-point directory `/mnt/samqfs_iso`:

```
[root@linux ~]# mount -t nfs 192.168.0.2:/samqfs /mnt/samqfs
[root@linux ~]# mount -o ro,loop -t iso9660 \
/mnt/samqfs/STK_SAM-QFS_5.4_linux.iso /mnt/samqfs_iso
[[root@linux ~]#
```

In the second example, we mount a local image that we have copied into the temporary directory `/home/samqfs_temp`:

```
[root@linux ~]# cd /home/samqfs_temp
[root@linux ~]# mount -o ro,loop -t iso9660 STK_SAM-QFS_5.4_linux.iso /mnt
```

6. Change to the **Linux1/** subdirectory of the mounted ISO image, and run the installer. Use the command `./Install`.

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

7. 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 SAM-QFS installation program compiles custom kernel modules so that it can support the largest possible number of variations.

8. Follow the on-screen instructions.
9. 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 `lm` 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 1 8 3 2 3posix 3pm 3perl 5 4 9 6 7
```

10. If the planned SAM-QFS configuration includes additional Linux client hosts, repeat this procedure from the beginning until the client software is installed on all hosts.
11. Otherwise, go to ["Using the samsetup Configuration Wizard"](#) on page 5-1 or ["Configuring the Basic File System"](#) on page 6-1.

Uninstalling SAM-QFS Software

This section outlines following procedures:

- [Uninstall SAM-QFS on a Solaris Host](#)
- [Uninstall the SAM-QFS Client on a Linux Host.](#)

Caution: Do not uninstall software if you intend to upgrade or reinstall SAM-QFS using an existing configuration! Uninstalling removes all configuration files. Instead, use one of the upgrade methods outlined in ["Install, Upgrade, or Downgrade SAM-QFS Software on Oracle Solaris Hosts"](#) on page 4-6.

Uninstall SAM-QFS 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@solaris:~#
```
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@solaris:~# 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@solaris:~# 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@solaris:~# rm -R /var/opt/SUNWsamfs/  
root@solaris:~# rm -R /etc/opt/SUNWsamfs/  
root@solaris:~# rm -R /var/adm/sam-log/  
root@solaris:~#
```

5. Reboot the host.

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

6. Stop here.

Uninstall the SAM-QFS 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 SAM-QFS script `/var/opt/SUNWsamfs/Uninstall` (`/var/opt/SUNWqfs/Uninstall` if only QFS is installed).

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
```

Using the `samsetup` Configuration Wizard

The `samsetup` wizard is a simple, text-based, menu-driven utility that lets you quickly create and configure SAM-QFS 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 SAM-QFS 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 SAM-QFS 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 SAM-QFS:

- "[Configuring the Basic File System](#)" on page 6-1, provides essential information about the way SAM-QFS works and 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 a SAM-QFS archiving file system, you will need the information in "[Configure File System Protection](#)" on page 6-38. The `samsetup` wizard does not configure scheduled backups of critical file-system metadata and logs.
- If you require a SAM-QFS shared file system, review "[Accessing File Systems from Multiple Hosts](#)" on page 7-1 as well. The sections "[Accessing File Systems from Multiple Hosts Using SAM-QFS Software](#)" on page 7-1 and "[Configuring a SAM-QFS Shared File System](#)" on page 7-7 will be particularly relevant.
- If you need to use the additional features of the SAM-QFS, you should consult the relevant sections of this book for additional configuration instructions. See, for example, "[Configure Archival Media Validation](#)" on page 6-44, "[Enabling Support for Write Once Read Many \(WORM\) Files](#)" on page 6-51, "[Enabling Support for the Linear Tape File System \(LTFS\)](#)" on page 6-54, "[Preparing High-Availability Solutions](#)" on page 9-1, "[Configuring SAM-Remote](#)" on page 8-1, and "[Configuring the SAM-QFS Reporting Database](#)" on page 10-1.
- Finally, whether you configure file systems using `samsetup` or using the command line or File System Manager user interface, you should protect your work as

described in "[Backing Up the SAM-QFS Configuration](#)" on page 13-1.

Configuring the Basic File System

[QFS File Systems](#) are the basic building blocks of all SAM-QFS solutions. Used on their own, they offer high performance, effectively unlimited capacity, and support for extremely large files. When used with Storage Archive Manager and suitably configured archival storage, they become [SAM-QFS 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](#)
- [Configuring SAM-QFS Archiving File Systems](#)

Configuring QFS File Systems

Creating and configuring a basic QFS file system is straightforward. In each case, you perform the following tasks:

- Prepare the disk devices that will support the file system.
- Create a Master Configuration File (mcf).
- Create the file system using the `/opt/SUNWsamfs/sbin/sammkfs` command.
- Add the new file system to the host's virtual file system configuration by editing the `/etc/vfstab` file.
- Mount the new file system.

The process can be performed using either the graphical File System Manager interface or a text editor and commandline terminal. But in the examples, we use the editor-and-commandline method to make the parts of the process explicit and thus easier to understand.

For simplicity and convenience during an initial SAM-QFS configuration, the procedures in this section set file-system mount options in the configuration file for the Solaris virtual file system, `/etc/vfstab`. But most options can also be set in an optional `/etc/opt/SUNWsamfs/samfs.cmd` file or from the command line. See the `samfs.cmd` and `mount_samfs` man pages for details.

Prepare Disk Storage for a QFS File System

Before you start the configuration process, select the disk resources required for your planned configuration. You can use raw device slices, ZFS zvol volumes, or Solaris Volume Manager volumes.

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@solaris:~#
```

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 SAM-QFS File System Manager graphical user interface or by using a text editor. In the example below, we use the `vi` text editor:

```
root@solaris:~# 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.

In this example, the file system is named `qfsms`:

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

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 SAM-QFS. In this example, we use `100` for the `qfsms` file system:

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

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

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
#-----
qfsms           100      ms        qfsms    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 `qfsms` 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
#-----
qfsms           100      ms        qfsms    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`:

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

11. Enter the family set name *of the file system* 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 `qfsms` identifies the disk equipment as part of the hardware for the file system.

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

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

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

The `sam-fsd` command reads SAM-QFS configuration files and initializes file systems. It will stop if it encounters an error:

```
root@solaris:~# 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@solaris:~# sam-fsd
Problem in mcf file /etc/opt/SUNWsamfs/mcf for filesystem qfsms
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:

```
qfsms          100      ms          qfsms          on
/dev/dsk/c0t0d0s0 101      md          qfsms          on
/dev/dsk/c0t3d0so 102      md          qfsms          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@solaris:~# 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. Create a mount-point directory for the new file system, 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 `/qfsms` mount-point directory and set permissions to 755 (`-rwxr-xr-x`):

```
root@solaris:~# mkdir /qfsms
root@solaris:~# chmod 755 /qfsms
```

17. Tell the SAM-QFS software to reread the `mcf` file and reconfigure itself accordingly. Use the command `samd config`.

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

18. 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@solaris:~# /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@solaris:~#
```

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

The SAM-QFS 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@solaris:~# sammkfs qfsms
Building 'qfsms' 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 were using `mr` devices needed to specify a non-default DAU size that better met our I/O requirements, we could do so by using the `sammkfs` command with the `-a` option:

```
root@solaris:~# sammkfs -a 16 qfs2ma
```

For additional information, see the `sammkfs` man page.

20. Back up the operating system's `/etc/vfstab` file.

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

21. Add the new file system to the operating system's virtual file system configuration. Open the file in a text editor, and start a line for the `qfsms` family set device:

```
#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    -
...
qfsms  -      /qfsms  samfs  -
```

22. In the sixth column of the `/etc/vfstab` file, `Mount at Boot`, enter `no` in most cases.

```
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    -
...
qfsms    -      /qfsms  samfs   -      no
```

23. To specify round-robin allocation, add the `stripe=0` mount 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 -
...
qfsms - /qfsms samfs - no stripe=0
```

24. To specify striped allocation, add the `stripe=stripe-width` mount option, where `stripe-width` is the number of Disk Allocation Units (DAUs) that should be written to each disk in the stripe.

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 -
/proc - /proc proc - no -
...
qfsms - /qfsms samfs - no stripe=1
```

Here, the `stripe=1` option specifies a stripe width of 1 DAU and a write size of two DAUs. So, when the file system writes two DAUs at a time, it writes one to each of the two md disk devices in the `qfsms` family set.

25. Make any other desired changes to the `/etc/vfstab` file.

For example, to mount the file system in the background if the metadata server is not responding, you would add the `bg` mount option to the `Mount Options` field:

```
#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 -
...
qfsms - /qfsms samfs - no stripe=1,bg
```

26. Save the `vfstab` file, and close the editor.

```
...
qfsms - /qfsms samfs - no stripe=1
:wq
root@solaris:~#
```

27. Mount the new file system:

```
root@solaris:~# mount /qfsms
```

28. The file system is now complete and ready to use.

Where to go from here:

- If you are using Storage Archive Manager to set up an archiving file system, see "[Configuring SAM-QFS Archiving File Systems](#)" on page 6-11

- 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 6-51.
- 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 6-54.
- If you have additional requirements, such as multiple-host file-system access or high-availability configurations, see ["Beyond the Basics"](#) on page 6-55.

Configure a High-Performance `ma` File System

Once the SAM-QFS 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@solaris:~#
```

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 SAM-QFS File System Manager graphical user interface or by using a text editor. In the example below, we use the `vi` text editor:

```
root@solaris:~# 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 `qfsma`, increment the equipment ordinal to 200, set the equipment type to `ma`, set the family set name to `qfsma`, and set the device state on:

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

```
qfsma          200      ma      qfsma  on
  /dev/dsk/c0t0d0s0  201      mm      qfsma  on
```

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

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

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

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

The `sam-fsd` command reads SAM-QFS configuration files and initializes file systems. It will stop if it encounters an error:

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

- 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@solaris:~# sam-fsd
Problem in mcf file /etc/opt/SUNWsamfs/mcf for filesystem qfsma
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      qfsma  on
  /dev/dsk/c0t0d0s0  201      mm      qfsma  on
  /dev/dsk/c0t0d0s!  202      md      qfsma  on
  /dev/dsk/c0t3d0s0  203      md      qfsma  on
```

- 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@solaris:~# 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@solaris:~# sammkfs qfsma
Building 'qfsma' 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-kilobyte DAU, we would use the `sammkfs` command with `-a` option:

```
root@solaris:~# sammkfs -a 16 qfsma
```

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

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

14. Add the new file system to the operating system's virtual file system configuration. Open the `/etc/vfstab` file in a text editor, and start a line for the `qfsma` family set.

```
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   -
```

15. In the sixth column of the `/etc/vfstab` file, `Mount at Boot`, enter `no`.

```
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   -     no
```

16. To specify round-robin allocation, add the `stripe=0` mount option:

```
#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 - no stripe=0
```

- 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 -
...
qfsma - /qfsma samfs - no stripe=1
```

- 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`, SAM-QFS 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 -
...
qfsma - /qfsma samfs - no ...,stripe=2
```

- Otherwise, save the `vfstab` file.

```
...
qfsma - /qfsma samfs - no stripe=1
:wq
root@solaris:~#
```

- Mount the new file system:

```
root@solaris:~# mount /qfsms
```

The basic file system is now complete and ready to use.

21. If you are using Storage Archive Manager to set up an archiving file system, see ["Configuring SAM-QFS Archiving File Systems"](#) on page 6-11.
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 6-51.
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 6-54.
24. If you have additional requirements, such as multiple-host file-system access or high-availability configurations, see ["Beyond the Basics"](#) on page 6-55.
25. Otherwise, go to ["Configuring Notifications and Logging"](#) on page 11-1.

Configuring SAM-QFS Archiving File Systems

Archiving file systems combine one or more QFS ma- or ms-type file systems with archival storage and Oracle StorageTek Storage Archive Manager (SAM) software. The SAM software integrates secondary disk storage and/or removable media into the basic file-system operations, so that files are maintained in multiple copies on varied media. This redundancy provides continuous data protection and supports policy-driven retention and efficient storage of extremely large files.

- [Prepare Disk Storage for Archival Copies](#)
- [Configure the Archiving File System](#)
- [Mount the Archiving File System](#)
- [Configure the Archiving Process](#)
- [Catalog Archival Media Stored in a Network-Attached Tape Library](#)
- [Configure File System Protection](#)
- [Configure Archival Media Validation](#)
- [Enable the WORM Support on a SAM-QFS File System](#)

Prepare Disk Storage for Archival Copies

SAM-QFS archiving file systems can copy files from the primary file-system disk cache to either tape volumes or disk-based file systems that have been configured as *disk archives*. In the latter case, SAM-QFS uses each file system more or less as it would a tape cartridge and addresses it using an assigned volume serial number (VSN). Disk-archive volumes can be significantly more responsive when small files are frequently archived, re-accessed, and/or modified, because random-access disk devices do not incur the mounting and positioning overhead associated with sequential-access tape devices.

1. Determine the number of file systems that you are likely to need. For best performance, one SAM-QFS operation should read or write to one disk volume at a time, as with tape volumes. So the number of required volumes depends of the workload that you identified when gathering and defining requirements.

In typical deployments, a number between 15 and 30 volumes is usually about right.

2. Identify the disk resources and total capacity that can be made available for disk archiving.

3. Calculate the number of disk volumes that you can actually create from the available resources. Allow 10 to 20 terabytes per volume. If the total available capacity is less than 10 terabytes, you can create a single archive volume.
4. Configure a file system for each archive volume.

You can use any combination of local or NFS-mounted, QFS, ZFS, and/or UFS file systems as archive volumes (NFS-mounted volumes can be particularly useful for creating off-site archive copies).

Do not try to use subdirectories of a single file system as archival volumes. If multiple volumes are defined on a single set of physical devices, multiple SAM-QFS operations will contend for the same resources. This situation can drastically increase disk overhead and severely reduce 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 UFS file systems mounted on a remote server named server.
5. If you configure one or more QFS file systems as archival storage volumes, assign each a family set name and a range of equipment ordinal numbers that clearly identifies it as an archival storage volume.

Clearly distinguishing the QFS archival storage file system from other SAM-QFS primary file systems makes configuration easier to understand and maintain. In this example, the new file system DISKVOL1 indicates its function. In the `mcf` file, this name and the equipment ordinal 800 will distinguish the disk archive from `samms` and 100, the family set name and ordinal number that we will use when we create an archiving SAM-QFS file system in subsequent examples:

```
# Archiving file systems:
#
# Equipment          Equipment Equipment Family   Device Additional
# Identifier         Ordinal   Type     Set     State Parameters
#-----
#
# Archival storage for copies:
#
# Equipment          Equipment Equipment Family   Device Additional
# Identifier         Ordinal   Type     Set     State Parameters
#-----
DISKVOL1          800      ms       DISKVOL1 on
/dev/dsk/c6t0d1s7   801      md       DISKVOL1 on
/dev/dsk/c4t0d2s7   802      md       DISKVOL1 on
```

6. On the SAM-QFS host, create a single parent directory to hold the mount points for the archival disk volumes, much as a physical tape library holds archival tape volumes.

In the example, we create the directory `/diskvols`.

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

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

```

8. On the SAM-QFS host, back up the `/etc/vfstab` file. Then open it in an editor, add entries for each archival file system, and add the mount option `nosam` to each 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:~# cp /etc/vfstab /etc/vfstab.backup
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:~#

```

9. On the SAM-QFS host, mount the archival file system(s).

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

```

10. Now, [Prepare Removable Media Libraries and Drives](#).

Prepare Removable Media Libraries and Drives

This section addresses the following tasks:

- [Configure an Oracle StorageTek ACSLS Network-Attached Automated Library](#)
- [Configure Labeling Behavior for Barcoded Removable Media](#)
- [Set Drive Timing Values](#)

Configure an Oracle StorageTek ACSLS Network-Attached Automated Library

If you have an Oracle StorageTek ACSLS network-attached library, you can configure it as follows or you can use the SAM-QFS Manager graphical user interface to automatically discover and configure the library (for instructions on using SAM-QFS Manager, see the online help).

Proceed as follows:

1. Log in to the SAM-QFS server host as `root`.

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

2. Change to the `/etc/opt/SUNWsamfs` directory.

```
root@solaris:~# 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@solaris:~# vi /etc/opt/SUNWsamfs/acsls1params
# Configuration File for an ACSLS Network-Attached Tape Library 1
```

4. Enter the parameters and values that the SAM-QFS software will use when communicating with the ACSLS-attached library.

The SAM-QFS 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 SAM-QFS software.
- `ssihost=hostname` specifies the hostname that identifies a multihomed SAM-QFS 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 SAM-QFS 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=lsnum, cap=capnum)` specifies the ACSLS address of a cartridge access port (CAP), where `acsnum` is the Automated Cartridge System (ACS) number for the library, `lsnum` 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 ACSLS 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=Drivenum)` [shared] specifies the ACSLS address of a drive that is attached to the client, where `device-path-name` identifies the device on the SAM-QFS 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 `Drivenum` 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 SAM-QFS 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@solaris:~# 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@solaris:~# 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@solaris:~#
```

6. If required, [Configure Labeling Behavior for Barcoded Removable Media](#) or [Set Drive Timing Values](#).
7. Otherwise, go to "[Configure the Archiving File System](#)" on page 6-19.

Configure Labeling Behavior for Barcoded Removable Media

If you have a tape library that uses a barcode reader, you can configure SAM-QFS to base volume labels on the barcodes by using the `labels` directive in the `defaults.conf` file. Proceed as follows:

1. Log in to the SAM-QFS host as `root`.

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

2. If you need the library to automatically label each volume using the first six characters of the barcode on the media and have not changed the defaults, stop here. If required [Set Drive Timing Values](#). Otherwise, go to "[Configure the Archiving File System](#)" on page 6-19.

By default, if a library holds a barcode reader and barcoded media, SAM-QFS software automatically labels the volumes with the first six characters in the barcode.

3. If you require a non-default behavior or if you have previously overridden the default, open the file `/etc/opt/SUNWsamfs/defaults.conf` in a text editor.

In the example, we open the file in the vi editor:

```
root@solaris:~# vi /opt/SUNWsamfs/examples/defaults.conf
...
```

4. Locate the directive line `labels =`, if present add it if it is not present.

In the example, we add the directive:

```
root@solaris:~# vi /etc/opt/SUNWsamfs/defaults.conf
# These are the defaults.
...
labels =
```

5. 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 SAM-QFS software now automatically relabels an unlabeled tape using the first six characters of the tape's barcode as the label:

```
root@solaris:~# vi /etc/opt/SUNWsamfs/defaults.conf
...
labels = barcodes
:wq
root@solaris:~#
```

6. To enable automatic labeling based on the last six characters of the barcode on a tape, 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 SAM-QFS software automatically relabels an unlabeled tape using uses the last six characters of the tape's barcode as the label:

```
root@solaris:~# vi /etc/opt/SUNWsamfs/defaults.conf
...
labels = barcodes_low
:wq
root@solaris:~#
```

7. To disable automatic labeling and configure SAM-QFS to read labels from tapes, set the value of the `labels` directive to `read`. Save the file, and close the editor.

When the `labels` directive is set to the value `read`, the SAM-QFS software cannot automatically relabel tapes:

```
root@solaris:~# vi /etc/opt/SUNWsamfs/defaults.conf
...
labels = read
idle_unload = 0
...
:wq
root@solaris:~#
```

8. If required, [Set Drive Timing Values](#).

- Otherwise, go to ["Configure the Archiving File System"](#) on page 6-19.

Set Drive Timing Values

By default, the SAM-QFS 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 SAM-QFS 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 SAM-QFS software waits before unloading an idle drive is 600 seconds (10 minutes).
- The amount of time that SAM-QFS software waits before unloading an idle drive that is shared by two or more SAM-QFS servers is 600 seconds (10 minutes).

To change the default timing values, proceed as follows:

- If you are not logged in, log in to the SAM-QFS host as `root`.

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

- Open the `/etc/opt/SUNWsamfs/defaults.conf` file in a text editor.

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

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

- 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, SAM-QFS 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@solaris:~# 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
```

- If required, specify the amount of time that SAM-QFS 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, SAM-QFS 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@solaris:~# 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 SAM-QFS 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@solaris:~# 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 SAM-QFS 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 SAM-QFS 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@solaris:~# 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@solaris:~# 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@solaris:~#

```

8. Otherwise, [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 a General-Purpose ms File System](#) or [Configure a General-Purpose ms File System](#).

While you can use the SAM-QFS File System 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 `samms` and the equipment ordinal number 100:

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

```
# Archiving file systems:
```

```
#
```

# Equipment Identifier	Equipment Ordinal	Equipment Type	Family Set	Device State	Additional Parameters
samms	100	ms	samms	on	
/dev/dsk/c1t3d0s3	101	md	samms	on	
/dev/dsk/c1t3d0s4	102	md	samms	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 900, the range following the range chosen for our disk archive:

```
# Archival storage for copies:
```

```
#
```

# Equipment Identifier	Equipment Ordinal	Equipment Type	Family Set	Device State	Additional Parameters
DISKVOL1	800	ms	DISKVOL1	on	
/dev/dsk/c6t0d1s7	801	md	DISKVOL1	on	
/dev/dsk/c4t0d2s7	802	md	DISKVOL1	on	
/dev/scsi/changer/c1t0d5	900				

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 `library1`:

```
# Archival storage for copies:
```

```
#
```

# Equipment Identifier	Equipment Ordinal	Equipment Type	Family Set	Device State	Additional Parameters
DISKVOL1	800	ms	DISKVOL1	on	
/dev/dsk/c6t0d1s7	801	md	DISKVOL1	on	
/dev/dsk/c4t0d2s7	802	md	DISKVOL1	on	
/dev/scsi/changer/c1t0d5	900	rb	library1	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/library1cat`:

```
# Archival storage for copies:
#
# Equipment          Equipment Equipment Family   Device Additional
# Identifier         Ordinal   Type    Set     State  Parameters
#-----
DISKVOL1            800      ms      DISKVOL1 on
/dev/dsk/c6t0d1s7  801      md      DISKVOL1 on
/dev/dsk/c4t0d2s7  802      md      DISKVOL1 on
/dev/scsi/changer/c1t0d5 900      rb      library1 on
...catalog/library1cat
```

- 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 3-4. 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 901. The `tp` equipment type specifies a generic SCSI-attached tape drive:

```
# Archival storage for copies:
#
# Equipment          Equipment Equipment Family   Device Additional
# Identifier         Ordinal   Type    Set     State  Parameters
#-----
DISKVOL1            800      ms      DISKVOL1 on
/dev/dsk/c6t0d1s7  801      md      DISKVOL1 on
/dev/dsk/c4t0d2s7  802      md      DISKVOL1 on
/dev/scsi/changer/c1t0d5 900      rb      library1 on
...catalog/library1cat
/dev/rmt/1cbn      901      tp      library1 on
/dev/rmt/0cbn     902      tp      library1 on
/dev/rmt/2cbn     903      tp      library1 on
/dev/rmt/3cbn     904      tp      library1 on
```

- Finally, if you wish to configure a SAM-QFS 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`:

```
# Archival storage for copies:
#
# Equipment          Equipment Equipment Family   Device Additional
# Identifier         Ordinal   Type    Set     State  Parameters
#-----
DISKVOL1            800      ms      DISKVOL1 on
/dev/dsk/c6t0d1s7  801      md      DISKVOL1 on
/dev/dsk/c4t0d2s7  802      md      DISKVOL1 on
/dev/scsi/changer/c1t0d5 900      rb      library1 on
...catalog/SL150cat
```

```

/dev/rmt/0cbn          901      tp      library1 on
/dev/rmt/1cbn          902      tp      library1 on
/dev/rmt/2cbn          903      tp      library1 on
/dev/rmt/3cbn          904      tp      library1 on
historian            999      hy      -        -
...catalog/historian_cat

```

7. Save the mcf file, and close the editor.

```

...
/dev/rmt/3cbn          904      tp      library1 on
historian              999      hy      -        -
...catalog/historian_cat
:wq
root@solaris:~#

```

8. Check the mcf file for errors by running the `sam-fsd` command. Correct any errors found.

The `sam-fsd` command reads SAM-QFS configuration files and initializes file systems. It will stop if it encounters an error:

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

9. If you are using one or more file systems as archival storage volumes, 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 three disk-based archival volumes: `DISKVOL1` is the QFS file system that we created locally for this purpose. `DISKVOL2` to `DISKVOL15` are UFS file systems. All are mounted on the `/diskvols/` directory:

```

root@solaris:~# vi /etc/opt/SUNWsamfs/diskvols.conf
# Volume
# Serial      Resource
# Number      Path
# -----
DISKVOL1    /diskvols/DISKVOL1
DISKVOL2    /diskvols/DISKVOL2
...
DISKVOL15   /diskvols/DISKVOL3

```

10. Create a mount-point directory for the new file system, 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 `/samms` mount-point directory and set permissions to 755 (`-rwxr-xr-x`):

```

root@solaris:~# mkdir /samms
root@solaris:~# chmod 755 /samms

```

11. Tell the SAM-QFS software to reread the mcf file and reconfigure itself accordingly. Correct any errors reported and repeat as necessary

```
root@solaris:~# /opt/SUNWsamfs/sbin/samd config
```

12. 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.
2. Back up the Solaris `/etc/vfstab` file, and open it in a text editor.

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

```
root@solaris:~# cp /etc/vfstab /etc/vfstab.backup
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     -
...
samms       -      /samms  samfs   -     yes    -
```

3. Set the *high-water mark*, the percentage disk cache utilization that causes SAM-QFS to release previously archived files from disk. In the last column of the SAM-QFS 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 need to access. But you also want to do as little staging as possible, so that you can avoid the overhead associated with mounting removable media volumes.

If the primary cache is implemented using the latest high-speed disk or solid-state devices, set the high-water mark value at 95%. Otherwise use 80-85%. In the example, we set the high-water mark to 85%:

```
root@solaris:~# cp /etc/vfstab /etc/vfstab.backup
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     -
...
samms       -      /samms  samfs   -     yes    high=85
```

4. Set the *low-water mark*, the percentage disk cache utilization that causes SAM-QFS to stop releasing previously archived files from disk. In the last column of the SAM-QFS 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. For performance reasons, you want to keep as many recently active files in cache as you can, particularly when files are frequently requested and modified. This keeps staging-related overhead to a minimum. But you do not want previously cached files to consume space needed for new files and newly accessed files that have to be staged to disk from archival copies.

If the primary cache is implemented using the latest high-speed disk or solid-state devices, set the low-water mark value at 90%. Otherwise use 70-75%. In the example, based on local requirements, we set the high-water mark to 75%:

```

root@solaris:~# cp /etc/vfstab /etc/vfstab.backup
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    -
...
samms   -      /samms  samfs  -     yes   high=85,low=75

```

5. 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 SAM-QFS file-system entry.

Partial releasing lets SAM-QFS 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 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@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    -
...
samms   -      /samms  samfs  -     yes   ...
maxpartial=128,partial=64

```

6. 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     -
...
samms            -      /samms         samfs   -     yes    ...
,partial=64
DISKVOL1        -      /diskvols/DISKVOL1 samfs -     yes    nosam
server:/DISKVOL2 -      /diskvols/DISKVOL2 nfs     -     yes
...
server:/DISKVOL15 -      /diskvols/DISKVOL15 nfs     -     yes
```

7. Save the `/etc/vfstab` file, and close the editor.

```
...
server:/DISKVOL15 -      /diskvols/DISKVOL15 nfs     -     yes
:wq
root@solaris:~#
```

8. Mount the SAM-QFS archiving file system

```
root@solaris:~# mount /samms
```

9. 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 SAM-QFS 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 *StorageTek Storage Archive Manager and 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 File System Protection](#) and maintain them properly, 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@solaris:~#
```

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@solaris:~# vi /etc/opt/SUNWsamfs/archiver.cmd
# Configuration file for SAM-QFS 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 SAM-QFS scanned the whole file system periodically. In general, you should not change this directive unless you must do so for compatibility with legacy SAM-QFS configurations.

```
# Configuration file for SAM-QFS 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 `samms`:

```

# Configuration file for SAM-QFS archiving file systems
#-----
# General Directives
archivemeta = off                # default
examine = noscan                 # default
#-----
# Archive Set Assignments
fs = samms                      # SAM-QFS Archiving File System

```

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 SAM-QFS to write the archiver log to a non-SAM-QFS 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/samms.archive.log` here, with the file-system assignment directives:

```

root@solaris:~# vi /etc/opt/SUNWsamfs/archiver.cmd
...
#-----
# Archive Set Assignments
fs = samms                      # SAM-QFS Archiving File System
logfile = /var/adm/samms.archive.log

```

6. On the next line, assign files from this file system to 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 SAM-QFS starts to search for files (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 example, we put all files found in the `samms` file system in a single archive set named `all`. We specify the path using a dot (`.`) to start the search in the mount point directory itself (`/samms`).

```

...
#-----
# Archive Set Assignments
fs = samms                      # SAM-QFS Archiving File System
logfile = /var/adm/samms.archive.log
all .

```

7. Next, add copy directives for the `allsamms` archive set of the `samms` file system. 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 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`.
- `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 three copies, each with `-norelease` until the copy reaches an archive age of 15 minutes. Copy 1 will be made using the archival disk volumes, while copies 2 and 3 will be made to tape media:

```
...
#-----
# Archive Set Assignments
fs = samms                                # SAM-QFS Archiving File System
logfile = /var/adm/samms.archive.log
all .
    1 -norelease 15m
    2 -norelease 15m
    3 -norelease 15m
```

8. Define archive sets for any remaining file systems.

In the example, we have configured a QFS file system, `DISKVOL1`, as archival media for the copy process. So we start an entry for `fs = DISKVOL1`. But we do not want to be making archival copies of archival copies. So we do not specify a log file, and we use a special archive set called `no_archive` that prevents archiving for the files in this file system:

```
...
#-----
# Archive Set Assignments
fs = samms                                # SAM-QFS Archiving File System
logfile = /var/adm/samms.archive.log
all .
    1 -norelease 15m
    2 -norelease 15m
    3 -norelease 15m
fs = DISKVOL1                            # QFS File System (Archival Media)
no_archive .
```

9. 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
```

10. If you need to set any common copy parameters that apply to all copies of all archive sets, enter a line of the form `allsets -param value ...` where `allsets` is the special archive set that represents all configured archive sets and `-param value ...` represents one or more parameter/value pairs separated by spaces.

For full descriptions of the parameters and their values, see the ARCHIVE SET COPY PARAMETERS SECTION section of the `archiver.cmd` man page.

The directive in 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:

```
...
#-----
# Copy Parameter Directives
params
allsets -sort path -offline_copy stageahead
```

11. If you need to set copy parameters for specific copies in all archive sets, enter a line of the form `allsets.copy-number -param value ...` where `allsets` is the special archive set that represents all configured archive sets, `copy-number` is the number of the copy to which the directive applies, and `-param value ...` represents one or more parameter/value pairs separated by spaces.

For full descriptions of the parameters and their values, see the ARCHIVE SET COPY PARAMETERS SECTION section of the `archiver.cmd` man page).

In the example, the directive `allsets.1 -startage 10m -startsize 500M -drives 10 -archmax 1G`, optimizes copy 1 for disk volumes. It starts archiving when the first file selected for archiving has been waiting for 10 minutes or 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 can be no larger than one gigabyte. The remaining directives, `allsets.2 -startage 24h -startsize 20G -drives 2 -archmax 24G -reserve set` and `allsets.3 -startage 48h -startsize 20G -drives 2 -archmax 24G -reserve set`, optimize copies 2 and 3 for tape media. They starts archiving when the first file selected for archiving has been waiting for 24 or 48 hours, respectively, or when the total size of all waiting files is at least 20 gigabytes. A maximum of 2 drives can be used to make these copies (adjust this number to suit your infrastructure) and each `tar` file in the copy can be no larger than 24 gigabytes. The `-reserve set` insures that copies 2 and 3 of each archive set are made using tape media that only contains copies from the same archive set:

```
...
#-----
# Copy Parameter Directives
params
allsets -sort path -offline_copy stageahead
allsets.1 -startage 10m -startsize 500M -drives 10 -archmax 1G
allsets.2 -startage 24h -startsize 20G -drives 2 -archmax 24G -reserve set
allsets.3 -startage 48h -startsize 20G -drives 2 -archmax 24G -reserve set
```

Note that the examples in this section assume the use of disk volumes for archiving. If you use only tape volumes, specify two copies and archive to tape

more frequently. The following configuration is optimal for most file systems, once you adjust the specified number of drives to suit your infrastructure:

```
allsets -sort path -offline_copy stageahead -reserve set
allsets.1 -startage 8h -startsize 8G -drives 2 -archmax 10G
allsets.2 -startage 24h -startsize 20G -drives 2 -archmax 24G
```

12. If you need to set a directive for a specific archive set and copy, enter a line of the form *archive-set-name.copy-number -param value ...*, where *archive-set-name* is the name that you used for the archive set, *copy-number* is the number of the copy to which the directive applies, and *-param value ...* represents one or more parameter/value pairs separated by spaces.

For full descriptions of the parameters and their values, see the ARCHIVE SET COPY PARAMETERS SECTION section of the `archiver.cmd` man page).

In the example below, two archive sets are defined for the `corpfs` file system: `hq` and `branches`. Note that the copy directives for `hq.1` and `hq.2` apply only to archive set `hq`. Archive set `branches` is unaffected:

```
#-----
# Archive Set Assignments
fs = corpfs
logfile = /var/adm/corporatefs.archive.log
hq /corpfs/hq/
    1 -norelease 15m
    2 -norelease 15m
branches /corpfs/branches/
    1 -norelease 15m
    2 -norelease 15m
#-----
# Copy Parameter Directives
params
hq.1 -drives 4
hq.2 -drives 2
```

13. When you have set all required copy parameters, close the copy parameters list by entering the `endparams` keyword on a new line:

```
root@solaris:~# vi /etc/opt/SUNWsamfs/archiver.cmd
...
#-----
# Copy Parameter Directives
params
allsets -sort path -offline_copy stageahead
allsets.1 -startage 10m -startsize 500M -drives 10 -archmax 1G
allsets.2 -startage 24h -startsize 20G -drives 2 -archmax 24G -reserve set
allsets.3 -startage 48h -startsize 20G -drives 2 -archmax 24G -reserve set
endparams
```

14. 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"](#), and *volumes* is a regular expression that matches one or more volume serial numbers (VSNs). Close the directives list with the `endvsnpools` keyword.

Media pools are optional, and you do not generally want to restrict the media available to the archiving process. So in these examples, we do not define media pools. For more information, see the VSN POOL DEFINITIONS SECTION of the `archiver.cmd` man page.

15. Next, start identifying the archival media that your archive set copies should use. On a new line, enter the keyword `vsns`:

```
...
#-----
# Copy Parameter Directives
params
allsets -sort path -offline_copy stageahead
allsets.1 -startage 10m -startsize 500M -drives 10 -archmax 1G
allsets.2 -startage 24h -startsize 20G -drives 2 -archmax 24G -reserve set
allsets.3 -startage 48h -startsize 20G -drives 2 -archmax 24G -reserve set
endparams
#-----
# VSN Directives
vsns
```

16. 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"](#), and `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 send the first copy of every archive set to archival disk media (type `dk`) with the volume serial numbers in the range `DISKVOL1` to `DISKVOL15`. We send the second copy of every archive set to tape media (type `tp`) with volume serial numbers in the range `VOL000` to `VOL199` and the third copy to tape media (type `tp`) with volume serial numbers in the range `VOL200` to `VOL399`:

```
...
#-----
# Copy Parameter Directives
params
allsets -sort path -offline_copy stageahead
allsets.1 -startage 10m -startsize 500M -drives 10 -archmax 1G
allsets.2 -startage 24h -startsize 20G -drives 2 -archmax 24G -reserve set
allsets.3 -startage 48h -startsize 20G -drives 2 -archmax 24G -reserve set
endparams
#-----
# VSN Directives
vsns
allsets.1 dk DISKVOL[1-15]
allsets.2 tp VOL[0-1][0-9][0-9]
allsets.2 tp VOL[2-3][0-9][0-9]
```

17. 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
allsets.1 -startage 10m -startsize 500M -drives 10 -archmax 1G
allsets.2 -startage 24h -startsize 20G -drives 2 -archmax 24G -reserve set
allsets.3 -startage 48h -startsize 20G -drives 2 -archmax 24G -reserve set
```

```

endparams
#-----
# VSN Directives
vsns
allsets.1 dk DISKVOL[1-15]
allsets.2 tp VOL[0-1][0-9][0-9]
allsets.2 tp VOL[2-3][0-9][0-9]
endvsns
:wq
root@solaris:~#

```

18. 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@solaris:~# archiver -lv
Reading '/etc/opt/SUNWsamfs/archiver.cmd'.
...
13: #File System Directives
14: #
15: fs = samms
16: logfile = /var/adm/samms.archive.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@solaris:~#

```

19. 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@solaris:~# 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
...
allsets.1
    startage: 10m startsize: 500M drives 10: archmax: 1G
Volumes:
    DISKVOL1 (/diskvols/DISKVOL15)
    ...
    DISKVOL15 (/diskvols/DISKVOL3)
Total space available: 150T
allsets.2
    startage: 24h startsize: 20G drives: 2 archmax: 24G reserve: set
Volumes:

```

```

VOL000
...
VOL199
Total space available: 300T
allsets.3
  startage: 48h startsize: 20G drives: 2 archmax: 24G reserve: set
Volumes:
VOL200
...
VOL399
Total space available: 300T
root@solaris:~#

```

20. Tell the SAM-QFS software to reread the `archiver.cmd` file and reconfigure itself accordingly. Use the `samd config` command.

```
root@solaris:~# /opt/SUNWsamfs/sbin/samd config
```

21. 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@solaris:~# vi /etc/opt/SUNWsamfs/releaser.cmd
#releaser.cmd
logfile = /var/opt/SUNWsamfs/releaser.log
list_size = 300000
:wq
root@solaris:~#

```

22. 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 based is to allow 5000 stage requests per gigabyte of host memory.

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

```

root@solaris:~# vi /etc/opt/SUNWsamfs/stager.cmd
#stager.cmd
logfile = /var/opt/SUNWsamfs/stager.log
maxactive = 300000
:wq
root@solaris:~#

```

23. Recycling is not enabled by default. So, if you require recycling of removable media volumes, go to ["Configuring the Recycling Process"](#) on page 6-33.
24. If the `mcf` file for the archiving SAM-QFS 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 6-36.

25. If you need to be able to verify the data integrity of archival tape volumes, go to "[Configure Archival Media Validation](#)" on page 6-44.
26. Otherwise, "[Configure File System Protection](#)" on page 6-38.

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:

- [Configure Recycling by Archive Set](#)

When you recycle media by archive set, you add recycling directives to the `archiver.cmd` file and 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 rather than by libraries. In a SAM-QFS 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 archival disk volumes and/or SAM-Remote.

- [Configure Recycling by Library](#)

When you recycle media by library, you add recycling directives to a `recycler.cmd` file and can 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 SAM-QFS file-system host as `root`.
2. Open the `/etc/opt/SUNWsamfs/archiver.cmd` file in a text editor.

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

3. In the `/etc/opt/SUNWsamfs/archiver.cmd` file, scroll down to the `copy params` section.


```

root@solaris:~# vi /etc/opt/SUNWsamfs/archiver.cmd
...
#-----
# 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
```

4. 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 `allsets.1` and `allsets.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@solaris:~# vi /etc/opt/SUNWsamfs/archiver.cmd
...
#-----
# Copy Parameters Directives
params
allsets -sort path -offline_copy stageahead
allsets.1 -startage 6h -startsize 6G -startcount 500000
allsets.1 -recycle_mingain 30 -recycle_hwm 60
allsets.2 -startage 6h -startsize 6G -startcount 500000
allsets.2 -recycle_mingain 90 -recycle_hwm 60
endparams
#-----
# VSN Directives
vsns
all.1 dk DISKVOL1
all.2 tp VOL0[0-1][0-9]
endvsns
:wq
[root@solaris:~#
```

5. 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.
6. If errors were found in the `archiver.cmd` file, correct them, and then re-check the file.
7. 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 SAM-QFS to write logs to a non-SAM-QFS directory, such as `/var/adm/`. In the example, we use the `vi` editor, and specify `/var/adm/recycler.log`:

```
root@solaris:~# vi /etc/opt/SUNWsamfs/recycler.cmd
logfile = /var/adm/recycler.log
:wq
root@solaris:~#
```

8. Open the `/etc/opt/SUNWsamfs/scripts/recycler.sh` script in a text editor, and enter shell commands for handling recycled removable media volumes.

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 the file 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 for long-term, historical preservation. By default, the script reminds the root user to set up the script.

9. If the `mcf` file for the archiving SAM-QFS 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 6-36.
10. Otherwise, go to ["Configure File System Protection"](#) on page 6-38.

Configure Recycling by Library

1. Log in to the SAM-QFS file-system host as root.
2. Create the `/etc/opt/SUNWsamfs/recycler.cmd` file in a text editor.

In the example, we use the vi editor.

```
root@solaris:~# vi /etc/opt/SUNWsamfs/recycler.cmd
# Configuration file for SAM-QFS archiving file systems
#-----
```

3. Specify a path and file name for the recycler log using the `logfile` directive.

Configure SAM-QFS to write logs to a non-SAM-QFS directory, such as `/var/adm/`. In the example, we specify `/var/adm/recycler.log`:

```
root@solaris:~# vi /etc/opt/SUNWsamfs/recycler.cmd
# Configuration file for SAM-QFS 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@solaris:~# vi /etc/opt/SUNWsamfs/recycler.cmd
# Configuration file for SAM-QFS 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@solaris:~# vi /etc/opt/SUNWsamfs/recycler.cmd
# Configuration file for SAM-QFS 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 SAM-QFS configuration. Then save the `recycler.cmd` file, and close the editor.

```
root@solaris:~# vi /etc/opt/SUNWsamfs/recycler.cmd
# Configuration file for SAM-QFS archiving file systems
#-----
logfile = /var/adm/recycler.log
no_recycle tp VOL[0-9][2-9][0-9]
library1 -hwm 95 -mingain 60
:wq
root@solaris:~#
```

7. If the `mcf` file for the archiving SAM-QFS 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 6-36
8. Otherwise, go to "[Configure File System Protection](#)" on page 6-38.

Catalog Archival Media Stored in a Network-Attached Tape Library

After you mount a file system, the SAM-QFS software creates catalogs for each automated library that is configured in the `mcf` file. However, if you have a network-attached automated library, you must populate the library's catalog.

The appropriate method to use to populate a library's catalog depends on the number of volumes that you include in the catalog.

To catalog a large number of volumes stored in one of these network-attached libraries, you run the SAM-QFS `build_cat` command against an input file that lists the volumes. Each entry in the file represents one volume using four, space-delimited fields:

Proceed as follows:

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


```
root@solaris:~#
```
2. If the archiving file system uses an Oracle StorageTek ACSLS-attached tape library, draw the required SAM-QFS archival media from the library's scratch pool and

generate the catalog automatically. Use the `samimport -c volumes -s pool` command, *volumes* is the number of volumes needed and *pool* is the name of the scratch media pool defined for the library. Stop here.

In the example, we request 20 tape volumes drawn from the pool called `scratch`:

```
root@solaris:~# samimport -c 20 -s scratch
```

3. If the archiving file system uses an IBM 3494 network-attached 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. Stop here.

The IBM 3494 library is configured as a single logical library when the Additional Parameters field of the `mcf` file specifies `access=private`. If `access=shared`, the IBM 3494 library is divided into multiple logical libraries, and you must use the method specified below.

4. Otherwise, if the archiving file system uses a shared IBM 3494 network-attached library or any other network-attached library, create a catalog input file using a text editor.

In the example, we use the `vi` editor to create the file `inputs18500cat`:

```
root@solaris:~# vi inputs18500cat
~
"~/inputs18500cat" [New File]
```

5. 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 SAM-QFS catalog. In the example, this is the first record, so we enter 0:

```
0
~
"~/inputs18500cat" [New File]
```

6. 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 01"`. In this example, the VSN of the first volume does not contain spaces:

```
0 VOL001
~
"~/s18500catinput" [New File]
```

7. 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
~
"~/s18500catinput" [New File]
```

8. 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 `ti` (for StorageTek T10000 media) or `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 Oracle StorageTek SL8500 ACSLS-attached tape library with StorageTek T10000 tape drives, so we enter `ti`:

```
0 VOL001 VOL001 ti
~
"~/sl8500catinput" [New File]
```

9. Repeat steps 3-6 to create additional records for each of the volumes that you intend to use with SAM-QFS. Then save the file.

```
0 VOL001 VOL001 ti
1 VOL002 VOL002 ti
...
13 VOL014 VOL014 ti
:wq
root@solaris:~#
```

10. 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 SAM-QFS 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 `SL8500`:

```
root@solaris:~# build_cat input_vsns /var/opt/SUNWsamfs/catalog/SL8500
```

11. 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.

12. 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 a SAM-QFS archiving file system, file data is automatically protected by the archiver: modified files are automatically copied to archival storage media, such as tape. But if you backed up only your files and then suffered an unrecoverable failure in a disk device or RAID group, you would have the data but no easy way to use it. You would have to create a substitute file system, identify each file, determine its proper location within the new file system, ingest it, and recreate lost relationships between it and users, applications, and other files. This kind of recovery is, at best, a daunting and long drawn-out process.

So, for fast, efficient recovery, you have to actively protect the file-system metadata that make files and archives copies usable. You must back up directory paths, inodes, access controls, symbolic links, and pointers to copies archived on removable media.

You protect SAM-QFS 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 a SAM-QFS 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 SAM-QFS *tar* utilities. However, this information is volatile. Like most system logs, the archiver log grows rapidly and must thus be overwritten frequently. If you do not make regular copies to compliment your recovery points, you will not have log information when you need it.

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. The remainder of this section provides instructions for creating and managing recovery points. It contains the following subsections:

- [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@solaris:~#
```

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 `/samms`. So we have created the directory `/zfs1/samms_recovery` on the independent file system `/zfs1`:

```
root@solaris:~# mkdir /zfs1/samms_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 `/samms`. So we create the directory `/var/samms_archlogs`:

```
root@solaris:~# mkdir /var/samms_archlogs
```

7. Next, [Automatically Create Recovery Points and Save 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 SAM-QFS File System 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 *StorageTek Storage Archive Manager and StorageTek QFS Software File System Manager User's Guide*.

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 archiving log. For each archiving file system that you have configured, proceed as follows:

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

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

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@solaris:~# 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. 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.
- *hour minutes* 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:

- ((the 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 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/sam1_recovery` for any files that have not been modified for 72 hours (3 days) or more and deletes any it finds:

```
# 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/sam1_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 `;` (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 `/sam1` directory. Note that the crontab entry is still a single line, even though the line appears to wrap to fit the format of this book (the apparent end of the line has been escaped by a back slash):

```
# 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/sam1_recovery -type f -mtime +72 -print | \
xargs -l1 rm -f; cd /sam1;
```

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 directory` is the `samfsdump` command option that specifies the location where the recovery point file will be saved and `directory` is the directory that we created to hold recovery points for this file system.
 - `'date +%y%m%d'` uses the Solaris `date` command with the formatting template `+%y%m%d` to create 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, 140122, January 22, 2014).
 - `;` (semicolon) marks the end of the command line.
 - `)` (the 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/sam1_recovery`. Note that the crontab entry is still a single line, even

though the line appears to wrap to fit the format of this book (the apparent ends of lines have been escaped by back slashes):

```
# 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/saml_recovery -type f -mtime +72 -print | \
xargs -l1 rm -f; cd /saml ; /opt/SUNWsamfs/sbin/samfsdump \
-f /zfs1/saml_recovery/'date +%y%m%d'
```

7. 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.
- *hour minutes* 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/saml_recovery -type f -mtime +72 -print | \
xargs -l1 rm -f; cd /saml ; /opt/SUNWsamfs/sbin/samfsdump \
-f /zfs1/saml_recovery/'date +%y%m%d'
```

8. Continuing on the same line, enter a shell command that moves the current archiver log to a location and gives it a unique name. Enter the text (`mv /var/adm/saml.archive.log /var/saml_archlogs/"date +%y%m%d";`.

This step saves log entries that would be overwritten if left in the active log file.

```
...
30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh
10 2 * * * ( find /saml_recovery/dumps -type f -mtime +72 -print | xargs -l1 rm
-f; \ cd /saml ; /opt/SUNWsamfs/sbin/samfsdump -f /zfs1/saml_recovery/'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/saml.archive.log`).

In the example, note that the crontab entry is still a single line, even though the line appears to wrap to fit the format of this book (the apparent end of the line has been escaped by a back slash):

```
...
30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh
10 2 * * * ( find /saml_recovery/dumps -type f -mtime +72 -print | xargs -l1 rm
-f; \ cd /saml ; /opt/SUNWsamfs/sbin/samfsdump -f /zfs1/saml_recovery/'date
+%y%m%d'
```

```
touch /var/adm/sam1.archive.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 /sam1_recovery/dumps -type f -mtime +72 -print | xargs -l1 rm
-f; \ cd /sam1 ; /opt/SUNWsamfs/sbin/samfsdump -f /zfs1/sam1_recovery/'date
+%\%y%\%m%\%d' )
15 3 * * 0 ( mv /var/adm/sam1.archive.log /var/sam1_archlogs/"date +%\%y%\%m%\%d"; \
touch /var/adm/sam1.archive.log )
:wq
root@solaris:~#
```

Where to go from here:

- If you need to be able to verify the data integrity of archival tape volumes, go to ["Configure Archival Media Validation"](#) on page 6-44.
- 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 6-51.
- 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 6-54
- If you have additional requirements, such as multiple-host file-system access or high-availability configurations, see ["Beyond the Basics"](#) on page 6-55.
- 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, and 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.

SAM-QFS supports media validation in two ways:

- You can [Configure SAM-QFS to Support Data Integrity Validation](#) to validate data on StorageTek T10000 tape media, either manually or automatically under SAM-QFS Periodic Media Verification.
- You can also [Configure SAM-QFS Periodic Media Verification](#) to automatically validate data on both StorageTek T10000 tape media and other formats, such as LTO Ultrium.

Configure SAM-QFS to Support Data Integrity Validation

Data Integrity Validation is a feature of Oracle StorageTek tape drives that works with the SAM-QFS 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 SAM-QFS `tpverify` (tape-verify) command to perform this step on demand.

To configure Data Integrity Validation, proceed as follows:

1. Log in to the SAM-QFS server as `root`.

In the example, the metadata server is named `samfs-mds`:

```
[samfs-mds]root@solaris:~#
```

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

```
[samfs-mds]root@solaris:~# uname -r
5.11
[samfs-mds]root@solaris:~#
```

3. Make sure that the archival storage equipment defined in the SAM-QFS `mcf` file includes compatible tape drives: StorageTek T10000C (minimum firmware level 1.53.315) or T10000D.

4. If any SAM-QFS operations are currently running, stop them using the `samd stop` command.

```
[samfs-mds]root@solaris:~# samd stop
```

5. 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:

```
[samfs-mds]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.
...
div = off
```

6. 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 SAM-QFS archiver software will not verify complete file copies after they are archived.

```
[samfs-mds]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.
```

```
...
div = on
:wq
[samfs-mds]root@solaris:~#
```

7. 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 rereads the newly stored data and checksums, recalculates, and, compares the stored and calculated results.

```
[samfs-mds]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.
...
div = verify
#narcopy = true
:wq
[samfs-mds]root@solaris:~#
```

8. Tell the SAM-QFS software to reread the `defaults.conf` file and reconfigure itself accordingly. Use the `samd config` command.

```
[samfs-mds]root@solaris:~# /opt/SUNWsamfs/sbin/samd config
```

9. If you stopped SAM-QFS operations in an earlier step, restart them now using the `samd start` command.

```
[samfs-mds]root@solaris:~# samd start
```

10. Data Integrity Validation is now configured.

Where to go from here:

- If you need to automate data integrity validation, go to "[Configure SAM-QFS Periodic Media Verification](#)" on page 6-46.
- 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 6-51.
- 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 6-54
- If you have additional requirements, such as multiple-host file-system access or high-availability configurations, see "[Beyond the Basics](#)" on page 6-55.

Configure SAM-QFS Periodic Media Verification

Starting with Release 5.4, you can set up Periodic Media Verification (PMV) for SAM-QFS 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 a SAM-QFS 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 SAM-QFS takes when errors are detected. SAM-QFS can, for example, automatically rearchive files that contain errors and/or recycle tape volumes that contain errors.

1. If you have not already done so, [Configure SAM-QFS to Support Data Integrity Validation](#).

2. Log in to the SAM-QFS server as `root`.

In the example, the metadata server is named `samfs-mds`:

```
[samfs-mds]root@solaris:~#
```

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

```
[samfs-mds]root@solaris:~# uname -r
5.11
[samfs-mds]root@solaris:~#
```

4. Open the `/etc/opt/SUNWsamfs/verifyd.cmd` file in a text editor.

In the example, we open the file in the `vi` editor:

```
[samfs-mds]root@solaris:~# vi /etc/opt/SUNWsamfs/verifyd.cmd
```

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:

```
# Enable SAM-QFS 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`:

```
pmv = on
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 SAM-QFS to Support Data Integrity Validation](#)" on page 6-44).

- 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`:

```
pmv = on
run_time = always
pmv_method = legacy
```

8. To use all available libraries and drives for verification, enter the line `pmv_scan = all`.

```
pmv = on
run_time = always
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.

```
pmv = on
run_time = always
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:

```
pmv = on
run_time = always
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:

```

pmv = on
run_time = always
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:

```

pmv = on
run_time = always
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`.

```

pmv = on
run_time = always
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 SAM-QFS to flag the media for recycling after 10 errors have been detected:

```

pmv = on
run_time = always
pmv_method = legacy
pmv_scan = disable
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 *SECONDSs*, *MINUTESm*, *HOURLh*, *DAYSd*, and/or *YEARSy* 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:

```

pmv = on
run_time = always
pmv_method = legacy
pmv_scan = disable
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`.

```

pmv = on
run_time = always
pmv_method = legacy

```

```
pmv_scan = disable
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:

```
pmv = on
run_time = always
pmv_method = legacy
pmv_scan = all
pmv_vsns = ^VOL0[01][0-9] ^VOL23[0-9] VOL400
```

SAM-QFS 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-SAM-QFS) 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. SAM-QFS 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. So define the desired verification policy. Enter the line `pmv_policy = verified age vertime` with, optionally, the parameters `modified age modtime`, and/or `mounted age mnttime`, where *vertime*, *modtime*, and *mnttime* are lists of value-unit pairs where the values are non-negative integers and the units:

- *vertime*, *modtime*, and *mnttime* are lists of value-unit pairs
- The values are non-negative integers.
- The units are *y* (years), *m* (months), *d* (days), *H* (hours), *M* (minutes), and *S* (seconds).

The `verified age` parameter specifies the minimum time that must have passed since the volume was last verified. The `modified age` parameter specifies the minimum time that must have passed since the volume was last modified. The `mounted age` parameter specifies the minimum time that must have passed since the volume was last 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:

```
pmv = on
run_time = always
pmv_method = legacy
pmv_scan = all
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.
20. Check the `verifyd.cmd` file for errors by running the `sam-fsd` command. Correct any errors found.

The `sam-fsd` command reads SAM-QFS configuration files and initializes file systems. It will stop if it encounters an error:

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

You have finished configuring periodic media verification.

21. 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 6-51.
22. 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 6-54.
23. If you have additional requirements, such as multiple-host file-system access or high-availability configurations, see ["Beyond the Basics"](#) on page 6-55.
24. 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 SAM-QFS 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 relax 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 a SAM-QFS 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 SAM-QFS 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.
- `YEARSyDAYSDhOURShMINUTESm` 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 SAM-QFS 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 6-54
11. If you have additional requirements, such as multiple-host file-system access or high-availability configurations, see ["Beyond the Basics"](#) on page 6-55.
12. Otherwise, go to ["Configuring Notifications and Logging"](#) on page 11-1.

Enabling Support for the Linear Tape File System (LTFS)

Starting with Release 5.4, SAM-QFS 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 SAM-QFS sites, when typical wide-area network (WAN) connections are too slow or too expensive for the task.

For information on using and administering LTFS volumes, see the `samltps` man page and the *StorageTek Storage Archive Manager and QFS Software Maintenance and Administration Guide*.

To enable SAM-QFS LTFS support, proceed as follows:

1. Log in to the SAM-QFS metadata server as `root`.

```
[samfs-mds]root@solaris:~#
```

2. If any SAM-QFS operations are currently running, stop them using the `samd stop` command.

```
[samfs-mds]root@solaris:~# samd stop
```

3. Open the `/etc/opt/SUNWsamfs/defaults.conf` in a text editor.

In the example, we open the file in the `vi` editor:

```
[samfs-mds]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.
...
```

4. In the `defaults.conf` file, add the line `ltfs = mountpoint workers volumes`, where `mount point` 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 and `volumes` is an optional maximum number of tape volumes per drive. Then save the file, and close the editor.

In the example, we specify the LTFS mount point `s /mnt/ltfs` and accept the defaults for the other parameters:

```
[samfs-mds]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.
...
ltfs = /mnt/ltfs
:wq
[samfs-mds]root@solaris:~#
```

5. Tell the SAM-QFS software to reread the `defaults.conf` file and reconfigure itself accordingly. Correct any errors reported and repeat as necessary.

```
[samfs-mds]root@solaris:~# /opt/SUNWsamfs/sbin/samd config
```

6. If you stopped SAM-QFS operations in an earlier step, restart them now using the `samd start` command.

```
[samfs-mds]root@solaris:~# samd start
```

7. SAM-QFS 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 6-55.

8. Otherwise, go to ["Configuring Notifications and Logging"](#) on page 11-1.

Beyond the Basics

This completes basic installation and configuration of SAM-QFS 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 SAM-QFS Software"](#) on page 7-1 and/or ["Accessing File Systems from Multiple Hosts Using NFS and SMB/CIFS"](#) on page 7-29.
- If you need to configure a high-availability QFS file system or SAM-QFS archiving file system, see ["Preparing High-Availability Solutions"](#) on page 9-1.
- If you need to configure a SAM-QFS archiving file system to share archival storage hosted at a remote location, see ["Configuring SAM-Remote"](#) on page 8-1.
- If you plan on using the sideband database feature, go to ["Configuring the SAM-QFS Reporting Database"](#) on page 10-1.
- Otherwise, go to ["Configuring Notifications and Logging"](#) on page 11-1.

Accessing File Systems from Multiple Hosts

SAM-QFS 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:

- [Accessing File Systems from Multiple Hosts Using SAM-QFS Software](#)
- [Accessing File Systems from Multiple Hosts Using NFS and SMB/CIFS](#)

Accessing File Systems from Multiple Hosts Using SAM-QFS Software

SAM-QFS 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.

SAM-QFS 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.

You cannot implement a SAM-QFS shared file system in a segmented-file environment.

- Mandatory locks are not supported.

An EACCES error is returned if the mandatory lock is set. Advisory locks are supported, however. For more information about advisory locks, see the `fcntl` man page.

SAM-QFS software hosts can access file system data in either of two ways:

- by [Configuring a SAM-QFS Single-Writer, Multiple-Reader File System](#)
- by [Configuring a SAM-QFS Shared File System](#).

Each has 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.

Configuring a SAM-QFS 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.

```
[swriterfs1-mds]root@solaris:~#
```

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 a General-Purpose `ms` File System](#) or [Configure a 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 `swriterfs1-writer` using the `vi` text editor. The example specifies an `ma` file system with the equipment identifier and family set name `swriterfs1` and the equipment ordinal number 300:

```
[swriterfs1-writer]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment      Equipment  Equipment  Family    Device  Additional
# Identifier      Ordinal    Type       Set       State   Parameters
#-----
swriterfs1      300       ma        swriterfs1 on
/dev/dsk/c0t0d0s0 301       mm        swriterfs1 on
/dev/dsk/c0t3d0s0 302       mr        swriterfs1 on
/dev/dsk/c0t3d0s1 303       mr        swriterfs1 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
#-----
swriterfs1      300       ma        swriterfs1 on
/dev/dsk/c0t0d0s0 301       mm        swriterfs1 on
/dev/dsk/c0t3d0s0 302       mr        swriterfs1 on
/dev/dsk/c0t3d0s1 303       mr        swriterfs1 on
```

```
:wq
[swriterfs1-writer]root@solaris:~#
```

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

The `sam-fsd` command reads SAM-QFS configuration files and initializes file systems. It will stop if it encounters an error:

```
[swriterfs1-mds]root@solaris:~# sam-fsd
```

5. Tell the SAM-QFS service to re-read the `mcf` file and reconfigure itself accordingly. Use the command `samd config`.

```
[swriterfs1-writer]root@solaris:~# samd config
```

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 6-7.

In the example, the command creates the single-writer/multi-reader file system `swriterfs1`:

```
[swriterfs1-writer]root@solaris:~# sammkfs swriterfs1
Building 'swriterfs1' 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.

```
[swriterfs1-writer]root@solaris:~# cp /etc/vfstab /etc/vfstab.backup
```

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 6-7.

In the example, we open the `/etc/vfstab` file in the `vi` text editor and add a line for the `swriterfs1` family set device:

```
[swriterfs1-writer]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      -
...
swriterfs1 -          /swriterfs1 samfs -     no
```

9. In the `Mount Options` column of the `/etc/vfstab` file, enter the `writer` mount option.

Caution: Allowing more than one host to mount a multiple-reader, single-writer file system using the `writer` option can corrupt the file system! You must make sure that only one host is the `writer` at any given time!

```
#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 -
...
swriterfs1 - /swriterfs1 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 automatically at boot time, enter `yes` in the Mount at Boot field. 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 -
...
swriterfs1 - /swriterfs1 samfs - yes 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 -
...
swriterfs1 - /swriterfs1 samfs - yes writer, bg
:wq
[swriterfs1-writer]root@solaris:~#
```

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 `/swriterfs1` mount-point directory and set permissions to 755 (`-rwxr-xr-x`):

```
[swriterfs1-writer]root@solaris:~# mkdir /swriterfs1
[swriterfs1-writer]root@solaris:~# chmod 755 /swriterfs1
```

13. Mount the new file system:

```
[swriterfs1-writer]root@solaris:~# mount /swriterfs1
```

14. Once the shared file system has been created, [Configure the Readers](#).

Configure the Readers

For each host that you are configuring as a reader (a host that mounts the file system read-only), proceed as follows:

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

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 a SAM-QFS 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 `swriterfs1-writer`, except that the paths to the devices are specified starting from the host `swriterfs1-reader1`:

```
[swriterfs1-reader1]root@solaris:~# samfsconfig /dev/dsk/*
# Family Set 'swriterfs1' Created Thu Nov 21 07:17:00 2013
# Generation 0 Eq count 4 Eq meta count 1
#
sharefs          300      ma      sharefs  -
/dev/dsk/c1t0d0s0 301      mm      sharefs  -
/dev/dsk/c1t3d0s0 302      mr      sharefs  -
/dev/dsk/c1t3d0s1 303      mr      sharefs  -
```

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 use the `samd buildmcf` command to run `samfsconfig` and create the client `mcf` file automatically.

In the example, the `mcf` file for the host, `swriterfs1-reader1` looks like this once we add the commented out column headings:

```
[swriterfs1-reader1]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment      Equipment  Equipment  Family    Device    Additional
# Identifier      Ordinal   Type       Set       State     Parameters
#-----
sharefs          300      ma      sharefs  -
/dev/dsk/c1t0d0s0 301      mm      sharefs  -
/dev/dsk/c1t3d0s0 302      mr      sharefs  -
/dev/dsk/c1t3d0s1 303      mr      sharefs  -
```

4. Make sure that the `Device State` field is set to `on` for all devices. Then save the `mcf` file.

```
[swriterfs1-reader1]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment      Equipment  Equipment  Family    Device    Additional
# Identifier      Ordinal   Type       Set       State     Parameters
#-----
sharefs          300      ma      sharefs  on
/dev/dsk/c1t0d0s0 301      mm      sharefs  on
/dev/dsk/c1t3d0s0 302      mr      sharefs  on
/dev/dsk/c1t3d0s1 303      mr      sharefs  on
:wq
[swriterfs1-reader1]root@solaris:~#
```

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

The `sam-fsd` command reads SAM-QFS configuration files and initializes file systems. It will stop if it encounters an error:

```
[swriterfs1-reader1]root@solaris:~# sam-fsd
```

6. Back up the operating system's `/etc/vfstab` file.

```
[swriterfs1-reader1]root@solaris:~# cp /etc/vfstab /etc/vfstab.backup
```

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 `swriterfs1` family set device:

```
[swriterfs1-reader1]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    -      -
...
swriterfs1 -      /swriterfs1 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    -      -
...
swriterfs1 -      /swriterfs1 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    -      -
...
swriterfs1 -      /swriterfs1 samfs -       yes    writer,bg
:wq
[swriterfs1-reader1]root@solaris:~#
```

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 `/swriterfs1`

mount-point directory and set permissions to 755 (-rwxr-xr-x), just as we did on the writer host:

```
[swriterfs1-reader1]root@solaris:~# mkdir /swriterfs1
[swriterfs1-reader1]root@solaris:~# chmod 755 /swriterfs1
```

11. Mount the new file system:

```
[swriterfs1-reader1]root@solaris:~# mount /swriterfs1
```

12. Repeat this procedure until all reader hosts have been configured to mount the file system read-only.
13. Stop here. You have configured the SAM-QFS multiple-reader, single-writer file system.

Configuring a SAM-QFS Shared File System

SAM-QFS shared file systems give multiple SAM-QFS 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:

- [Configuring a File System Metadata Server for Sharing](#)
- [Configuring File System Clients for Sharing](#)
- [Configuring Archival Storage for a Shared File System](#)

Configuring a File System Metadata Server for Sharing

To configure a metadata server to support a shared file system, carry out the tasks listed below:

- [Create a Hosts File on the Active Metadata Server](#)
- [Create the Shared File System on the Active Server](#)
- [Mount the Shared File System on the Active Server](#)

Create a Hosts File on the Active Metadata Server On the active metadata server, you must create a hosts file that lists network address information for the servers and clients of a shared file system. The hosts file is stored alongside the `mcf` file in the `/etc/opt/SUNWsamfs/` directory. During the initial creation of a shared file system, the `sammkfs -S` command configures sharing using the settings stored in this file. So create it now, using the procedure below.

1. Log in to the server as `root`.

```
[sharefs-mds]root@solaris:~#
```

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.sharefs` using the `vi` text editor. We add some optional headings, starting each line with a hash sign (`#`), indicating a comment:

```
[sharefs-mds]root@solaris:~# vi /etc/opt/SUNWsamfs/hosts.sharefs
# /etc/opt/SUNWsamfs/hosts.sharefs
#
#Host Name          Network Interface      Server  On/  Additional
#-----          -----
#-----          -----          -----  ---  -----
```

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.sharefs
#
#Host Name          Network Interface      Server  On/  Additional
#-----          -----
#-----          -----          -----  ---  -----
sharefs-mds      10.79.213.117
```

4. Add a third column, separated from the network address by whitespace characters. In this column, enter the ordinal number of the server (1 for the active metadata server, 2 for the first potential metadata server, and so on).

In this example, there is only one metadata server, so we enter 1:

```
# /etc/opt/SUNWsamfs/hosts.sharefs
#
#Host Name          Network Interface      Server  On/  Additional
#-----          -----
#-----          -----          -----  ---  -----
sharefs-mds        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) marks the host 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).

```
# /etc/opt/SUNWsamfs/hosts.sharefs
#
#Host Name          Network Interface      Server  On/  Additional
#-----          -----
#-----          -----          -----  ---  -----
sharefs-mds        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.sharefs
#
#Host Name          Network Interface      Server  On/  Additional
#-----          -----
#-----          -----          -----  ---  -----
sharefs-mds        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 `sharefs-mds_alt` is a potential metadata server with the server ordinal 2. Until and unless we activate it as a metadata server, it will be a client reader:

```
# /etc/opt/SUNWsamfs/hosts.sharefs
#
#Host Name          Network Interface      Server  On/  Additional
#-----          -
sharefs-mds         10.79.213.117         1      0   server
sharefs-mds_alt     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, `sharefs-client1` and `sharefs-client2`.

```
# /etc/opt/SUNWsamfs/hosts.sharefs
#
#Host Name          Network Interface      Server  On/  Additional
#-----          -
sharefs-mds         10.79.213.117         1      0   server
sharefs-mds_alt     10.79.213.217         2      0
sharefs-client1    10.79.213.133         0     0
sharefs-client2    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.sharefs` and exit the vi editor:

```
# /etc/opt/SUNWsamfs/hosts.sharefs
#
#Host Name          Network Interface      Server  On/  Additional
#-----          -
sharefs-mds         10.79.213.117         1      0   server
sharefs-mds_alt     10.79.213.217         2      0
sharefs-client1     10.79.213.133         0      0
sharefs-client2     10.79.213.147         0      0
:wq
[sharefs-mds]root@solaris:~#
```

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 Server](#).

Create the Shared File System on the Active Server Proceed as follows:

1. Log in to the server as root.

```
[sharefs-mds]root@solaris:~#
```

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 either [Configure a General-Purpose ms File System](#) or [Configure a High-Performance ma File System](#).

In the example below, we edit the `mcf` file on the host `sharefs-mds` using the vi text editor. The example specifies an `ma` file system with the equipment identifier and family set name `sharefs` and the equipment ordinal number 300:

```
[sharefs-mds]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment          Equipment  Equipment  Family    Device    Additional
```

# Identifier	Ordinal	Type	Set	State	Parameters
sharefs	300	ma	sharefs	on	
/dev/dsk/c0t0d0s0	301	mm	sharefs	on	
/dev/dsk/c0t3d0s0	302	mr	sharefs	on	
/dev/dsk/c0t3d0s1	303	mr	sharefs	on	

- 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
sharefs	300	ma	sharefs	on	shared
/dev/dsk/c0t0d0s0	301	mm	sharefs	on	
/dev/dsk/c0t3d0s0	302	mr	sharefs	on	
/dev/dsk/c0t3d0s1	303	mr	sharefs	on	

- Save the /etc/opt/SUNWsamfs/mcf file, and quit the editor.

In the example, we save the changes and exit the vi editor:

```
sharefs          300      ma      sharefs  on      shared
/dev/dsk/c0t0d0s0 301      mm      sharefs  on
/dev/dsk/c0t3d0s0 302      mr      sharefs  on
/dev/dsk/c0t3d0s1 303      mr      sharefs  on

:wq
[sharefs-mds]root@solaris:~#
```

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

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

```
[sharefs-mds]root@solaris:~# sam-fsd
```

- Tell the SAM-QFS service to reread the mcf file and reconfigure itself accordingly. Correct any errors reported and repeat as necessary.

```
[sharefs-mds]root@solaris:~# samd config
```

- 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 6-7.

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.sharefs file and creates the shared file system sharefs:

```
[sharefs-mds]root@solaris:~# sammkfs -S sharefs
Building 'sharefs' will destroy the contents of devices:
  /dev/dsk/c0t0d0s0
  /dev/dsk/c0t3d0s0
  /dev/dsk/c0t3d0s1
Do you wish to continue? [y/N]yes ...
```

- Next, [Mount the Shared File System on the Active Server](#).

Mount the Shared File System on the Active Server

1. Log in to the server as root.

```
[sharefs-mds]root@solaris:~#
```

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

```
[sharefs-mds]root@solaris:~# cp /etc/vfstab /etc/vfstab.backup
```

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 6-7.

In the example, we open the /etc/vfstab file in the vi text editor and add a line for the sharefs family set device:

```
[sharefs-mds]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    -      -
...
sharefs -      /sharefs samfs   -      yes
```

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    -      -
...
sharefs -      /sharefs samfs   -      yes    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    -      -
...
sharefs -      /sharefs samfs   -      yes    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    -      -
...
sharefs -      /sharefs samfs   -      no    shared, bg
:wq
[sharefs-mds]root@solaris:~#
```

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 `/sharefs` mount-point directory and set permissions to 755 (`-rwxr-xr-x`):

```
[sharefs-mds]root@solaris:~# mkdir /sharefs
[sharefs-mds]root@solaris:~# chmod 755 /sharefs
```

8. Mount the new file system:

```
[sharefs-mds]root@solaris:~# mount /sharefs
```

9. If your hosts are configured with multiple network interfaces, consider "[Use Local Hosts Files to Route Network Communications](#)" on page 7-19.
10. Otherwise, once the shared file system has been created on the metadata server, start "[Configuring File System Clients for Sharing](#)" on page 7-12.

Configuring File System Clients for Sharing

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.

```
[sharefs-client1]root@solaris:~#
```

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

```
[sharefs-client1]root@solaris:~# 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 `sharefs-client1` 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, `sharefs-mds`, only the host-assigned device controller numbers differ:

```
[sharefs-client1]root@solaris:~# samfsconfig /dev/dsk/*
# Family Set 'sharefs' Created Thu Feb 21 07:17:00 2013
# Generation 0 Eq count 4 Eq meta count 1
```

```
#
sharefs          300          ma          sharefs    -
/dev/dsk/c1t0d0s0  301          mm          sharefs    -
/dev/dsk/c1t3d0s0  302          mr          sharefs    -
/dev/dsk/c1t3d0s1  303          mr          sharefs    -
```

- 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 SAM-QFS 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 `sharefs-client2` has access to the data devices only. So the `samfsconfig` output looks like this:

```
[sharefs-client2]root@solaris:~# samfsconfig /dev/dsk/*
# Family Set 'sharefs' Created Thu Feb 21 07:17:00 2013
#
# Missing slices
# Ordinal 0
# /dev/dsk/c4t3d0s0  302          mr          sharefs    -
# /dev/dsk/c4t3d0s1  303          mr          sharefs    -
```

- 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, `sharefs-client1`, has access to the metadata devices for the file system, so the `mcf` file starts out looking like this:

```
[sharefs-client1]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment          Equipment Equipment Family   Device  Additional
# Identifier          Ordinal   Type     Set     State   Parameters
#-----
sharefs              300      ma      sharefs -
/dev/dsk/c1t0d0s0   301      mm      sharefs -
/dev/dsk/c1t3d0s0   302      mr      sharefs -
/dev/dsk/c1t3d0s1   303      mr      sharefs -
```

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

```
[sharefs-client2]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment          Equipment Equipment Family   Device  Additional
# Identifier          Ordinal   Type     Set     State   Parameters
#-----
# /dev/dsk/c4t3d0s0  302      mr      sharefs -
# /dev/dsk/c4t3d0s1  303      mr      sharefs -
```

- 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, `sharefs-client1`, has access to the metadata:

```
[sharefs-client1]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
# Equipment          Equipment Equipment Family   Device  Additional
# Identifier          Ordinal   Type     Set     State   Parameters
#-----
sharefs              300      ma      sharefs -      shared
/dev/dsk/c1t0d0s0   301      mm      sharefs -
```

- ```

/dev/dsk/c1t3d0s0 302 mr sharefs -
/dev/dsk/c1t3d0s1 303 mr sharefs -

```
- 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

```

[sharefs-client2]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
sharefs 300 ma sharefs - shared
/dev/dsk/c4t3d0s0 302 mr sharefs -
/dev/dsk/c4t3d0s1 303 mr sharefs -

```

- 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:

```

[sharefs-client2]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
sharefs 300 ma sharefs on shared
nodev 301 mm sharefs on
/dev/dsk/c4t3d0s0 302 mr sharefs -
/dev/dsk/c4t3d0s1 303 mr sharefs -

```

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

```

[sharefs-client2]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
sharefs 300 ma sharefs on shared
nodev 301 mm sharefs on
/dev/dsk/c4t3d0s0 302 mr sharefs -
/dev/dsk/c4t3d0s1 303 mr sharefs -

```

- 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, sharefs-client1, has access to the metadata devices for the file system, so the mcf file ends up looking like this:

```

[sharefs-client1]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
sharefs 300 ma sharefs on shared
/dev/dsk/c1t0d0s0 301 mm sharefs on
/dev/dsk/c1t3d0s0 302 mr sharefs on
/dev/dsk/c1t3d0s1 303 mr sharefs on
:wq
[sharefs-client1]root@solaris:~#

```

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

```

[sharefs-client2]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters

```

```
#-----
sharefs 300 ma sharefs on shared
nodev 301 mm sharefs on
/dev/dsk/c4t3d0s0 302 mr sharefs on
/dev/dsk/c4t3d0s1 303 mr sharefs on
:wq
[sharefs-client2]root@solaris:~#
```

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

The `sam-fsd` command reads SAM-QFS configuration files and initializes file systems. It will stop if it encounters an error. In the example, we check the `mcf` file on `sharefs-client1`:

```
[sharefs-client1]root@solaris:~# sam-fsd
```

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.

```
[sharefs-client1]root@solaris:~#
```

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

```
[sharefs-client1]root@solaris:~# cp /etc/vfstab /etc/vfstab.backup
```

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

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

```
[sharefs-client1]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 -
...
sharefs - /sharefs 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 -
...
sharefs - /sharefs samfs - no -
:wq
[sharefs-client1]root@solaris:~#
```

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 `/sharefs` mount-point directory and set permissions to 755

(`-rwxr-xr-x`):

```
[sharefs-client1]root@solaris:~# mkdir /sharefs
[sharefs-client1]root@solaris:~# chmod 755 /sharefs
```

6. Mount the shared file system:

```
[sharefs-client1]root@solaris:~# mount /sharefs
```

7. If the shared file system includes Linux clients, [Create the Shared File System on the Linux Clients](#).
8. If you are configuring a SAM-QFS shared archiving file system, go to your next task, "[Configuring Archival Storage for a Shared File System](#)" on page 7-21.
9. Otherwise, stop here. You have configured the SAM-QFS 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`.

```
[sharefs-clientL][root@linux ~]#
```

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

Since Linux hosts do not have access to the metadata devices for the file system, the `samfsconfig` cannot find the metadata devices and thus cannot fit the SAM-QFS 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 `sharefs-clientL` looks like this:

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

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, `sharefs-clientL`, starts out looking like this:

```
[sharefs-clientL][root@linux ~]# vi /etc/opt/SUNWsamfs/mcf
Equipment Equipment Equipment Family Device Additional
```

```
Identifier Ordinal Type Set State Parameters
#-----
/dev/sda4 302 mr sharefs -
/dev/sda5 303 mr sharefs -
```

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

```
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
sharefs 300 ma sharefs - shared
/dev/sda4 302 mr sharefs -
/dev/sda5 303 mr sharefs -
```

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

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

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

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

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

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

```
:wq
[sharefs-clientL][root@linux ~]#
```

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

The `sam-fsd` command reads SAM-QFS 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, `sharefs-clientL`:

```
[sharefs-clientL][root@linux ~]# sam-fsd
```

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.

```
[sharefs-clientL][root@linux ~]#
```

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

```
[sharefs-clientL][root@linux ~]# 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 sharefs-clientL, we open the file in the vi text editor and add a line for the sharefs family set device:

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

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

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

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

Linux clients support the following additional mount options:

- rw, ro
- retry
- meta\_timeo
- rdlease, wrlease, aplease
- minallocsz, maxallocsz
- noauto, auto

In the example, we add the option noauto:

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

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

```
#File
#Device Mount System Mount Dump Pass
```

```
#to Mount Point Type Options Frequency Number
#-----
...
/proc /proc proc defaults
sharefs /sharefs samfs shared,noauto 0 0
:wq
[sharefs-clientL][root@linux ~]#
```

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 `/sharefs` mount-point directory and set permissions to 755 (-rwxr-xr-x):

```
[sharefs-clientL][root@linux ~]# mkdir /sharefs
[sharefs-clientL][root@linux ~]# chmod 755 /sharefs
```

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:

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

9. If you are configuring a SAM-QFS shared archiving file system, go to your next task, "[Configuring Archival Storage for a Shared File System](#)" on page 7-21
10. Otherwise, stop here. You have configured the SAM-QFS shared file system.

**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 the Active Metadata Server](#)" on page 7-7). But local hosts files can be useful when you need to selectively route network traffic between file-system hosts that have multiple network interfaces.

Each file-system host identifies the network interfaces for the other hosts by first checking the `/etc/opt/SUNWsamfs/hosts.family-set-name` file on the metadata server. Then it checks for its own, specific `/etc/opt/SUNWsamfs/hosts.family-set-name.local` file. If there is no local hosts file, the host uses the interface addresses specified in the global hosts file in the order specified in the global file. But if there is a local hosts file, the host compares it with the global file and uses only those interfaces that are listed in both files in the order specified in the local file. By using different addresses in each file, you can thus control the interfaces used by different hosts.

To configure local hosts files, use the procedure outlined below:

1. On the metadata server host and each potential metadata server host, create a copy of the file system's global hosts file, `/etc/opt/SUNWsamfs/hosts.family-set-name`, as described in "[Create a Hosts File on the Active Metadata Server](#)" on page 7-7.

For the examples in this section, the shared file system, `sharefs2nic`, includes an active metadata server, `sharefs2-mds`, and a potential metadata server, `sharefs2-mds_alt`, each with two network interfaces. There are also two clients, `sharefs2-client1` and `sharefs2-client2`.

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 `/etc/opt/SUNWsamfs/hosts.sharefs2`, the file system's global host file, specifies a private network address in the Network Interface field of the entries for the active and potential servers and a hostname for the interface address of each client:

```
[sharefs2-mds]root@solaris:~# vi /etc/opt/SUNWsamfs/hosts.sharefs2
/etc/opt/SUNWsamfs/hosts.sharefs2
#
#Host Name Network Interface Server On/ Additional
#-----
sharefs2-mds 172.16.0.129 1 0 server
sharefs2-mds_alt 172.16.0.130 2 0
sharefs2-client1 sharefs2-client1 0 0
sharefs2-client2 sharefs2-client2 0 0
:wq
[sharefs2-mds]root@solaris:~#
```

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.sharefs2.local`, lists only private addresses for active and potential servers:

```
[sharefs2-mds]root@solaris:~# vi /etc/opt/SUNWsamfs/hosts.sharefs2.local
/etc/opt/SUNWsamfs/hosts.sharefs2
#
#Host Name Network Interface Server On/ Additional
#-----
sharefs2-mds 172.16.0.129 1 0 server
sharefs2-mds_alt 172.16.0.130 2 0
:wq
[sharefs2-mds]root@solaris:~# ssh root@sharefs2-mds_alt
Password:
[sharefs2-mds_alt]root@solaris:~# vi /etc/opt/SUNWsamfs/hosts.sharefs2.local
/etc/opt/SUNWsamfs/hosts.sharefs2.local
#
#Host Name Network Interface Server On/ Additional
#-----
sharefs2-mds 172.16.0.129 1 0 server
sharefs2-mds_alt 172.16.0.130 2 0
:wq
[sharefs2-mds_alt]root@solaris:~# exit
[sharefs2-mds]root@solaris:~#
```

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 file includes only the hostnames of the active and potential metadata servers:

```
[sharefs2-mds]root@solaris:~# ssh root@sharefs2-client1
Password:
[sharefs2-client1]root@solaris:~# vi /etc/opt/SUNWsamfs/hosts.sharefs2.local
/etc/opt/SUNWsamfs/hosts.sharefs2.local
#
Server On/ Additional
#Host Name Network Interface Ordinal Off Parameters
#-----
sharefs2-mds sharefs2-mds 1 0 server
sharefs2-mds_alt sharefs2-mds_alt 2 0
[sharefs2-client1]root@solaris:~# exit
[sharefs2-mds]root@solaris:~# ssh root@sharefs2-client2
Password:
[sharefs2-client2]root@solaris:~# vi /etc/opt/SUNWsamfs/hosts.sharefs2.local
/etc/opt/SUNWsamfs/hosts.sharefs2.local
#
Server On/ Additional
#Host Name Network Interface Ordinal Off Parameters
#-----
sharefs2-mds sharefs2-mds 1 0 server
sharefs2-mds_alt sharefs2-mds_alt 2 0
[sharefs2-client2]root@solaris:~# exit
[sharefs2-mds]root@solaris:~#
```

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 7-10.
5. If you started this procedure while configuring a client, you should now ["Mount the Shared File System on the Solaris Clients"](#) on page 7-15.

### Configuring Archival Storage for a Shared File System

To set up the archival storage for an archiving SAM-QFS shared file system, carry out the following tasks:

- [Connect Tape Drives to Server and Datamover Hosts Using Persistent Bindings](#)
- [Configure the Hosts of an Archiving File System to 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 (or full information on creating persistent bindings, see the `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.

```
[sharefs-mds]root@solaris:~#
```

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 3-4.

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 test editor.

In the example, we use the vi editor:

```
[sharefs-mds]root@solaris:~# 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. The line should be 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 the device's position in the Solaris device tree. Choose a node number that is high enough to avoid conflicts with any devices that Solaris configures automatically (Solaris starts from node 0).

In the example, we note the device address for the first device in the library, w500104f0008120fe, and see that the device is currently attached to the host at rmt/1:

```
[sharefs-mds] 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 rmt/60 to the number 1 drive in the library, w500104f0008120fe:

```
[sharefs-mds]root@solaris:~# 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
```

- Continue to add lines to the `/etc/devlink.tab` file for each tape device that is assigned for SAM-QFS 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`:

```
[sharefs-mds]root@solaris:~# 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 next three Solaris device nodes, maintaining the same order as in the library:

```
[sharefs-mds]root@solaris:~# 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
[sharefs-mds]root@solaris:~#
```

- Delete all existing links to the tape devices in `/dev/rmt`.

```
[sharefs-mds]root@solaris:~# rm /dev/rmt/*
```

- 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:

```
[sharefs-mds]root@solaris:~# devfsadm -c tape
```

- Add the same lines to the `/etc/devlink.tab` file, delete the links in `/dev/rmt`, and run `devfsadm -c tape` on each potential metadata server and datamover in the shared file system configuration.

In the example, we have a potential metadata server, `sharefs-mds_alt`, and a datamover client, `sharefs-client1`. So we edit the `/etc/devlink.tab` files on each to match that on the active server, `sharefs-mds`. Then we delete the existing links in `/dev/rmt` on `sharefs-mds_alt` and `sharefs-client1`, and run `devfsadm -c tape` on each:

```
[sharefs-mds_alt]root@solaris:~# 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
[sharefs-mds_alt]root@solaris:~# rm /dev/rmt/*
[sharefs-mds_alt]root@solaris:~# devfsadm -c tape
[sharefs-mds_alt]root@solaris:~# ssh sharefs-client1
```

```

Password:
[sharefs-client1]root@solaris:~# 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
[sharefs-client1]root@solaris:~# rm /dev/rmt/*
[sharefs-client1]root@solaris:~# devfsadm -c tape
[sharefs-client1]

```

9. Now, go to ["Configuring a File System Metadata Server for Sharing"](#) on page 7-7.

**Configure the Hosts of an 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.

```
[sharefs-host]root@solaris:~#
```

2. Open the /etc/opt/SUNWsamfs/mcf file in a text editor.

In the example, we use the vi editor.

```

[sharefs-host]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
sharefs 100 ms sharefs on
/dev/dsk/c1t3d0s3 101 md sharefs on
/dev/dsk/c1t3d0s4 102 md sharefs 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:

```

[sharefs-host]root@solaris:~# 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 library1:

```
Archival storage for copies:
#
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
/dev/scsi/changer/clt0d5 900 rb library1 on
```

- In the Additional Parameters column, enter the path where the library catalog will be stored.

Note that, due to document layout limitations, the example abbreviates the long path to the library catalog `var/opt/SUNWsamfs/catalog/library1cat`:

```
Archival storage for copies:
#
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
/dev/scsi/changer/clt0d5 900 rb library1 on .../library1cat
```

- Next, add an entry for each tape drive using the persistent equipment identifiers that we established in the procedure ["Connect Tape Drives to Server and Datamover Hosts Using Persistent Bindings"](#) on page 7-21.

```
Archival storage for copies:
#
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
DISKVOL1 800 ms DISKVOL1 on
/dev/dsk/c6t0d1s7 801 md DISKVOL1 on
/dev/dsk/c4t0d2s7 802 md DISKVOL1 on
/dev/scsi/changer/clt0d5 900 rb library1 on .../library1cat
/dev/rmt/60cbn 901 tp library1 on
/dev/rmt/61cbn 902 tp library1 on
/dev/rmt/62cbn 903 tp library1 on
/dev/rmt/63cbn 904 tp library1 on
```

- Finally, if you wish to configure a SAM-QFS 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/clt0d5 900 rb library1 on
...catalog/library1cat
/dev/rmt/60cbn 901 tp library1 on
/dev/rmt/61cbn 902 tp library1 on
/dev/rmt/62cbn 903 tp library1 on
/dev/rmt/63cbn 904 tp library1 on
historian 999 hy - - .../historian_cat
```

- Save the `mcf` file, and close the editor.

```

...
/dev/rmt/3cbn 904 tp library1 on
historian 999 hy - - ../historian_cat
:wq
[sharefs-host]root@solaris:~#

```

10. Check the `mcf` file for errors by running the `sam-fsd` command. Correct any errors found.

The `sam-fsd` command reads SAM-QFS configuration files and initializes file systems. It will stop if it encounters an error:

```
[sharefs-host]root@solaris:~# sam-fsd
```

11. Tell the SAM-QFS service to reread the `mcf` file and reconfigure itself accordingly. Correct any errors reported and repeat as necessary.

```
[sharefs-host]root@solaris:~# samd config
```

12. Repeat this procedure until all active and potential metadata servers and all datamover clients have been configured to use the archival storage.

**Distribute Tape I/O Across the Hosts of the Shared Archiving File System** Starting with SAM-QFS Release 5.4, 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 SAM-QFS implementations. As your archiving needs increase, you now have the option of either replacing SAM-QFS 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, for each client that will serve as a datamover, [Connect Tape Drives to Server and Datamover Hosts Using Persistent Bindings](#). 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 `samsharefs-mds`:

```
[samsharefs-mds]root@solaris:~#
```

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

```
[samsharefs-mds]root@solaris:~# uname -r
5.11
[samsharefs-mds]root@solaris:~#
```

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 `samsharefs-client1` and `samsharefs-client2` remotely using `ssh` and get the Solaris version from the log-in banner:

```
[samsharefs-mds]root@solaris:~# ssh root@samsharefs-client1
Password:
Oracle Corporation SunOS 5.11 11.1 September 2013
[samsharefs-client1]root@solaris:~# exit
```

```
[samsharefs-mds]root@solaris:~# ssh root@samsharefs-client2
Password:
Oracle Corporation SunOS 5.11 11.1 September 2013
[samsharefs-client2]root@solaris:~# exit
[samsharefs-mds]root@solaris:~#
```

6. On the metadata server, open the `/etc/opt/SUNWsamfs/defaults.conf` in a text editor. 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 open the file in the `vi` editor and add the line `distio = on`:

```
[samsharefs-mds]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.
...
distio = on
```

7. Next, select the device types that should participate in distributed I/O. To use device type `dev` with distributed I/O, add the line `dev_distio = on` to the `defaults.conf` file. To exclude device type `dev` from distributed I/O, add the line `dev_distio = off`. Save the file.

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 exclude LTO drives:

```
[samsharefs-mds]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.
...
distio = on
li_distio = off
:wq
[samsharefs-mds]root@solaris:~#
```

8. On each client that will serve as a datamover, edit the `defaults.conf` file so that it matches the file on the server.
9. 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 `samsharefs-client1`:

```
[samsharefs-client1]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
samsharefs 800 ms samsharefs on
...
Archival storage for copies:
/dev/rmt/60cbn 901 ti on
/dev/rmt/61cbn 902 ti on
/dev/rmt/62cbn 903 ti on
```

```

/dev/rmt/63cbn 904 ti on

```

10. 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 `samsharefs-client1`, so we use the family set name `library1` for the tape devices

```

[samsharefs-client1]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
samsharefs 800 ms samsharefs on
...
Archival storage for copies:
/dev/scsi/changer/clt0d5 900 rb library1 on .../library1cat
/dev/rmt/60cbn 901 ti library1 on
/dev/rmt/61cbn 902 ti library1 on
/dev/rmt/62cbn 903 ti library1 on
/dev/rmt/63cbn 904 ti library1 on
:wq
[samsharefs-client1]root@solaris:~#

```

11. 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 `samsharefs-client2`, so we use the hyphen as the family set name for the tape devices:

```

[samsharefs-client2]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
samsharefs 800 ms samsharefs on
...
Archival storage for copies:
/dev/rmt/60cbn 901 ti - on
/dev/rmt/61cbn 902 ti - on
/dev/rmt/62cbn 903 ti - on
/dev/rmt/63cbn 904 ti - on
:wq
[samsharefs-client2]root@solaris:~#

```

12. 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 `samsharefs-mds` and use the `vi` editor to turn distributed I/O off for copy 1:

```

[samsharefs-mds]root@solaris:~# vi /etc/opt/SUNWsamfs/archiver.cmd
#archiver.cmd
...
params
allsets -sort path -offline_copy stageahead
allsets.1 -startage 10m -startsize 500M -startcount 500000 -distio off

```

```
allsets.2 -startage 24h -startsize 20G -startcount 500000 -reserve set
:wq
[samsharefs-mds]root@solaris:~#
```

13. Check the configuration files for errors by running the `sam-fsd` command. Correct any errors found.

The `sam-fsd` command reads SAM-QFS configuration files and initializes file systems. It will stop if it encounters an error. In the example, we run the command on the server, `sharefs-mds`:

```
[sharefs-mds]root@solaris:~# sam-fsd
```

14. Tell the SAM-QFS service to read the modified configuration files and reconfigure itself accordingly. Correct any errors reported and repeat as necessary.

```
[sharefs-mds]root@solaris:~# samd config
```

15. 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:

```
[samsharefs-mds]root@solaris:~# samcmd g
Shared clients samcmd 5.4.dist_tapeio 11:09:13 Jul 2 2014
samcmd on samsharefs-mds
samsharefs is shared, server is samsharefs-mds, 2 clients 3 max
ord hostname seqno nomsgs status config conf1 flags
 1 samsharefs-mds 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 samsharefs-client1 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

...
```

16. If you plan on using the sideband database feature, go to "[Configuring the SAM-QFS Reporting Database](#)" on page 10-1.

17. 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 SAM-QFS 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 SAM-QFS software's native support for multiple-host file-system access (see "[Accessing File Systems from Multiple Hosts Using SAM-QFS Software](#)" on page 7-1). The following sections outline the basic configuration steps:

- [Sharing SAM-QFS File Systems Using NFS](#)

- [Sharing SAM-QFS File Systems Using SMB/CIFS](#)

## Sharing SAM-QFS File Systems Using NFS

Carry out the following tasks:

- [Disable Delegation Before Using NFS 4 to Share a SAM-QFS Shared File System](#)
- [Configure NFS Servers and Clients to Share SAM-QFS WORM Files and Directories](#), if necessary.
- [Configure the NFS Server on the SAM-QFS Host](#)
- [Share the SAM-QFS File System as an NFS Share](#)
- [Mount the NFS-Shared SAM-QFS File System on the NFS Clients](#)

### Disable Delegation Before Using NFS 4 to Share a SAM-QFS Shared File System

If you use NFS to share a SAM-QFS shared file system, you need to make sure that the SAM-QFS 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 SAM-QFS 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 SAM-QFS 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 host of the SAM-QFS file system that you want to share using NFS. Log in as root.

If the file system is a SAM-QFS shared file system, log in to the metadata server for the file system. In the examples below, the server name is `qfsnfs`.

```
[qfsnfs]root@solaris:~#
```

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.

```
[qfsnfs]root@solaris:~# 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.

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

```
[qfsnfs]root@solaris:~# 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
[qfsnfs]root@solaris:~#
```

4. If the SAM-QFS file system that you intend to share supports the Write-Once Read-Many (WORM) feature, you should [Configure NFS Servers and Clients to](#)

[Share SAM-QFS WORM Files and Directories](#) now.

5. Otherwise, [Configure the NFS Server on the SAM-QFS Host](#).

### Configure NFS Servers and Clients to Share SAM-QFS WORM Files and Directories

1. Log in to the host of the SAM-QFS file system that you want to share using NFS. Log in as root.

If the file system is a SAM-QFS shared file system, log in to the metadata server for the file system. In the examples below, the server name is `qfsnfs` and the client name is `nfsclient1`.

```
[qfsnfs]root@solaris:~#
```

2. If the SAM-QFS 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 `qfsnfs` and the client `nfsclient1`. 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:

```
[qfsnfs]root@solaris:~# uname -r
5.11
[qfsnfs]root@solaris:~# modinfo | grep -i "nfs.* version 4"
258 7a600000 86cd0 28 1 nfs (network filesystem version 4)
[qfsnfs]root@solaris:~# ssh root@nfsclient1
Password: ...
[nfsclient1]root@solaris:~# uname -r
5.11
[nfsclient1]root@solaris:~# modinfo | grep -i "nfs.* version 4"
278 ffffffff8cba000 9df68 27 1 nfs (network filesystem version 4)
[nfsclient1]root@solaris:~# exit
[qfsnfs]root@solaris:~#
```

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:

```
[qfsnfs]root@solaris:~# sharectl set -p server_versmax=4 nfs
[qfsnfs]root@solaris:~# ssh root@nfsclient1
Password ...
[nfsclient1]root@solaris:~# sharectl set -p server_versmax=4 nfs
[nfsclient1]root@solaris:~# exit
[qfsnfs]root@solaris:~#
```

4. Next, [Configure the NFS Server on the SAM-QFS Host](#)

### Configure the NFS Server on the SAM-QFS Host

Before clients can successfully mount a SAM-QFS file system using Network File System (NFS), you must configure the NFS server so that it does not attempt to share the SAM-QFS 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 host of the SAM-QFS file system that you want to share using NFS. Log in as root.

If the file system is a SAM-QFS shared file system, log in to the metadata server for the file system. In the examples below, the server name is `qfsnfs`.

```
[qfsnfs]root@solaris:~#
```

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`:

```
[qfsnfs]root@solaris:~# svccfg export /network/nfs/server > /var/tmp/server.xml
[qfsnfs]root@solaris:~#
```

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`:

```
[qfsnfs]root@solaris:~# 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 SAM-QFS 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
[qfsnfs]root@solaris:~#

```

5. Validate the manifest file using the `svccfg validate` command.

```
[qfsnfs]root@solaris:~# 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:

```
[qfsnfs]root@solaris:~# 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
[qfsnfs]root@solaris:~# vi /var/tmp/server.xml
```

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:

```
[qfsnfs]root@solaris:~# svccfg validate /var/tmp/server.xml
[qfsnfs]root@solaris:~# svcadm disable nfs/server
```

8. Delete the existing NFS server configuration using the `svccfg delete nfs/server` command.

```
[qfsnfs]root@solaris:~# svccfg delete nfs/server
```

9. Import the manifest file into the Service Management Facility (SMF) using the `svccfg import` command.

```
[qfsnfs]root@solaris:~# 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.

```
[qfsnfs]root@solaris:~# 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:

```
[qfsnfs]root@solaris:~# svcs -d svc:/network/nfs/server:default
STATE STIME FMRI
...
online Nov_01 svc:/network/qfs/shared-mount:default
...
```

12. Next, [Share the SAM-QFS File System as an NFS Share](#).

### Share the SAM-QFS File System as an NFS Share

Share the SAM-QFS 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 SAM-QFS file system that you want to share using NFS. Log in as root.

If the file system is a SAM-QFS shared file system, log in to the metadata server for the file system. In the examples below, the server name is `qfsnfs`.

```
[qfsnfs]root@solaris:~#
```

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` is 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 `/qfsm`s file system read/write with clients `nfsclient1` and `nfsclient2` and read-only with `nfsclient3`:

```
[qfsnfs]root@solaris:~# share -F nfs -o rw=nfsclient1:nfsclient2 \
ro=nfsclient3 /qfsm
```

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:

```
[qfsnfs]root@solaris:~# share -F nfs
/qfsm sec=sys,rw=nfsclient1:nfsclient2,ro=nfsclient3
[qfsnfs]root@solaris:~#
```

4. Next, [Mount the NFS-Shared SAM-QFS File System on the NFS Clients](#).

### Mount the NFS-Shared SAM-QFS 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 `nfsclient1`:

```
[nfsclient1]root@solaris:~#
```

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

```
[nfsclient1]root@solaris:~# cp /etc/vfstab /etc/vfstab.backup
```

3. Open the /etc/vfstab file in a text editor.

In the example, we use the vi editor.

```
[nfsclient1]root@solaris:~# vi /etc/vfstab
#File Device
#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 qfsnfs, the shared file system is named qfsms, and the mount point on the server is /qfsms:

```
#File Device
#Device to Mount System fsck at Mount
#to Mount fsck Point Type Pass Boot Options
#-----
/devices - /devices devfs - no -
...
qfsnfs:/qfsms
```

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
#Device to Mount System fsck at Mount
#to Mount fsck Point Type Pass Boot Options
#-----
/devices - /devices devfs - no -
...
qfsnfs:/qfsms -
```

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 /qfsnfs:

```
#File Device
#Device to Mount System fsck at Mount
#to Mount fsck Point Type Pass Boot Options
#-----
/devices - /devices devfs - no -
...
qfsnfs:/qfsms - /qfsnfs
```

7. In the fourth column of the /etc/vfstab file, enter the file-system type nfs.

```
#File Device
#Device to Mount System fsck at Mount
#to Mount fsck Point Type Pass Boot Options
#-----
/devices - /devices devfs - no -
...
qfsnfs:/qfsms - /qfsnfs nfs
```

We use the `nfs` file-system type, because the client mounts the remote QFS file system as an NFS file system.

- 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 -
...
qfsnfs:/qfsms - /qfsnfs nfs -
```

- 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 -
...
qfsnfs:/qfsms - /qfsnfs nfs - yes
```

- 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 -
...
qfsnfs:/qfsms - /qfsnfs nfs - yes soft,retrans=120,timeo=3000
```

- If you are using NFS 2, set the `rsiz` mount parameter to 32768.

Accept the default value for other versions of NFS.

The `rsiz` mount parameter sets the read buffer size to 32768 bytes (vs. the default, 8192 bytes). The example shows what an NFS 2 configuration would like:

```
#File Device Mount
#Device to Mount System fsck at Mount
#to Mount fsck Point Type Pass Boot Options
#-----
/devices - /devices devfs - no -
...
qfsnfs2:/qfs2 - /qfsnfs2 nfs - yes ...,rsiz=32768
```

12. If you are using NFS 2, set the `wsiz` mount parameter to 32768.

Accept the default value for other versions of NFS.

The `wsiz` 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 like:

```
#File Device Mount System fsck at Mount
#Device to Mount System fsck at Mount
#to Mount fsck Point Type Pass Boot Options
#-----
/devices - /devices devfs - no -
...
qfsnfs2:/qfs2 - /qfsnfs2 nfs - yes ... ,wsiz=32768
```

13. Save the `/etc/vfstab` file and exit the editor.

```
#File Device Mount System fsck at Mount
#Device to Mount System fsck at Mount
#to Mount fsck Point Type Pass Boot Options
#-----
/devices - /devices devfs - no -
...
qfsnfs:/qfsms - /qfsnfs nfs - yes soft,retrans=120,timeo=3000
:wq
[nfsclient1]root@solaris:~#
```

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 `/qfsnfs`:

```
[nfsclient1]root@solaris:~# mkdir /qfsnfs
```

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 `/qfsnfs` mount-point directory and set permissions to 755 (`-rwxr-xr-x`):

```
[nfsclient1]root@solaris:~# mkdir /qfsnfs
[nfsclient1]root@solaris:~# chmod 755 /qfsnfs
```

16. Mount the shared file system:

```
[nfsclient1]root@solaris:~# mount /qfsnfs
```

17. If you plan on using the sideband database feature, go to "[Configuring the SAM-QFS Reporting Database](#)" on page 10-1.

18. Otherwise, go to "[Configuring Notifications and Logging](#)" on page 11-1.

## Sharing SAM-QFS File Systems Using SMB/CIFS

SMB makes SAM-QFS 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).

Starting with Release 5.4, SAM-QFS 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 SAM-QFS Release 5.4 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 SAM-QFS File Systems for Sharing With SMB/CIFS.](#)
- [Configure the SMB Server for Windows Active Directory Domains or Workgroups.](#)
- [Share the SAM-QFS File System as an SMB/CIFS Share.](#)

### **Review Oracle Solaris SMB Configuration and Administration Documentation**

The sections below outline the parts of the SMB configuration process as they apply to SAM-QFS 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 11.1* in the *Oracle Solaris 11.1 Information Library*.

### **Explicitly Map Windows Identities for the SMB Server (Optional)**

While SAM-QFS 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. See the *Managing SMB and Windows Interoperability in Oracle Solaris 11.1* for full instructions.

### **Configure SAM-QFS File Systems for Sharing With SMB/CIFS**

SAM-QFS 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 SAM-QFS Release 5.4 use NFS version-4 ACLs by default on Solaris 11. But, if you need to share existing SAM-QFS file systems with SMB/CIFS clients, you must convert the existing POSIX-style ACLs using the appropriate procedure:

- [Convert a SAM-QFS Unshared File System that Uses POSIX-Style ACLs](#)
- [Convert a SAM-QFS File Shared System that Uses POSIX-Style ACLs](#)

**Convert a SAM-QFS 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 `qfs-host`

```
[qfs-host]root@solaris:~#
```

2. Make sure that the host runs Oracle Solaris 11.1 or higher. Use the command `uname -r`.

```
[qfs-host]root@solaris:~# uname -r
5.11
[qfs-host]root@solaris:~#
```

3. Unmount the file system using the command `umount mount-point`, where `mount-point` is the mount point of the SAM-QFS file system.

See the `umount_samfs` man page for further details. In the examples below, the server name is `qfs-host` and the file system is `/qfsms`:

```
[qfs-host]root@solaris:~# umount /qfsms
```

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 `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).

```
[qfs-host]root@solaris:~# samfsck -F -A /qfsms
```

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).

```
[qfs-host]root@solaris:~# samfsck -F -a /qfsms
```

6. Now, [Configure the SMB Server for Windows Active Directory Domains or Workgroups](#).

### Convert a SAM-QFS File Shared System that Uses POSIX-Style ACLs

1. Log in to the file-system metadata server (MDS) as `root`.

In the example, we log in to the metadata server `sharedqfs-mds`:

```
[sharedqfs-mds]root@solaris:~#
```

2. Make sure that the metadata server runs Oracle Solaris 11.1 or higher. Use the command `uname -r`.

```
[sharedqfs-mds]root@solaris:~# uname -r
5.11
[sharedqfs-mds]root@solaris:~#
```

3. Log in to each SAM-QFS 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 `sharedqfs-client1` and `sharedqfs-client2` using `ssh` to get the Solaris version from the log-in banner:

```
[sharedqfs-mds]root@solaris:~# ssh root@sharedqfs-client1
Password:
Oracle Corporation SunOS 5.11 11.1 September 2013
[sharedqfs-client1]root@solaris:~#
```

```
[sharedqfs-mds]root@solaris:~# ssh root@sharedqfs-client2
```

```

Password:
Oracle Corporation SunOS 5.11 11.1 September 2013
[sharedqfs-client2]root@solaris:~#

```

4. Unmount the SAM-QFS shared file system from each SAM-QFS client using the command `umount mount-point`, where *mount-point* is the mount point of the SAM-QFS file system.

See the `umount_samfs` man page for further details. In the example, we unmount `/sharedqfs1` from our two clients, `sharedqfs-client1` and `sharedqfs-client2`:

```

Oracle Corporation SunOS 5.11 11.1 September 2013
[sharedqfs-client1]root@solaris:~# umount /sharedqfs
[sharedqfs-client1]root@solaris:~#

```

```

Oracle Corporation SunOS 5.11 11.1 September 2013
[sharedqfs-client2]root@solaris:~# umount /sharedqfs
[sharedqfs-client1]root@solaris:~#

```

5. Unmount the SAM-QFS 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 SAM-QFS 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 a SAM-QFS 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 a SAM-QFS client. See the `umount_samfs` man page for further details.

In the example, we unmount the `/sharedqfs` file system from the metadata server `sharedqfs-mds` while allowing 60 seconds for clients to unmount:

```

[sharedqfs-mds]root@solaris:~# umount -o await_clients=60 /sharedqfs

```

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` 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 a SAM-QFS file system named `/sharedqfs`:

```

[sharedqfs-mds]root@solaris:~# samfsck -F -A /sharedqfs

```

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 SAM-QFS file system named `/qfsma`:

```

[sharedqfs-mds]root@solaris:~# samfsck -F -a /sharedqfs

```

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 the 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 SAM-QFS file system is to be shared.

2. Log in to the host of the SAM-QFS file system that you want to share using SMB/CIFS. Log in as `root`.

If the file system is a SAM-QFS shared file system, log in to the metadata server for the file system. In the examples below, the server name is `qfssmb`.

```
[qfssmb]root@solaris:~#
```

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:

```
[qfssmb]root@solaris:~# 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 `svcadm disable` command.

```
[qfssmb]root@solaris:~# svcadm disable svc:/network/samba
[qfssmb]root@solaris:~# 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.

```
[qfssmb]root@solaris:~# svcadm enable -r smb/server
```

6. Make sure that the system clock on the SAM-QFS 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 SAM-QFS host to use the same servers. Create an `/etc/inet/ntpclient.conf` file on the SAM-QFS host and start the `ntpd`

daemon using the `svcadm enable ntp` command (see the `ntpd` man page and your Oracle Solaris administration documentation for full information).

- Otherwise, synchronize the SAM-QFS host with the domain controller by running the `ntpdate domain-controller-name` command (see the `ntpdate` man page for details) or manually set the system clock on the SAM-QFS 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

```
[qfssmb]root@solaris:~# smbadm join -u admin -o smbsharing this.example.com
```

8. Now [Share the SAM-QFS 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 SAM-QFS file system should join.

The default workgroup is named `WORKGROUP`.

2. Log in to the host of the SAM-QFS file system. Log in as `root`.

If the file system is a SAM-QFS shared file system, log in to the metadata server for the file system. In the examples below, the server name is `qfssmb`.

```
[qfssmb]root@solaris:~#
```

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:

```
[qfssmb]root@solaris:~# 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 `svcadm disable` command.

Samba and SMB servers cannot be used together on a single Oracle Solaris system.

```
[qfssmb]root@solaris:~# svcadm disable svc:/network/samba
[qfssmb]root@solaris:~# 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.

```
[qfssmb]root@solaris:~# svcadm enable -r smb/server
```

6. Join the workgroup using the `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`.

```
[qfssmb]root@solaris:~# smbadm join -w crossplatform
```

7. Configure the SAM-QFS 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:

```
[qfssmb]root@solaris:~# 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
[qfssmb]root@solaris:~#
```

See the `pam_smb_passwd` man page for further details.

8. Once the `pam_smb_passwd` module has been installed, use the `passwd local-username` command 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`:

```
[qfssmb]root@solaris:~# passwd smbsamqfs
```

9. Now [Share the SAM-QFS File System as an SMB/CIFS Share](#).

### Share the SAM-QFS File System as an SMB/CIFS Share

Share the SAM-QFS 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 SAM-QFS file system that you want to share using SMB/CIFS. Log in as `root`.

If the file system is a SAM-QFS shared file system, log in to the metadata server for the file system. In the examples below, the server name is `qfssmb`.

```
[qfssmb]root@solaris:~#
```

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`, can include 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 specified 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=[*|[-][hostname|netgroup|domainname.suffix|ipaddress|@netname][:...]]`

The `rw` policy grants or denies read-write 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 criteria for deciding whether clients can or cannot access the share. The criteria can include specified hostnames, network groups, full LDAP/DNS domain names, and/or the symbol @ plus all or part of an IP address or domain name. A minus sign (-) preceding an entry denies access to that list item. 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=[*|[-][hostname|netgroup|domainname.suffix|ipaddress|@netname][:...]]`

The `ro` policy grants or denies read-only access to any client that matches the access list.

- `none=[* | [-][hostname | netgroup | domainname.suffix | @ipaddress | @netname]][:...]`

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 `/qfsms` file system read/write with clients `smbclient1` and `smbclient2` and read-only with `smbclient3`:

```
[qfssmb]root@solaris:~# share -F smb -o rw=smbclient1:smbclient2 \
ro=smbclient3 /qfsms
```

When you enter the command, the system automatically restarts the SMB server daemon, `smbd`.

3. Check the sharing parameters. Use the command `share -F nfs`.

In the example, the command output shows that we have correctly configured the share:

```
[qfssmb]root@solaris:~# share -F smb
/qfsms sec=sys,rw=smbclient1:smbclient2,ro=smbclient3
```

4. If you plan on using the sideband database feature, go to "[Configuring the SAM-QFS Reporting Database](#)" on page 10-1.
5. Otherwise, go to "[Configuring Notifications and Logging](#)" on page 11-1.



---

---

## Configuring SAM-Remote

Oracle StorageTek SAM-Remote client and server software lets SAM-QFS servers access archival storage resources that are hosted on remote SAM-QFS servers. Shared resources and client/server relationships are defined in the `/etc/opt/SUNWsamfs/mcf` file using special SAM-Remote equipment types. Typically, one or more SAM-Remote server hosts are configured with archival solid-state disk (SSD) or magnetic disk storage and an attached, automated tape library. One or more SAM-Remote client hosts are configured with archival solid-state or disk storage only. The archiving policy on each SAM-Remote client specifies one or more copies in the local disk archive and a remote copy on the SAM-Remote server's archival solid-state or disk storage. When multiple servers are configured, the archiving policy on each server host can specify one or more archival copies on media hosted by another server. So, if a server cannot access a local library, it can access its archival data by requesting the required files as a client. The other host then acts as the server and automatically retrieves the files from its own, local library.

You can address a number of otherwise difficult archiving and data-protection requirements by configuring file-system servers as SAM-Remote client/server pairs.

- Automated creation and maintenance of off-site backup copies  
The archiving policies on each host in a client/server pair specify one or more archival copies on the host's local media and one copy on media hosted by its opposite number.
- Centralized archival storage resources for QFS file systems hosted at remote locations.  
The archiving policies on a SAM-QFS server in a regional office or satellite campus specify one or more archival copies on a local disk archive and one copy on tape media hosted by a SAM-Remote server in a centrally located data center.
- Access to remote archival storage resources when local resources are unavailable.  
The archiving policies on each host in a client/server pair specify one or more archival copies on the host's local media and one copy on media hosted by its opposite number. If a host cannot access its local library, it can access its archival data by requesting the required files from its opposite number.

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 SAM-QFS 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 SAM-QFS 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`:

```
[server1]root@solaris:~#
```

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`:

```
[server1]root@solaris:~# ssh root@client1
Password: ...
[client1]root@solaris:~#
```

3. Make sure that SAM-QFS 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 SAM-QFS software:

```
[server1]root@solaris:~# samcmd l
Usage information samcmd 5.4 11:44:35 Apr 18 2014
samcmd on server1
...
[client1]root@solaris:~# samcmd l
Usage information samcmd 5.4 11:44:35 Apr 18 2014
samcmd on client1
...
```

4. Using the Solaris Image Packaging System (IPS), update host software as necessary until all SAM-Remote servers and clients are at the same revision levels.
5. Next, [Stop SAM-QFS Processes](#).

## Stop SAM-QFS Processes

1. Log in to the SAM-Remote server host as `root`.

In the example, the server is named `server1`:

```
[server1]root@solaris:~#
```

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:

```
[server1]root@solaris:~# samcmd c
Device configuration samcmd 5.4 17:37:31 Dec 31 2013
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. Idle the removable media devices in the robotic library. Use the command `samcmd idle equipment-ordinal`, where *equipment-ordinal* is the equipment ordinal number specified for the device in the `/etc/opt/SUNWsamfs/mcf` file.

Idling the devices lets running processes complete any current tasks before stopping.

```
[server1]root@solaris:~# samcmd idle 801
[server1]root@solaris:~# samcmd idle 802
[server1]root@solaris:~# samcmd idle 803
[server1]root@solaris:~# samcmd idle 804
```

4. Once all removable media devices are idle, stop the archiving processes. Use the command `/opt/SUNWsamfs/sbin/samd stop`.

```
[server1]root@solaris:~# samd stop
```

5. Next, [Configure the SAM-Remote Server](#).

## Configure the SAM-Remote Server

A SAM-Remote server is a SAM-QFS file-system host that makes its attached archiving equipment—disk archives, robotic tape libraries, and tape drives—available to remote clients that are themselves SAM-QFS file-system hosts. The SAM-Remote server must mount at least one QFS file system to start SAM-QFS 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`:

```
[server1]root@solaris:~#
```

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 QFS archiving file system, `fs600`, three disk-archive file systems, `afs700`, `afs710`, and `afs720`, 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:

```
[server1]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
#=====
QFS File-System Equipment Definitions:
#
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
=====
Archiving Equipment Definitions:
Local tape library "rb800"
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
/dev/scsi/changer/clt0d5 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

```

- At the end of the archiving equipment definitions, start an entry for the equipment that the server will make 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 new equipment:

```

...
=====
Archiving Equipment Definitions:
#
...
#
Server's remotely shared archival media equipment "am500"
#
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
/etc/opt/SUNWsamfs/samremote 500

```

- In the Equipment Type field of the new entry, enter `ss`, for SAM-Remote server equipment.

```

...
Server's remotely shared archival media equipment "am500"
#
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
/etc/opt/SUNWsamfs/samremote 500 ss

```

- 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 `am500` to the new equipment:

```

...
Server's remotely shared archival media equipment "am500"
#
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
/etc/opt/SUNWsamfs/samremote 500 ss am500 on

```

6. If you plan to configure more than ten SAM-Remote clients, configure an additional server for each successive group of one to ten clients. Then save the file and close the editor.

In the example, we do not plan to add more than ten clients:

```
...
/etc/opt/SUNWsamfs/samremote 500 ss am500 on
:wq
[server1]root@solaris:~#
```

7. 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`:

```
[server1]root@solaris:~#
```

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, indicated by a hash (`#`) sign:

```
[server1]root@solaris:~# 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`:

```
[server1]root@solaris:~# 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:

```
[server1]root@solaris:~# 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 media used by this equipment (see [Appendix A, "Glossary of Equipment Types"](#) for a full list of SAM-QFS 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 a library supports more than one type, you must specify each type in a new entry:

```
media
 900 ti VOL500 VOL501
 900 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
[server1]root@solaris:~#
```

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
- 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`:

```
[client1]root@solaris:~#
```

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 QFS archiving file system, `fs100` and one disk-archive file system, `afs200`. Note that the example includes clarifying headings that may not be present in actual files and abbreviates lengthy device paths.

```
[client1]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
Client's /etc/opt/SUNWsamfs/mcf file
#=====
QFS File-System Equipment Definitions:
#
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
#=====
Archiving Equipment Definitions:
#
Archival File System "afs200"
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
afs200 200 ms afs200 on
/dev/dsk/c10t60...7Bd0s7 210 md afs200 on
/dev/dsk/c10t60...48d0s7 211 md afs200 on
```

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, the client configuration file is `/etc/opt/SUNWsamfs/am500`. We assign it the equipment ordinal number 400. We also add some headings as comments:

```
...
```

```

...
Client's remote archival media equipment "am500"
#
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
/etc/opt/SUNWsamfs/am500 400

```

4. In the `Equipment Type` field of the new entry, enter `sc`, for SAM-Remote client equipment.

```

Client's remote archival media equipment "am500"
#
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
/etc/opt/SUNWsamfs/am500 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 `am500` to the new equipment.

```

Client's remote archival media equipment "am500"
#
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
/etc/opt/SUNWsamfs/am500 400 sc am500 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/rddevice-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`:

```

Client's remote archival media equipment "am500"
#
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
/etc/opt/SUNWsamfs/am500 400 sc am500 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`:

```

Client's remote archival media equipment "am500"
#
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
/etc/opt/SUNWsamfs/am500 400 sc am500 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.

```
Client's remote archival media equipment "am500"
#
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
/etc/opt/SUNWsamfs/am500 400 sc am500 on
/dev/samrd/rd0 410 rd rd
/dev/samrd/rd1 420 rd 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 `am500`:

```
Client's remote archival media equipment "am500"
#
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
/etc/opt/SUNWsamfs/am500 400 sc am500 on
/dev/samrd/rd0 410 rd am500
/dev/samrd/rd1 420 rd am500
```

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`:

```
Client's remote archival media equipment "am500"
#
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
/etc/opt/SUNWsamfs/am500 400 sc am500 on
/dev/samrd/rd0 410 rd am500 on
/dev/samrd/rd1 420 rd am500 on
:wq
[client1]root@solaris:~#
```

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`:

```
[client1]root@solaris:~#
```

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 `am500`. We also document the file with some descriptive comments, indicated by a hash (`#`) sign:

```
[client1]root@solaris:~# vi /etc/opt/SUNWsamfs/am500
Client's SAM-Remote client configuration file: /opt/SUNWsamfs/am500
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`:

```
[client1]root@solaris:~# vi /etc/opt/SUNWsamfs/samremote
Client's SAM-Remote server configuration file: /opt/SUNWsamfs/am500
This file identifies the host of the SAM-Remote server.
server1
:wq
[client1]root@solaris:~#
```

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`:

```
[client1]root@solaris:~#
```

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:

```
[client1]root@solaris:~# vi /etc/opt/SUNWsamfs/archiver.cmd
...
#-----
Copy Parameter Directives
params
all -sort path -offline_copy direct
all.1 -startage 10m -startsize 500M -drives 10
all.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, `all.3`:

```
#-----
Copy Parameter Directives
Copy Parameter Directives
params
all -sort path -offline_copy direct
all.1 -startage 10m -startsize 500M -drives 10
all.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, `all.1`, will be made using the local disk archive volume, `afs200`:

```
...
endparams
#-----
VSN Directives
vsns
all.1 dk afs200
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 `all.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
all.1 dk afs200
all.2 tp VOL0[0-1][0-9]
endvsns
:wq
[client1]root@solaris:~#
```

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`:

```
[server1]root@solaris:~#
```

2. Start the SAM-QFS processes on the server. Use the command `samd start`:

```
[server1]root@solaris:~# 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:

```
[server1]root@solaris:~# samcmd s
Device status samcmd 5.4 17:51:05 Jan 2 2014
samcmd on server1
ty eq state device_name fs status
rb 800 on /dev/scsi/changer/clt0d5 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 am500 -----o-r

```

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, and 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 8-3 and ["Create the samremote Server Configuration File"](#) on page 8-5.

5. On the server, check the connection status of the remote, shared-device 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):

```

[server1]root@solaris:~# samcmd R
Remote server eq: 500 addr: 00003858 samcmd 5.4 17:51:05 Jan 2 2014
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

```

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:

```

[server1]root@solaris:~# getent hosts client1
192.10.10.3 client1
[server1]root@solaris:~# getent hosts 192.10.10.3
192.10.10.3 client1
[server1]root@solaris:~# ping 192.10.10.3
192.10.10.31 is alive
[server1]root@solaris:~# getent hosts client2
10.1.229.97 client2
[server1]root@solaris:~# getent hosts 10.1.229.97
10.1.229.97 client2
[server1]root@solaris:~# ping 10.1.229.97
192.10.10.31 is alive
[server1]root@solaris:~# ssh root@client1
Password: ...
[client1]root@solaris:~# getent hosts server1
192.10.201.12 server1
...
[client1]root@solaris:~# exit
[server1]root@solaris:~# ssh root@client2
Password: ...
[client2]root@solaris:~# getent hosts server1
192.10.201.12 server1
...
[client2]root@solaris:~# exit
[server1]root@solaris:~#

```

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 8-7 and ["Create the SAM-Remote Client Configuration File"](#) on page 8-9.

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 8-9.

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 8-5.

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/am500`. The output correctly lists the volumes that `client1` can access, VOL000 to VOL019:

```
[server1]root@solaris:~# samcmd v 400
Robot catalog samcmd 5.4 18:30:35 Jan 2 2014
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
[server1]root@solaris:~#
```

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 8-5 and ["Create the SAM-Remote Client Configuration File"](#) on page 8-9.

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`:

```
[client1]root@solaris:~#
```

2. Start the SAM-QFS processes on the client host. Use the command `samd start`:

```
[client1]root@solaris:~# samd start
```

3. On the client host, check the status of the shared-device client. Use the command `samcmd s`.

In the example, the SAM-Remote server equipment (type `sc`) with equipment ordinal number 400 is on and operating normally:

```
[client1]root@solaris:~# samcmd s
Device status samcmd 5.4 18:06:51 Jan 2 2014
samcmd on client1
ty eq state device_name fs status
sc 400 on /etc/opt/SUNWsamfs/am500 am500 -----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 8-7 and ["Create the SAM-Remote Client Configuration File"](#) on page 8-9.

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 `all.2` will be made using one of the remote tape volumes in the range `VOL000-VOL019`, as specified in the `samremote` server configuration file:

```
[client1]root@solaris:~# 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: all.1 dk afs200
34: all.2 tp VOL0[0-1][0-9]
36: endvsns
[client1]root@solaris:~#
```

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`:

```
[server1]root@solaris:~#
```

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:

```
[client1]root@solaris:~# vi /etc/opt/SUNWsamfs/archiver.cmd
...
#-----
Copy Parameter Directives
params
all -sort path -offline_copy direct
all.1 -startage 10m -startsize 500M -drives 10
all.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 cannot specify recycling by library. You must *configure recycling by archive sets*, in the `params` section of the `archiver.cmd` file.

In the example, we add recycling directives for archive sets `all.1` and `all.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` schedules no more than 1 removable media volume for recycling at a time:

```
#-----
Copy Parameters Directives
params
all -sort path -offline_copy direct
all.1 -startage 10m -startsize 500M -drives 10
all.1 -recycle_mingain 90
all.2 -startage 24h -startsize 20G -drives 2 -reserve set offline_copy none
```

```
all.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 and not to volumes that are accessible from the clients*. In the example, the server's recycling directives for copy all.1 apply to the disk archive equipment afs700, not to afs710 (which is reserved for client1) or afs720 (which is reserved for client2). The server's recycling directives for copy all.2 apply to tape volumes VOL100-VOL199, but not to volumes VOL000-VOL019 (which are reserved for client1) or to volumes VOL020-VOL039 (which are reserved for client2):

```
...
endparams
#-----
VSN Directives
vsns
all.1 dk afs700
all.2 tp VOL1[0-9][0-9]
endvsns
```

4. Save the archiver.cmd file, and close the editor.

```
...
endvsns
:wq
[server1]root@solaris:~#
```

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:

```
[server1]root@solaris:~# 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-recycle 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-recycle` 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-recycle` directive for media type `tp` (tape) volumes VOL000-VOL019 (assigned to client1) and VOL020-VOL039 (assigned to client2):

```
[server1]root@solaris:~# 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
[server1]root@solaris:~#
```

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`:

```
[client1]root@solaris:~#
```

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.

```
[client1]root@solaris:~# vi /etc/opt/SUNWsamfs/archiver.cmd
...
#-----
Copy Parameters Directives
params
allsets -sort path -offline_copy stageahead
allsets.1 -startage 6h -startsize 6G -startcount 500000
allsets.2 -startage 6h -startsize 6G -startcount 500000
allsets.2 -rearch_stage_copy 1
allsets.3 -startage 24h -startsize 20G -startcount 500000 -archmax 24G
allsets.3 -rearch_stage_copy 2
endparams
#-----
VSN Directives
vsns
all.1 dk afs200
all.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 cannot specify recycling by library. You must *configure recycling by archive sets*, in the params section of the `archiver.cmd` file.

In the example, we add recycling directives for archive sets `allsets.1` and `allsets.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 1 removable media volume for recycling at a time.

```
#-----
Copy Parameters Directives
params
allsets -sort path -offline_copy stageahead
allsets.1 -startage 6h -startsize 6G -startcount 500000
allsets.1 -recycle_mingain 90
allsets.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

all.1 apply to the client's local disk archive equipment, afs200. The client's recycling directives for copy all.2 apply to the server-provided remote tape volumes VOL000-VOL019:

```
...
endparams
#-----
VSN Directives
vsns
all.1 dk afs200
all.2 tp VOL0[0-1][0-9]
endvsns
:wq
[client1]root@solaris:~#
```

4. Save the archiver.cmd file, and close the editor.

```
...
endvsns
:wq
[client]root@solaris:~#
```

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-recycle directives.

In the example, we use the vi editor. We specify the default location for the log file:

```
[client1]root@solaris:~# vi /etc/opt/SUNWsamfs/recycler.cmd
logfile = /var/adm/recycler.log
:wq
[client1]root@solaris:~#
```

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 SAM-QFS Reporting Database"](#) on page 10-1.
9. Otherwise, go to ["Configuring Notifications and Logging"](#) on page 11-1.

---

---

## Preparing High-Availability Solutions

SAM-QFS high-availability configurations are designed to maintain uninterrupted file-system and archiving services. SAM-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, SAM-QFS 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, [Understanding the Supported SAM-QFS High-Availability 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 SAM-QFS High-Availability Configurations

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 SAM-QFS configurations are thus tailored to four specific sets of deployment requirements:

- [HA-QFS, a High-Availability QFS Unshared, Standalone File-System Configuration](#)
- [HA-COTC, a QFS Shared File System with High-Availability Metadata Servers](#)
- [HA-SAM, a High-Availability, Archiving, QFS Shared File-System Configuration](#)
- [SC-RAC, a High-Availability QFS Shared File-System Configuration for Oracle RAC.](#)

#### HA-QFS, a High-Availability QFS Unshared, Standalone File-System Configuration

The High Availability QFS (HA-QFS) configuration insures that a QFS unshared, standalone file system remains accessible in the event of a host failure. The file system is configured on both nodes in a two-node cluster that Solaris Cluster software manages as a resource of type `SUNW.HAStoragePlus`. But 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.

Clients access data via high-availability 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 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 maintains the availability of a QFS metadata server so that QFS file system clients can continue to access their data even if a server fails. The file system is shared. QFS active and potential metadata servers are hosted on a two-node cluster managed by Solaris Cluster software. A SAM-QFS high-availability resource of type `SUNW.qfs` manages failover for the shared file-system servers within the cluster. All clients are hosted outside of the cluster. The clustered servers insure the availability of metadata, issue I/O licenses, and maintain the consistency of the file system.

If the node that hosts the active metadata server fails, Solaris Cluster software automatically activates the potential MDS on the healthy node and initiates failover. The QFS file system is shared, so it is already mounted on the newly activated metadata server node and remains mounted on the clients. Clients continue to receive metadata updates and I/O leases, so file-system can continue without interruption.

HA-COTC configurations must use high performance `ma` file systems with physically separate `mm` metadata devices and `mr` data devices. General purpose `ms` file systems and `md` devices are not supported. You can share HA-COTC file systems with non-SAM-QFS, network clients using the standard Network File System (NFS), but HA-NFS is not supported.

For implementation instructions, see ["High-Availability QFS Shared File Systems, Clients Outside the Cluster"](#) on page 9-5.

## **HA-SAM, a High-Availability, Archiving, QFS Shared File-System Configuration**

The High-Availability Storage Archive Manager (HA-SAM) configuration maintains the availability of an archiving file system by insuring that the QFS metadata server and the Storage Archive 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. A SAM-QFS high-availability resource of type `SUNW.qfs` manages failover for the servers.

Clients access data via high-availability Network File System (HA-NFS), NFS, or SMB/CIFS shares, with the active cluster node acting as a file server.

If the active SAM-QFS 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.

For implementation instructions, see ["High-Availability SAM-QFS Shared Archiving File Systems"](#) on page 9-20.

## **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.

## 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:

- [Create Unshared QFS File Systems on Both Cluster Nodes](#)
- [Configure the High-Availability QFS File System](#)
- If required, configure High-Availability Network File System (HA-NFS) sharing.

Detailed procedures for setting up HA-NFS are included in the *Oracle Solaris Cluster Data Service for Network File System (NFS) Guide* that is included in the *Oracle Solaris Cluster* online documentation library.

### 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`:

```
[qfs1mds-node1]root@solaris:~#
```

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 File System](#)" on page 6-2 or "[Configure a High-Performance File System](#)" on page 6-7. The HA-QFS configuration does not support QFS shared file systems.

3. Log in to remaining cluster node as `root`.

In the example, we log in to the host `qfs1mds-node2` using `ssh`:

```
[qfs1mds-node1]root@solaris:~# ssh root@qfs1mds-node2
Password:
[qfs1mds-node2]root@solaris:~#
```

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`:

```
[qfs1mds-node1]root@solaris:~#
```

2. If you have not already done so, define the `SUNW.HAStoragePlus` resource type for the Solaris Cluster software. Use the command `clresourcetype register SUNW.HAStoragePlus`.

`HAStoragePlus` is the Solaris Cluster resource type 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.

```
[qfs1mds-node1]root@solaris:~# clresourcetype register SUNW.HAStoragePlus
```

3. 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` (note that the command below is entered as a single line—the line break is escaped by the backslash):

```
[qfs1mds-node1]root@solaris:~# create -g qfsrg -t SUNW.HAStoragePlus \
-x FilesystemMountPoints=/global/samqfs1/qfs1
-x FilesystemCheckCommand=/bin/true haqfs
```

4. 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:

```
[qfs1mds-node1]root@solaris:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name Node Name Suspended Status

qfsrg qfs1mds-1 No Online
 qfs1mds-2 No Offline
```

5. 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 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:

```
[qfs1mds-node1]root@solaris:~# clresourcegroup switch -n qfs1mds-node2 qfsrg
[qfs1mds-node1]root@solaris:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name Node Name Suspended Status

qfsrg qfs1mds-1 No Offline
 qfs1mds-2 No Online
```

6. 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 resource group. Then use `clresourcegroup status` to check the result.

In the example, we successfully move the `qfsrg` resource group back to `qfs1mds-node1`:

```
[qfs1mds-node1]root@solaris:~# clresourcegroup switch -n qfs1mds-node1 qfsrg
[qfs1mds-node1]root@solaris:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name Node Name Suspended Status

qfsrg qfs1mds-node1 No Online
 qfs1mds-node2 No Offline
```

7. 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.
8. If you plan on using the sideband database feature, go to "[Configuring the SAM-QFS Reporting Database](#)" on page 10-1.
9. 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. This arrangement provides failover protection for QFS metadata and file-access leases, so file system clients do not lose access to their data if a server fails. But file-system clients and data devices remain outside the cluster, so that Solaris Cluster does contend with QFS software for control of QFS shared data.

To configure an HA-COTC file system, carry out the tasks below:

- [Create a QFS Shared File System Hosts File on Both HA-COTC Cluster Nodes](#)
- [Create Local Hosts Files on the QFS Servers and Clients Outside the HA-COTC 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 Servers](#)

- [Configure Hosts Outside the HA-COTC Cluster as QFS Shared File System Clients](#)
- If required, configure Network File System (NFS) shares, as described in "[Accessing File Systems from Multiple Hosts Using NFS and SMB/CIFS](#)" on page 7-29. High-Availability NFS (HA-NFS) is *not* supported.

## 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-node1` and `qfs1mds-node2`. We log in to the host `qfs1mds-node1`:

```
[qfs1mds-node1]root@solaris:~#
```

2. Display the cluster configuration using 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-node1` and `qfs1mds-node2`—that the cluster uses for data transport.

```
[qfs1mds-node1]root@solaris:~# cluster show
```

```
...
```

```
=== Cluster Nodes ===
```

```
Node Name: qfs1mds-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: qfs1mds-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` 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:

```
[qfs1mds-node1]root@solaris:~# vi /etc/opt/SUNWsamfs/hosts.qfs1
/etc/opt/SUNWsamfs/hosts.qfs1
#
#Host Name Network Interface Server On/ Additional
#----- -
#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-node1` and `qfs1mds-node2`, the hostnames of the cluster nodes that host the metadata servers for the file system:

```
#
#Host Name Network Interface Server On/ Additional
#----- -
#Ordinal Off Parameters
qfs1mds-node1
qfs1mds-node2
```

5. In the second column of each line, start supplying `Network Interface` information for host `Host Name`. Enter each HA-COTC cluster node's Solaris Cluster 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:

```
#
#Host Name Network Interface Server On/ Additional
#----- -
#Ordinal Off Parameters
qfs1mds-node1 clusternode1-priv,
qfs1mds-node2 clusternode2-priv,
```

6. Following the comma in the second column of each line, enter a virtual public hostname for the active metadata server, followed by spaces.

The HA-COTC 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 `qfs1mds-node1` to `qfs1mds-node2` and vice versa), we use a virtual hostname—`qfs1mds`—for both. Later, we will configure the Solaris Cluster software to always route requests for `qfs1mds` to the active metadata server:

```
#
#Host Name Network Interface Server On/ Additional
#----- -
#Ordinal Off Parameters
qfs1mds-node1 clusternode1-priv,qfs1mds
qfs1mds-node2 clusternode2-priv,qfs1mds
```

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-node1`, is the active metadata server, so it is ordinal 1 and the secondary node, `qfs1mds-node2`, is ordinal 2:

```
#
#Host Name Network Interface Server On/ Additional
#----- -
qfs1mds-node1 clusternode1-priv,qfs1mds 1
qfs1mds-node2 clusternode2-priv,qfs1mds 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).

```
#
#Host Name Network Interface Server On/ Additional
#----- -
qfs1mds-node1 clusternode1-priv,qfs1mds 1 0
qfs1mds-node2 clusternode2-priv,qfs1mds 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:

```
#
#Host Name Network Interface Server On/ Additional
#----- -
qfs1mds-node1 clusternode1-priv,qfs1mds 1 0 server
qfs1mds-node2 clusternode2-priv,qfs1mds 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:

```
#
#Host Name Network Interface Server On/ Additional
#----- -
qfs1mds-node1 clusternode1-priv,qfs1mds 1 0 server
qfs1mds-node2 clusternode2-priv,qfs1mds 2 0
qfs1client1 172.16.0.133 0 0
qfs1client2 172.16.0.147 0 0
:wq
[qfs1mds-node1]root@solaris:~#
```

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 Servers and Clients Outside the HA-COTC Cluster](#).

## Create Local Hosts Files on the QFS Servers and Clients Outside the HA-COTC 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-node1` and `qfs1mds-node2`. We log in to the host `qfs1mds-node1`:

```
[qfs1mds-node1]root@solaris:~#
```

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 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:

```
[qfs1mds-node1]root@solaris:~# vi /etc/opt/SUNWsamfs/hosts.qfs1.local
/etc/opt/SUNWsamfs/hosts.qfs1.local
#
#Host Name Network Interface Server On/ Additional
#----- -
#Ordinal Off Parameters
qfs1mds-node1 clusternode1-priv 1 0 server
qfs1mds-node2 clusternode2-priv 2 0
qfs1client1 172.16.0.133 0 0
qfs1client2 172.16.0.147 0 0
:wq
[qfs1mds-node1]root@solaris:~# ssh root@qfs1mds-node2
Password:
[qfs1mds-node2]root@solaris:~# vi /etc/opt/SUNWsamfs/hosts.qfs1.local
/etc/opt/SUNWsamfs/hosts.qfs1.local
#
#Host Name Network Interface Server On/ Additional
#----- -
#Ordinal Off Parameters
qfs1mds-node1 clusternode1-priv 1 0 server
qfs1mds-node2 clusternode2-priv 2 0
qfs1client1 172.16.0.133 0 0
qfs1client2 172.16.0.147 0 0
:wq
```

```
[qfs1mds-node2]root@solaris:~# exit
[qfs1mds-node1]root@solaris:~#
```

- Using a text editor, 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 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 virtual hostname for the active metadata server, `qfs1mds`. The Solaris Cluster software will route requests for `qfs1mds` to whichever server node is active:

```
[qfs1mds-node1]root@solaris:~# ssh root@qfsclient1
Password:
[qfs1client1]root@solaris:~# vi /etc/opt/SUNWsamfs/hosts.qfs1.local
/etc/opt/SUNWsamfs/hosts.qfs1.local
#
#Host Name Network Interface Server On/ Additional
#-----

qfs1mds qfs1mds 1 0 server
:wq
qfs1client1]root@solaris:~# exit
[qfs1mds-node1]root@solaris:~# ssh root@qfsclient2
Password:
[qfs1client2]root@solaris:~# vi /etc/opt/SUNWsamfs/hosts.qfs1.local
/etc/opt/SUNWsamfs/hosts.qfs1.local
#
#Host Name Network Interface Server On/ Additional
#-----

qfs1mds qfs1mds 1 0 server
:wq
[qfs1client2]root@solaris:~# exit
[qfs1mds-node1]root@solaris:~#
```

- 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 Node](#)
- [Exclude Data Devices from Cluster Control](#)
- [Mount the QFS File System on the Primary HA-COTC Node](#).

### Create a High-Performance QFS File System on the Primary HA-COTC Node

- 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-node1` is the primary node and active metadata server:

```
[qfs1mds-node1]root@solaris:~#
```

- Select the global storage devices that will be used for the QFS file system using 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:

```
[qfs1mds-node1]root@solaris:~# cldevice list -v
DID Device Full Device Path

d1 qfs1mds-node1:/dev/rdisk/c0t0d0
d2 qfs1mds-node1:/dev/rdisk/c0t6d0
d3 qfs1mds-node1:/dev/rdisk/clt1d0
d3 qfs1mds-node2:/dev/rdisk/clt1d0
d4 qfs1mds-node1:/dev/rdisk/clt2d0
d4 qfs1mds-node2:/dev/rdisk/clt2d0
d5 qfs1mds-node1:/dev/rdisk/clt3d0
d5 qfs1mds-node2:/dev/rdisk/clt3d0
d6 qfs1mds-node2:/dev/rdisk/c0t0d0
d7 qfs1mds-node2:/dev/rdisk/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`):

```
[qfs1mds-node1]root@solaris:~# 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.

```
[qfs1mds-node1]root@solaris:~# 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
[qfs1mds-node1]root@solaris:~#
```

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 SAM-QFS 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-node1`:

```
[qfs1mds-node1]root@solaris:~# sam-fsd
```

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-node1`, and creates a shared file system with the specified properties.

```
[qfs1mds-node1]root@solaris:~# sammkfs -S qfs1
```

7. Now [Exclude Data Devices from Cluster Control](#).

### Exclude 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 for QFS Data Devices in the HA-COTC Cluster](#) or
- [Place Shared Data Devices in a Local-Only Device Group on the HA-COTC Cluster](#).

#### Disable Fencing for 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-node1` is the primary node and active metadata server:

```
[qfs1mds-node1]root@solaris:~#
```

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:

```
[qfs1mds-node1]root@solaris:~# cldevice set -p
default_fencing=nofencing-noscrub d4
[qfs1mds-node1]root@solaris:~# cldevice set -p
default_fencing=nofencing-noscrub d5
```

3. Next, [Mount the QFS File System on the Primary HA-COTC Node](#).

#### 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-node1` is the primary node and active metadata server:

```
[qfs1mds-node1]root@solaris:~#
```

- Place all data (`mr`) devices that are part of the file system in a `localonly` device group using 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:

```
[qfs1mds-node1]root@solaris:~# cldevicegroup set -d d4,d5 -p localonly=true \
-n node1mds mdsdevgrp
```

- Next, [Mount the QFS File System on the Primary HA-COTC Node](#).

### Mount the QFS File System on the Primary HA-COTC Node

- 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-node1` is the primary node and active metadata server:

```
[qfs1mds-node1]root@solaris:~#
```

- Back up the operating system's `/etc/vfstab` file.

```
[qfs1mds-node1]root@solaris:~# cp /etc/vfstab /etc/vfstab.backup
```

- 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 `qfs1` file system. The hyphen keeps the operating system from attempting to check file system integrity using UFS tools:

```
[qfs1mds-node1]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 -
...
qfs1 -
```

- 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

```
5. 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
[qfs1mds-node1]root@solaris:~#

```
6. 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:
- ```

[qfs1mds-node1]root@solaris:~# mkdir -p /global/ha-cotc/qfs1

```
7. Mount the high-availability shared file system on the primary node.
- ```

[qfs1mds-node1]root@solaris:~# mount /global/ha-cotc/qfs1

```
8. 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 Node](#)
- [Mount the QFS File System on the Secondary HA-COTC 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-node2` is the secondary node and the potential metadata server:

```

[qfs1mds-node2]root@solaris:~#

```

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 SAM-QFS 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-node1`:

```

[qfs1mds-node2]root@solaris:~# sam-fsd

```



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 SAM-QFS software. Normally, SAM-QFS 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.

```
[qfs1mds-node1]root@solaris:~# cd /opt/cluster/lib/rgm/rtreg/
[qfs1mds-node1]root@solaris:~# ln -s /opt/SUNWsamfs/sc/etc/SUNW.qfs SUNW.qfs
```

4. Create a resource group for the QFS metadata server using 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:

```
[qfs1mds-node1]root@solaris:~# clresourcegroup create -n
qfs1mds-node1,qfs1mds-node2 \
qfsrg
```

5. In the new resource group, set up a virtual hostname 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 and `virtualMDS` is the virtual hostname.

Use the same virtual hostname that you used in the hosts files for the shared file system. In the example, we create the virtual host `qfs1mds` in the `qfsrg` resource group:

```
[qfs1mds-node1]root@solaris:~# clreslogicalhostname create -g qfsrg qfs1mds
```

6. Add the QFS file-system resources to the resource group using 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 virtual hostname 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 `hasqfs` 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`:

```
[qfs1mds-node1]root@solaris:~# clresource create -g qfsrg -t SUNW.qfs \
-x QFSFileSystem=/global/ha-cotc/qfs1 -y Resource_dependencies=qfs1mds hasqfs
```

7. Bring the resource group online using 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:

```
[qfs1mds-node1]root@solaris:~# clresourcegroup manage qfsrg
[qfs1mds-node1]root@solaris:~# clresourcegroup online -emM qfsrg
```

8. 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, `samlmds-node1`:

```
[qfs1mds-node1]root@solaris:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name Node Name Suspended Status

qfsrg qfs1mds-node1 No Online
 qfs1mds-node2 No Offline
```

9. 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 resource group. Then use `clresourcegroup status` to check the result.

In the example, we move the `qfsrg` resource group to `qfs1mds-node2` and confirm that the resource group comes online on the specified node:

```
[qfs1mds-node1]root@solaris:~# clresourcegroup switch -n qfs1mds-node2 qfsrg
[qfs1mds-node1]root@solaris:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name Node Name Suspended Status

qfsrg qfs1mds-node1 No Offline
 qfs1mds-node2 No Online
```

10. 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 resource group. Then use `clresourcegroup status` to check the result.

In the example, we successfully move the `qfsrg` resource group back to `qfs1mds-node1`:

```
[qfs1mds-node1]root@solaris:~# clresourcegroup switch -n qfs1mds-node1 qfsrg
[qfs1mds-node1]root@solaris:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name Node Name Suspended Status

qfsrg qfs1mds-node1 No Online
 qfs1mds-node2 No Offline
```

11. 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 as `root`, display the device configuration for the cluster using the `/usr/global/bin/cldevice list -v` command, and make a note of the `/dev/rdisk/` path corresponding to the device identifier for each QFS data (`mr`) device.

In the example, the QFS data devices are `d4` and `d5`:

```
[qfs1mds-node1]root@solaris:~# cldevice list -v
DID Device Full Device Path

d1 qfs1mds-node1:/dev/rdisk/c0t0d0
d2 qfs1mds-node1:/dev/rdisk/c0t6d0
d3 qfs1mds-node1:/dev/rdisk/c1t1d0
d3 qfs1mds-node2:/dev/rdisk/c1t1d0
d4 qfs1mds-node1:/dev/rdisk/c1t2d0
d4 qfs1mds-node2:/dev/rdisk/c1t2d0
d5 qfs1mds-node1:/dev/rdisk/c1t3d0
d5 qfs1mds-node2:/dev/rdisk/c1t3d0
d6 qfs1mds-node2:/dev/rdisk/c0t0d0
d7 qfs1mds-node2:/dev/rdisk/c0t1d0
```

2. Log in to the client host of the HA-COTC cluster as `root`.

In the example, `qfs1client1` is the client host:

```
[qfs1mds-node1]root@solaris:~# ssh root@qfs1client1
[qfs1client1]root@solaris:~#
```

3. On the client host, retrieve the configuration information for the shared file system using the `samfsconfig /dev/rdisk/*` command.

The `samfsconfig /dev/rdisk/*` 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:

```
[qfs1client1]root@solaris:~# samfsconfig /dev/rdisk/*
Family Set 'qfs1' Created Thu Dec 21 07:17:00 2013
Missing slices
Ordinal 0
/dev/rdisk/c1t2d0s0 102 mr qfs1 -
/dev/rdisk/c1t3d0s1 103 mr qfs1 -
```

4. Compare the `/dev/rdisk/` paths reported for the data (`mr`) devices by `samfsconfig` with those on the server cluster node, as listed by the `cldevice list` command.

The `samfsconfig` and `cldevice list` commands should indicate the same devices, though the controller numbers (`cN`) 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 server maps cluster device identifiers `d4` and `d5` to the paths `/dev/rdisk/c1t2d0` and `/dev/rdisk/c1t3d0`:

```
d4 qfs1mds-node1:/dev/rdisk/c1t2d0
d5 qfs1mds-node1:/dev/rdisk/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 -
```

5. On the client, open the `/etc/opt/SUNWsamfs/mcf` file in a text editor, and enter a line for the HA-COTC shared file system that is identical to those in the `mcf` files on the metadata servers.

In the example, we use the `vi` editor to create an entry for the file QFS share system `qfs1` (equipment ordinal number 100):

```
[qfs1client1]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
qfs1 100 ma qfs1 - shared
```

6. In the `mcf` file, under the entry for the HA-COTC shared file system, add a line for the file system's metadata (`mm`) devices. Use the same equipment ordinal numbers, family set and device state parameters as are used in the `mcf` files on the metadata servers. But use the keyword `nodev` in the Equipment Identifier column.

```
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
qfs1 100 ma qfs1 - shared
nodev 101 mm qfs1 -
```

7. Copy the data (`mr`) device information from the `samfsconfig` output and paste it into the `/etc/opt/SUNWsamfs/mcf` file, under the entry for the HA-COTC metadata (`mm`) devices. 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
[qfs1client1]root@solaris:~#
```

8. Check the `mcf` file for errors. Use the command `/opt/SUNWsamfs/sbin/sam-fsd`, and correct any errors found.

The `sam-fsd` command reads SAM-QFS 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`:

```
[qfs1client1]root@solaris:~# sam-fsd
```

9. 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:

```
[qfs1client1]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 -
...
qfs1 - /global/ha-cotc/qfs1 samfs - no shared

```

```
:wq
```

```
[qfs1client1]root@solaris:~#
```

10. Create the mount point for the high-availability shared file system on the client.

```
[qfs1client1]root@solaris:~# mkdir -p /global/qfs1
```

11. Mount the high-availability shared file system on the client.

```
[qfs1client1]root@solaris:~# mount /global/qfs1
```

```
[qfs1client1]root@solaris:~# exit
```

```
[qfs1mds-node1]root@solaris:~#
```

12. Repeat this procedure until all HA-COTC clients have been configured.
13. If you plan on using the sideband database feature, go to "[Configuring the SAM-QFS Reporting Database](#)" on page 10-1.
14. Otherwise, go to "[Configuring Notifications and Logging](#)" on page 11-1.

## High-Availability SAM-QFS Shared Archiving File Systems

The High-Availability Storage Archive Manager (HA-SAM) configuration maintains the availability of an archiving file system by insuring that the QFS metadata server and the Storage Archive 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 SAM-QFS server on the surviving node and transfers control over running operations. Since the QFS file system and the SAM application's local storage directories are shared and already mounted, access to data and metadata remains uninterrupted.

The HA-SAM configuration insures file-system consistency in a clustered environment by sending all I/O through the active metadata server. You share the HA-SAM 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-SAM file system with clients using NFS. But you must insure that the shares are exported exclusively from the active metadata server node.

High-availability archiving file systems depend on two Solaris Cluster resource types:

- `SUNW.qfs`

If the primary host fails, the `SUNW.qfs` resource manages failover for the QFS metadata server and Storage Archive Manager applications. The `SUNW.qfs` software is included with the SAM-QFS software distribution (for more information, see the `SUNW.qfs` man page).

- `SUNW.HAStoragePlus`

If the primary host fails, the `SUNW.HAStoragePlus` resource manages failover of the Storage Archive Manager's local storage. The SAM application maintains volatile archiving information—job queues and removable media catalogs—in the server

host's local file system. `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*).

To configure instances of the required components and integrate them into a working HA-SAM archiving configuration, carry out the following tasks:

- [Create a SAM-QFS Shared File System Hosts File on Both HA-SAM Cluster Nodes](#)
- [Create Local Hosts Files on the HA-SAM Servers](#)
- [Configure an Active QFS Metadata Server on the Primary HA-SAM Cluster Node](#)
- [Configure a Potential QFS Metadata Server on the Secondary HA-SAM Cluster Node](#)
- [Configure Failover of HA-SAM Local Storage](#)
- [Configure Failover of the HA-SAM Metadata Servers](#)
- If required, configure High-Availability Network File System (HA-NFS) sharing.

Detailed procedures for setting up HA-NFS are included in the *Oracle Solaris Cluster Data Service for Network File System (NFS) Guide* that is included in the *Oracle Solaris Cluster* online documentation library.

## Create a SAM-QFS Shared File System Hosts File on Both HA-SAM Cluster Nodes

In an archiving SAM-QFS shared file system, you must configure a hosts file on the metadata servers, so that the hosts on both nodes 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-SAM cluster as `root`.

In the example, `sam1mds-node1` is the primary node:

```
[sam1mds-node1]root@solaris:~#
```

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` and the `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—`sam1mds-node1` and `sam1mds-node2`—that the cluster uses for data transport.

Note that the display has been abbreviated using ellipsis (`...`) marks:

```
[sam1mds-node1]root@solaris:~# cluster show
...
=== Cluster Nodes ===
```

```

Node Name: sam1mds-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: sam1mds-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

```

- 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.sam1` 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:

```

[sam1mds-node1]root@solaris:~# vi /etc/opt/SUNWsamfs/hosts.sam1
/etc/opt/SUNWsamfs/hosts.sam1
#
#Host Name Network Interface Server On/ Additional
#----- -
#----- -
#----- -
#----- -
#----- -

```

- 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 `sam1mds-node1` and `sam1mds-node2`, the hostnames of the cluster nodes that host the metadata servers for the file system:

```

#
#Host Name Network Interface Server On/ Additional
#----- -
#----- -
#----- -
#----- -
#----- -
sam1mds-node1
sam1mds-node2

```

- In the second column of each line, start supplying `Network Interface` information for the hosts listed in the `Host Name` column. Enter each HA-SAM cluster node's Solaris Cluster private hostname or private network address followed by a comma.

The HA-SAM 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:

```

#
#Host Name Network Interface Server On/ Additional
#----- -
#----- -
#----- -
#----- -
#----- -
sam1mds-node1 clusternode1-priv,
sam1mds-node2 clusternode2-priv,

```

- Following the comma in the second column of each line, enter the public hostname for the active metadata server followed by spaces.

The HA-SAM 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 `samlmnds-node1` to `samlmnds-node2` and vice versa), we use a virtual hostname—`samlmnds`—for both. Later, we will configure the Solaris Cluster software to always route requests for `samlmnds` to the active metadata server:

```
Server On/ Additional
#Host Name Network Interface Ordinal Off Parameters
#-----
samlmnds-node1 clusternode1-priv, samlmnds
samlmnds-node2 clusternode2-priv, samlmnds
```

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, `samlmnds-node1`, is the active metadata server, so it is ordinal 1 and the secondary node, `samlmnds-node2`, is ordinal 2:

```
Server On/ Additional
#Host Name Network Interface Ordinal Off Parameters
#-----
samlmnds-node1 clusternode1-priv, samlmnds 1
samlmnds-node2 clusternode2-priv, samlmnds 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
#-----
samlmnds-node1 clusternode1-priv, samlmnds 1 0
samlmnds-node2 clusternode2-priv, samlmnds 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
#-----
samlmnds-node1 clusternode1-priv, samlmnds 1 0 server
samlmnds-node2 clusternode2-priv, samlmnds 2 0
:wq
[samlmnds-node1]root@solaris:~#
```

10. Place a copy of the global `/etc/opt/SUNWsamfs/hosts.family-set-name` file on the potential metadata server.
11. Now, [Create Local Hosts Files on the HA-SAM Servers](#).

## Create Local Hosts Files on the HA-SAM Servers

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.

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-SAM cluster as `root`.

In the example, `sam1mds-node1` is the primary node:

```
[sam1mds-node1]root@solaris:~#
```

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 interfaces for the networks 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.sam1.local`, lists only cluster private addresses for the active and potential servers:

```
[sam1mds-node1]root@solaris:~# vi /etc/opt/SUNWsamfs/hosts.sam1.local
#
#Host Name Network Interface Server On/ Additional
#----- -
#----- -
#----- -
sam1mds-node1 clusternode1-priv 1 0 server
sam1mds-node2 clusternode2-priv 2 0
:wq
[sam1mds-node1]root@solaris:~#
```

3. Log in to the secondary cluster node as `root`.

In the example, `sam1mds-node2` is the secondary node:

```
[sam1mds-node1]root@solaris:~# ssh root@sam1mds-node2
Password:
[sam1mds-node2]root@solaris:~#
```

4. Using a text editor, create a local hosts file on the potential 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 interfaces for the networks 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.sam1.local`, lists only cluster private addresses for the active and potential servers:

```
[sam1mds-node2]root@solaris:~# vi /etc/opt/SUNWsamfs/hosts.sam1.local
#
#Host Name Network Interface Server On/ Additional
#----- -
#----- -
#----- -
sam1mds-node1 clusternode1-priv 1 0 server
sam1mds-node2 clusternode2-priv 2 0
:wq
[sam1mds-node2]root@solaris:~# exit
[sam1mds-node1]root@solaris:~#
```

5. Next, [Configure an Active QFS Metadata Server on the Primary HA-SAM Cluster Node](#).

## Configure an Active QFS Metadata Server on the Primary HA-SAM Cluster Node

1. Select the cluster node that will serve as both the primary node for the HA-SAM cluster and the active metadata server for the QFS shared file system. Log in as `root`.

In the example, `sam1mds-node1` is the primary node:

```
[sam1mds-node1]root@solaris:~#
```

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:

```
[sam1mds-node1]root@solaris:~# cldevice list -v
DID Device Full Device Path

d1 sam1mds-node1:/dev/rdisk/c0t0d0
d2 sam1mds-node1:/dev/rdisk/c0t6d0
d3 sam1mds-node1:/dev/rdisk/clt1d0
d3 sam1mds-node2:/dev/rdisk/clt1d0
d4 sam1mds-node1:/dev/rdisk/clt2d0
d4 sam1mds-node2:/dev/rdisk/clt2d0
d5 sam1mds-node1:/dev/rdisk/clt3d0
d5 sam1mds-node2:/dev/rdisk/clt3d0
d6 sam1mds-node2:/dev/rdisk/c0t0d0
d7 sam1mds-node2:/dev/rdisk/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 `sam1`. We configure device `d3` as the metadata device (equipment type `mm`), and use `d4` and `d5` as data devices (equipment type `mr`):

```
[sam1mids-node1]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
sam1 100 ma sam1 -
/dev/did/dsk/d3s0 101 mm sam1 -
/dev/did/dsk/d4s0 102 mr sam1 -
/dev/did/dsk/d5s1 103 mr sam1 -
```

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.

```
[sam1mids-node1]root@solaris:~# vi /etc/opt/SUNWsamfs/mcf
Equipment Equipment Equipment Family Device Additional
Identifier Ordinal Type Set State Parameters
#-----
sam1 100 ma sam1 - shared
/dev/did/dsk/d3s0 101 mm sam1 -
/dev/did/dsk/d4s0 102 mr sam1 -
/dev/did/dsk/d5s1 103 mr sam1 -
:wq
[sam1mids-node1]root@solaris:~#
```

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 SAM-QFS configuration files and initializes file systems. It will stop if it encounters an error. In the example, we check the `mcf` file on host `sam1mids-node1`:

```
[sam1mids-node1]root@solaris:~# sam-fsd
```

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 a SAM-QFS file system with the specified properties.

```
[sam1mids-node1]root@solaris:~# sammkfs -S sam1
```

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 `sam1` file system. The hyphen keeps the operating system from attempting to check file system integrity using UFS tools:

```
[sam1mids-node1]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 -
...
sam1 -
```

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-sam/sam1`:

```
#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 -
...
sam1 - /global/ha-sam/sam1
```

9. Populate the remaining fields of the `/etc/vfstab` file record as you would for any SAM-QFS 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 -
...
sam1 - /global/ha-sam/sam1 samfs - no shared
:wq
[samlmds-node1]root@solaris:~#
```

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:

```
[samlmds-node1]root@solaris:~# mkdir -p /global/ha-sam/sam1
```

11. Mount the high-availability shared file system on the primary node.

```
[samlmds-node1]root@solaris:~# mount /global/ha-sam/sam1
```

12. Next, [Configure a Potential QFS Metadata Server on the Secondary HA-SAM Cluster Node](#).

## Configure a Potential QFS Metadata Server on the Secondary HA-SAM 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.

1. Log in to the secondary node of the HA-SAM cluster as `root`.

In the example, `samlmds-node2` is the secondary node:

```
[samlmds-node2]root@solaris:~#
```

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 SAM-QFS configuration files and initializes file systems. It will stop if it encounters an error. In the example, we check the `mcf` file on host `sam1mds-node1`:

```
[sam1mds-node2]root@solaris:~# sam-fsd
```

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 a SAM-QFS file system with the specified properties.

```
[sam1mds-node2]root@solaris:~# sammkfs sam1
```

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:

```
[sam1mds-node2]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 -
...
sam1 - /global/ha-samsam1 samfs - no shared
:wq
[sam1mds-node2]root@solaris:~#
```

6. Create the mount point for the high-availability shared file system on the secondary node.

```
[sam1mds-node2]root@solaris:~# mkdir -p /global/ha-sam/sam1
```

7. Mount the high-availability shared file system on the secondary node.

```
[sam1mds-node2]root@solaris:~# mount /global/ha-sam/sam1
```

8. Now [Configure Failover of HA-SAM Local Storage](#).

## Configure Failover of HA-SAM Local Storage

The Storage Archive Manager software maintains state information for archiving operations in the metadata server's local storage. By default, catalogs of archival media and queues of staging jobs reside in `/var/opt/SUNWsamfs/catalog` and `/var/opt/SUNWsamfs/stager`. Since local storage is no longer available to a surviving node following failover, we need to configure a high-availability storage resource to hold this data, relocate the subdirectories to the new resource, and provide symbolic links to the new file system in the default locations.

`HAStoragePlus` is the Solaris Cluster resource type that defines the storage resource that we need. This resource type manages dependencies between disk device groups, cluster file systems, and local file systems, and 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).

To create and configure the `HAStoragePlus` resource, proceed as follows:

1. Log in to the active metadata server (the primary cluster node of the HA-SAM cluster) as root.

In the example, the primary node, `samlmds-node1`, is the active metadata server:

```
[samlmds-node1]root@solaris:~#
```

2. Register the `SUNW.HAStoragePlus` resource type as part of the cluster configuration. Use the Solaris Cluster command `clresourcetype register SUNW.HAStoragePlus`.

```
[samlmds-node1]root@solaris:~# clresourcetype register SUNW.HAStoragePlus
```

3. Create the resource 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 resource group. This resource group will hold all cluster resources required by the HA-SAM configuration.
- `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 queue files.
- *resourcename* is the name that you have chosen for the resource itself.

In the example, we create a resource named `samlocalfs` of type `SUNW.HAStoragePlus` and add it to the resource group `samrg`. Then we configure the resource by setting the `SUNW.HAStoragePlus` extension properties: we set `FilesystemMountPoints` to `/sam_shared` and `AffinityOn` to `TRUE`:

```
[samlmds-node1]root@solaris:~# clresource create -g samrg -t \
SUNW.HAStoragePlus -x FilesystemMountPoints=/sam_shared \
-x AffinityOn=TRUE samlocalfs
```

4. Copy the `catalog/` and `stager/` directories from their default location in `/var/opt/SUNWsamfs/` to a temporary location.

In the example, we recursively copy the directories to `/var/tmp/`:

```
[samlmds-node1]root@solaris:~# cp -r /var/opt/SUNWsamfs/catalog \
/var/tmp/catalog
[samlmds-node1]root@solaris:~# cp -r /var/opt/SUNWsamfs/stager /var/tmp/stager
```

5. Delete `catalog/` and `stager/` directories from the default location, `/var/opt/SUNWsamfs/`.

```
[samlmds-node1]root@solaris:~# rm -rf /var/opt/SUNWsamfs/catalog
[samlmds-node1]root@solaris:~# rm -rf /var/opt/SUNWsamfs/stager
```

6. Create new `catalog/` and `stager/` directories under the mount point of the high-availability `HAStoragePlus` file system.

In the example, we create the new directories under the `/sam_shared` mount point:

```
[samlmds-node1]root@solaris:~# mkdir /sam_shared/catalog
[samlmds-node1]root@solaris:~# mkdir /sam_shared/stager
```

7. In the `/var/opt/SUNWsamfs/` directory on the active metadata server, create a symbolic link, `catalog`, to the new `catalog/` directory on the `HAStoragePlus` mount point.

When the Storage Archive Manager application looks for data in its local file system, the symbolic link will automatically redirect it to the highly available file system. In the example, we create a `catalog` link that points to the new location `/sam_shared/catalog`:

```
[sam1mds-node1]root@solaris:~# ln -s /sam_shared/catalog \
/var/opt/SUNWsamfs/catalog
```

8. In the `/var/opt/SUNWsamfs/` directory on the active metadata server, create a symbolic link, `stager`, to the new `stager/` directory on the `HAStoragePlus` mount point.

In the example, we create a `stager` link that points to the new location `/sam_shared/stager`:

```
[sam1mds-node1]root@solaris:~# ln -s /sam_shared/stager \
/var/opt/SUNWsamfs/stager
```

9. On the active metadata server, make sure that symbolic links have replaced the default `/var/opt/SUNWsamfs/catalog` and `/var/opt/SUNWsamfs/stager` locations and that the links point to the new locations in the high-availability file system.

In the example, the links are correct:

```
[sam1mds-node1]root@solaris:~# ls -l /var/opt/SUNWsamfs/catalog
lrwxrwxrwx 1 root other ... /var/opt/SUNWsamfs/catalog -> /sam_shared/catalog
[sam1mds-node1]root@solaris:~# ls -l /var/opt/SUNWsamfs/stager
lrwxrwxrwx 1 root other ... /var/opt/SUNWsamfs/stager -> /sam_shared/stager
```

10. Copy the contents of the `catalog/` and `stager/` directories from the temporary location to the new, high-availability, shared file system.

In the example, we create a `stager` link that points to the new location `/sam_shared/stager`:

```
[sam1mds-node1]root@solaris:~# cp -rp /var/tmp/catalog/* \
/var/opt/SUNWsamfs/catalog
[sam1mds-node1]root@solaris:~# cp -rp /var/tmp/stager/* \
/var/opt/SUNWsamfs/stager
```

11. Log in to the potential metadata server (the secondary node of the HA-SAM cluster) as `root`.

In the example, `sam1mds-node2` is the secondary node:

```
[sam1mds-node2]root@solaris:~#
```

12. In the `/var/opt/SUNWsamfs/` directory on the potential metadata server, create a symbolic link, `catalog`, to the new `catalog/` directory on the `HAStoragePlus` mount point.

When the Storage Archive Manager application looks for data in its local file system, the symbolic link will automatically redirect it to the highly available file system. In the example, we create a `catalog` link that points to the new location `/sam_shared/catalog`:

```
[sam1mds-node2]root@solaris:~# ln -s /sam_shared/catalog \
/var/opt/SUNWsamfs/catalog
```

13. In the `/var/opt/SUNWsamfs/` directory on the potential metadata server, create a symbolic link, `stager`, to the new `stager/` directory on the HAStoragePlus mount point.

In the example, we create a `stager` link that points to the new location `/sam_shared/stager`:

```
[sam1mds-node2]root@solaris:~# ln -s /sam_shared/stager \
/var/opt/SUNWsamfs/stager
```

14. On the potential metadata server, make sure that symbolic links have replaced the default `/var/opt/SUNWsamfs/catalog` and `/var/opt/SUNWsamfs/stager` locations, and make sure that the links point to the new locations in the high-availability file system.

In the example, the links are correct:

```
[sam1mds-node2]root@solaris:~# ls -l /var/opt/SUNWsamfs/catalog
lrwxrwxrwx 1 root other ... /var/opt/SUNWsamfs/catalog -> /sam_shared/catalog
[sam1mds-node2]root@solaris:~# ls -l /var/opt/SUNWsamfs/stager
lrwxrwxrwx 1 root other ... /var/opt/SUNWsamfs/stager -> /sam_shared/stager
```

15. Now [Configure Failover of the HA-SAM Metadata Servers](#).

## Configure Failover of the HA-SAM Metadata Servers

When you host a SAM-QFS 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 SAM-QFS software (see the `SUNW.qfs` man page for details). To create and configure the resource for an HA-SAM configuration, proceed as follows:

1. Log in to the primary cluster node of the HA-SAM cluster as `root`.

In the example, the primary node is `sam1mds-node1`:

```
[sam1mds-node1]root@solaris:~#
```

2. Define the resource type, `SUNW.qfs`, for the Solaris Cluster software using the command `clresourcetype register SUNW.qfs`.

```
[sam1mds-node1]root@solaris:~# clresourcetype register SUNW.qfs
```

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 SAM-QFS software. Normally, SAM-QFS 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.

```
[qfs1mds-node1]root@solaris:~# cd /opt/cluster/lib/rgm/rtreg/
[qfs1mds-node1]root@solaris:~# ln -s /opt/SUNWsamfs/sc/etc/SUNW.qfs SUNW.qfs
```

4. In the new resource group, set up a virtual hostname 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 and `virtualMDS` is the virtual hostname.

Use the same virtual hostname that you used in the hosts files for the shared file system. In the example, we add the virtual hostname `samlmids` to the `samrg` resource group:

```
[samlmids-node1]root@solaris:~# clreslogicalhostname create -g samrg samlmds
```

5. Add the SAM-QFS file-system resources to the resource group using the command `clresource create -g groupname -t SUNW.qfs -x QFSFileSystem=mount-point -y Resource_dependencies=virtualMDS resource-name`, where:

- `groupname` is the name that you have chosen for the resource group. This resource group will hold all cluster resources required by the HA-SAM configuration.
- `SUNW.qfs` is the Solaris Cluster resource type that supports failover of the metadata servers and the Storage Archive Manager application.
- `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 virtual hostname of the active metadata server.
- `resource-name` is the name that you have chosen for the resource itself.

In the example, we create a resource named `hasam` of type `SUNW.qfs` in the resource group `samrg`. We set the `SUNW.qfs` extension property `QFSFileSystem` to the `/global/ha-sam/sam1` mount point, and set the standard property `Resource_dependencies` to the virtual hostname for the active metadata server, `samlmids`:

```
[samlmids-node1]root@solaris:~# clresource create -g samrg -t SUNW.qfs \
-x QFSFileSystem=/global/ha-sam/sam1 -y Resource_dependencies=samlmids hasam
```

6. Create a dependency between the `SUNW.HAStoragePlus` resource that supports failover of Storage Archive Manager local files and the `SUNW.qfs` resource that manages failover of the active metadata server. Use the Solaris Cluster command `clresource set -p Resource_dependencies=dependency resource-name`, where `resource-name` is the name of the `SUNW.qfs` resource and `dependency` is name of the `SUNW.HAStoragePlus` resource.

In the example, we specify `samlocalfs` as a dependency of the `hasam` resource:

```
[samlmids-node1]root@solaris:~# clresource set -p Resource_dependencies=samlocal
fs hasam
```

7. Create a dependency between the `SUNW.qfs` resource that manages failover of the active metadata server and the entire HA-SAM resource group. Use the Solaris Cluster command `clresource set -p Resource_dependencies=dependency resource-name`, where `resource-name` is the name of the HA-SAM resource group and `dependency` is name of the `SUNW.qfs` resource.

In the example, we specify `hasam` as a dependency of the `samrg` resource group:

```
[samlmids-node1]root@solaris:~# clresource set -p Resource_dependencies=hasam \
samrg
```

8. 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 QFS resource group.

In the example, we bring the `samrg` resource group online:

```
[sam1mds-node1]root@solaris:~# clresourcegroup manage samrg
[sam1mds-node1]root@solaris:~# clresourcegroup online -emM samrg
```

9. Make sure that the QFS resource group is online. Use the Solaris Cluster `clresourcegroup status` command.

In the example, the `samrg` resource group is online on the primary node, `sam1mds-node1`:

```
[sam1mds-node1]root@solaris:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name Node Name Suspended Status

samrg sam1mds-node1 No Online
 sam1mds-node2 No Offline
```

10. 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 groupname`, where `node2` is the name of the secondary node and `groupname` is the name that you have chosen for the resource group. Then use `clresourcegroup status` to check the result.

In the example, we move the `samrg` resource group to `sam1mds-node2` and confirm that the resource group comes online on the specified node:

```
sam1mds-node1]root@solaris:~# clresourcegroup switch -n sam1mds-node2 samrg
[sam1mds-node1]root@solaris:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name Node Name Suspended Status

samrg sam1mds-node1 No Offline
 sam1mds-node2 No Online
```

11. 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 resource group. Then use `clresourcegroup status` to check the result.

In the example, we successfully move the `samrg` resource group back to `sam1mds-node1`:

```
sam1mds-node1]root@solaris:~# clresourcegroup switch -n sam1mds-node1 samrg
[sam1mds-node1]root@solaris:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name Node Name Suspended Status

samrg sam1mds-node1 No Online
 sam1mds-node2 No Offline
```

12. If required, configure High-Availability Network File System (HA-NFS) sharing now.

Detailed procedures for setting up HA-NFS are included in the *Oracle Solaris Cluster Data Service for Network File System (NFS) Guide* that is included in the *Oracle Solaris Cluster* online documentation library.

13. If you plan on using the sideband database feature, go to "[Configuring the SAM-QFS Reporting Database](#)" on page 10-1.
14. 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. Since the QFS file system is shared and already mounted on all nodes while I/O is coordinated through Oracle RAC, 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 a QFS Shared File System Hosts File on All SC-RAC Cluster Nodes](#)
- [Create Local Hosts Files on the QFS Servers and Clients Outside the HA-COTC Cluster](#)
- [Configure an Active QFS Metadata Server on the Primary SC-RAC Cluster Node or Configure QFS Metadata Servers on SC-RAC Nodes Using Software RAID Storage](#)
- [Configure a Potential QFS Metadata Server on the Remaining SC-RAC Cluster Nodes](#)
- [Configure Failover of the SC-RAC Metadata Servers](#)
- If required, configure Network File System (NFS) shares, as described in ["Accessing File Systems from Multiple Hosts Using NFS and SMB/CIFS"](#) on page 7-29. High-Availability NFS (HA-NFS) is *not* supported.

### Create a QFS Shared File System Hosts File 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`:

```
[qfs1rac-node1]root@solaris:~#
```

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` and the `Transport Adapter` name and `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:

```
[qfs1rac-node1]root@solaris:~# 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 some optional headings to show the columns in the hosts table, starting each line with a hash sign (#) to indicate a comment:

```
[qfs1rac-node1]root@solaris:~# vi /etc/opt/SUNWsamfs/hosts.qfs1rac
/etc/opt/SUNWsamfs/hosts.qfs1rac
#
#Host Name Network Interface Server On/ Additional
#----- -----
#----- ----- 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`.

```
#
#Host Name Network Interface Server On/ Additional
#----- -----
#----- -----
qfs1rac-node1
```

```
qfs1rac-node2
qfs1rac-node3
```

- 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,
```

- 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 virtual hostname, `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
```

- 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, `qfs1rac-node1`, is the active metadata server, so it is ordinal 1, the secondary 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
```

- 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

```

- 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
[qfs1rac-node1]root@solaris:~#

```

- Place a copy of the global `/etc/opt/SUNWsamfs/hosts.family-set-name` file on each node in the SC-RAC cluster.
- Now, [Configure an Active QFS Metadata Server on the Primary SC-RAC Cluster Node](#).

## Configure an Active QFS Metadata Server on the Primary SC-RAC Cluster Node

- 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`:

```
[qfs1rac-node1]root@solaris:~#
```

- 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 both nodes. So we select from devices `d3`, `d4`, and `d5` when configuring the high-availability QFS shared file system:

```

[qfs1rac-node1]root@solaris:~# cldevice list -v
DID Device Full Device Path

d1 qfs1rac-node1:/dev/rdisk/c0t0d0
d2 qfs1rac-node1:/dev/rdisk/c0t6d0
d3 qfs1rac-node1:/dev/rdisk/clt1d0
d3 qfs1rac-node2:/dev/rdisk/clt1d0
d3 qfs1rac-node3:/dev/rdisk/clt1d0
d4 qfs1rac-node1:/dev/rdisk/clt2d0
d4 qfs1rac-node2:/dev/rdisk/clt2d0
d4 qfs1rac-node3:/dev/rdisk/clt2d0
d5 qfs1rac-node1:/dev/rdisk/clt3d0
d5 qfs1rac-node2:/dev/rdisk/clt3d0

```

```

d5 qfs1rac-node3:/dev/rdisk/clt3d0
d6 qfs1rac-node2:/dev/rdisk/c0t0d0
d7 qfs1rac-node2:/dev/rdisk/c0t1d0
d8 qfs1rac-node3:/dev/rdisk/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`):

```

[qfs1rac-node1]root@solaris:~# 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.

```

[qfs1rac-node1]root@solaris:~# 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
[qfs1rac-node1]root@solaris:~#

```

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 SAM-QFS 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`:

```

[qfs1rac-node1]root@solaris:~# sam-fsd

```

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.

```

[qfs1rac-node1]root@solaris:~# sammkfs -S qfs1rac

```

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, 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:

```
[qfs1rac-node1]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 -
...
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 and `-` (*hyphen*) and `no` in the fifth and sixth columns.

```
#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
[qfs1rac-node1]root@solaris:~#
```

10. In the sixth 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
#-----
/devfs - /devfs devfs - no -
/proc - /proc proc - no -
...
qfs1rac - /global/sc-rac/qfs1rac samfs - no
shared, ...aplease=300
:wq
[qfs1rac-node1]root@solaris:~#
```

11. Create mount point for the high-availability shared file system.

```
[qfs1rac-node1]root@solaris:~# mkdir -p /global/sc-rac/qfs1rac
```

12. Mount the high-availability shared file system on the primary node.

```
[qfs1rac-node1]root@solaris:~# mount /global/sc-rac/qfs1rac
```

13. 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 the potential metadata servers. 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.

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:

```
[qfs1rac-node2]root@solaris:~#
```

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 SAM-QFS 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:

```
[qfs1rac-node2]root@solaris:~# sam-fsd
```

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:

```
[qfs1rac-node2]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 -
...
qfs1rac - /global/sc-rac/qfs1rac samfs - no

```

5. In the sixth 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 ...,aplease=300
:wq
[qfs1rac-node2]root@solaris:~#

```

6. Create the mount point for the high-availability shared file system on the secondary node.

```
[qfs1rac-node2]root@solaris:~# mkdir -p /global/sc-rac/qfs1rac
```

7. Mount the high-availability shared file system on the secondary node.

```
[qfs1rac-node2]root@solaris:~# mount /global/sc-rac/qfs1rac
```

8. Now [Configure Failover of the SC-RAC Metadata Servers](#).

## Configure Failover of the SC-RAC Metadata Servers

When you host a SAM-QFS 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 SAM-QFS 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`:

```
[qfs1rac-node1]root@solaris:~#
```

2. Define the QFS resource type, `SUNW.qfs`, for the Solaris Cluster software. Use the command `clresource type registerSUNW.qfs`.

```
[qfs1rac-node1]root@solaris:~# clresource type registerSUNW.qfs
```

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 SAM-QFS software. Normally, SAM-QFS 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.

```
[qfs1rac-node1]root@solaris:~# cd /opt/cluster/lib/rgm/rtreg/
```

```
[qfs1rac-node1]root@solaris:~# ln -s /opt/SUNWsamfs/sc/etc/SUNW.qfs SUNW.qfs
```

4. Create a resource group for the QFS metadata server using the Solaris Cluster command `clresource group create -n node-list group-name`, where `node-list` is a comma-delimited list of the two 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:

```
[qfs1rac-node1]root@solaris:~# clresource group create \
-n qfs1rac-node1,qfs1rac-node2 qfsracrg
```

5. In the new resource group, set up a virtual hostname 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 virtual hostname.

Use the same virtual hostname 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:

```
[qfs1rac-node1]root@solaris:~# clreslogicalhostname create \
-g qfsracrg qfs1rac-mds
```

6. Add the QFS file-system resources to the resource group using 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 virtual hostname 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 `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, and set the standard property

Resource\_dependencies to the logical host for the active metadata server, qfs1rac-mds:

```
[qfs1rac-node1]root@solaris:~# create -g qfsracrg -t SUNW.qfs \
-x QFSFileSystem=/global/sc-rac/qfs1rac \
-y Resource_dependencies=qfs1rac-mds scrac
```

7. 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:

```
[qfs1rac-node1]root@solaris:~# clresourcegroup manage qfsracrg
[qfs1rac-node1]root@solaris:~# clresourcegroup online -emM qfsracrg
```

8. 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`:

```
[qfs1rac-node1]root@solaris:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name Node Name Suspended Status

qfsracrg qfs1rac-node1 No Online
 qfs1rac-node2 No Offline
 qfs1rac-node3 No Offline
```

9. 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 resource group. Then use `clresourcegroup status` to check the result.

In the example, we move the resource group to `qfs1rac-node2` and `qfs1rac-node3`, confirming that the resource group comes online on the specified node:

```
[qfs1rac-node1]root@solaris:~# clresourcegroup switch -n qfs1rac-node2 qfsracrg
[qfs1rac-node1]root@solaris:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name Node Name Suspended Status

qfsracrg qfs1rac-node1 No Offline
 qfs1rac-node2 No Online
 qfs1rac-node3 No Offline

[qfs1rac-node1]root@solaris:~# clresourcegroup switch -n qfs1rac-node3 qfsracrg
[qfs1rac-node1]root@solaris:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name Node Name Suspended Status

qfsracrg qfs1rac-node1 No Offline
 qfs1rac-node2 No Offline
 qfs1rac-node3 No Online

[qfs1rac-node1]root@solaris:~#
```

10. 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 resource group. Then use `clresourcegroup status` to check the result.

In the example, we successfully move the `samr` resource group back to `qfs1rac-node1`:

```
[qfs1rac-node1]root@solaris:~# clresourcegroup switch -n qfs1rac-node1 qfsracrg
[qfs1rac-node1]root@solaris:~# clresourcegroup status
=== Cluster Resource Groups ===
Group Name Node Name Suspended Status

samr qfs1rac-node1 No Online
 qfs1rac-node2 No Offline
 qfs1rac-node3 No Offline
[qfs1rac-node1]root@solaris:~#
```

11. If you plan on using the sideband database feature, go to "[Configuring the SAM-QFS Reporting Database](#)" on page 10-1.
12. Otherwise, go to "[Configuring Notifications and Logging](#)" on page 11-1.

## Configure QFS Metadata Servers on SC-RAC Nodes Using 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 Solaris 11, but the Solaris Cluster 4 software continues to support Solaris Volume Manager on Solaris 11. So, to use the software, you must download and install the version that was included with the Solaris 10 9/10 release. For each node in the cluster, proceed as follows:

1. Log in to the node as `root`.

In the example, we configure cluster node `qfs2rac-node1`:

```
[qfs2rac-node1]root@solaris:~#
```

2. Check for locally available Solaris Volume Manager (SVM) packages. Use the command `pkg info svm`.

```
[qfs2rac-node1]root@solaris:~# 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
[qfs2rac-node1]root@solaris:~#
```

3. If no packages are found locally, check the Solaris Image Packaging System (IPS) repository. Use the command `pkg -r svm`.

```
[qfs2rac-node1]root@solaris:~# 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
[qfs2rac-node1]root@solaris:~#
```

4. Install the package using the command `pkg install storage/svm`:

```
[qfs2rac-node1]root@solaris:~# 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
[qfs2rac-node1]root@solaris:~#
```

5. When the installation finishes, check the location of `metadb`. Use the command `which metadb`.

```
[qfs2rac-node1]root@solaris:~# which metadb
/usr/sbin/metadb
[qfs2rac-node1]root@solaris:~#
```

6. Check the installation. Use the command `metadb`.

```
[qfs2rac-node1]root@solaris:~# metadb
[qfs2rac-node1]root@solaris:~#
```

7. If `metadb` returns an error, see if the `kernel/drv/md.conf` file exists.

```
[qfs2rac-node1]root@solaris:~# metadb
metadb: <HOST>: /dev/md/admin: No such file or directory
[qfs2rac-node1]root@solaris:~# ls -l /kernel/drv/md.conf
-rw-r--r-- 1 root sys 295 Apr 26 15:07 /kernel/drv/md.conf
[qfs2rac-node1]root@solaris:~#
```

- 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.

The content of the file should look like this:

```
[qfs2rac-node1]root@solaris:~# vi kernel/drv/md.conf
#####
#pragma ident "@(#)md.conf 2.1 00/07/07 SMI"
#
Copyright (c) 1992-1999 by Oracle Microsystems, Inc.
All rights reserved.
#
name="md" parent="pseudo" nmd=128 md_nsets=4;
#####
:wq
[qfs2rac-node1]root@solaris:~# chown root:sys kernel/drv/md.conf
[qfs2rac-node1]root@solaris:~# chmod 644
[qfs2rac-node1]root@solaris:~#
```

- 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:

```
[qfs2rac-node1]root@solaris:~# update_drv -f md
[qfs2rac-node1]root@solaris:~# ls -l /dev/md/admin
lrwxrwxrwx 1 root root 31 Apr 20 10:12 /dev/md/admin ->
../../../../devices/pseudo/md@0:admin
```

- Next, [Create Solaris Volume Manager Multi-Owner Disk Groups](#).

## Create Solaris Volume Manager Multi-Owner Disk Groups

- 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`:

```
[qfs2rac-node1]root@solaris:~#

[qfs2rac-node1]root@solaris:~# ssh root@qfs2rac-node2
Password:
[qfs2rac-node2]root@solaris:~#

[qfs2rac-node1]root@solaris:~# ssh root@qfs2rac-node3
Password:
[qfs2rac-node3]root@solaris:~#
```

- If you are using Oracle Solaris Cluster 4.x on Solaris 11.x and have not already done so, [Install Solaris Volume Manager on Solaris 11](#) on each node before proceeding further.

Solaris Volume Manager is not installed on Solaris 11 by default.

- 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 `cXtYdYsZ`.

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:

```
[qfs2rac-node1]root@solaris:~# metadb -a -f -c3 /dev/rdisk/c0t0d0
```

```
[qfs2rac-node2]root@solaris:~# metadb -a -f -c3 /dev/rdisk/c0t6d0
```

```
[qfs2rac-node3]root@solaris:~# metadb -a -f -c3 /dev/rdisk/c0t4d0
```

4. Create a Solaris Volume Manager multi-owner disk group on one node. Use the command `metaset -sdiskset -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:

```
[qfs2rac-node1]root@solaris:~# 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`.

```
[qfs2rac-node1]root@solaris:~# cldevice list -n -v
DID Device Full Device Path

d13 qfs2rac-node1:/dev/rdsk/c6t600C0FF00000000000332B62CF3A6B00d0
d14 qfs2rac-node1:/dev/rdsk/c6t600C0FF00000000000876E950F1FD9600d0
d15 qfs2rac-node1:/dev/rdsk/c6t600C0FF00000000000876E9124FAF9C00d0
...
[qfs2rac-node1]root@solaris:~#
```

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`.

```
[qfs2rac-node1]root@solaris:~# cldevice list -n -v
DID Device Full Device Path

d13 qfs2rac-node1:/dev/rdsk/c6t600C0FF00000000000332B62CF3A6B00d0
d14 qfs2rac-node1:/dev/rdsk/c6t600C0FF00000000000876E950F1FD9600d0
d15 qfs2rac-node1:/dev/rdsk/c6t600C0FF00000000000876E9124FAF9C00d0
d16 qfs2rac-node1:/dev/rdsk/c6t600C0FF00000000000332B28488B5700d0
d17 qfs2rac-node1:/dev/rdsk/c6t600C0FF0000000000086DB474EC5DE900d0
d18 qfs2rac-node1:/dev/rdsk/c6t600C0FF00000000000876E975EDA6A000d0
d19 qfs2rac-node1:/dev/rdsk/c6t600C0FF0000000000086DB47E331ACF00d0
d20 qfs2rac-node1:/dev/rdsk/c6t600C0FF00000000000876E9780ECA8100d0
d21 qfs2rac-node1:/dev/rdsk/c6t600C0FF000000000004CAD5B68A7A100d0
d22 qfs2rac-node1:/dev/rdsk/c6t600C0FF0000000000086DB43CF85DA800d0
d23 qfs2rac-node1:/dev/rdsk/c6t600C0FF000000000004CAD7CC3CDE500d0
d24 qfs2rac-node1:/dev/rdsk/c6t600C0FF0000000000086DB4259B272300d0
...
[qfs2rac-node1]root@solaris:~#
```

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`:

```
[qfs2rac-node1]root@solaris:~# 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 ...
[qfs2rac-node1]root@solaris:~#
```

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  $d_1$ ,  $d_2$ ,  $d_3$ ,  $d_4$ , and so on. Then we name each pair of RAID-0 volumes for the RAID-1 mirror that includes it:  $d_{10}$  and  $d_{11}$ ,  $d_{20}$  and  $d_{21}$ ,  $d_{30}$  and  $d_{31}$ ,  $d_{40}$  and  $d_{41}$ , etc.

2. Log in to the node where you created the multi-owner disk set as `root`.

In the examples above, we created the disk set on `qfs2rac-node1`:

```
[qfs2rac-node1]root@solaris:~#
```

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`:

```
[qfs2rac-node1]root@solaris:~# metainit -s dataset1 d10 1 1 /dev/did/dsk/d21s0
```

4. Create the remaining RAID-0 logical volumes.

```
[qfs2rac-node1]root@solaris:~# metainit -s dataset1 d11 1 1 /dev/did/dsk/d13s0
[qfs2rac-node1]root@solaris:~# metainit -s dataset1 d20 1 1 /dev/did/dsk/d14s0
[qfs2rac-node1]root@solaris:~# metainit -s dataset1 d21 1 1 /dev/did/dsk/d17s0
[qfs2rac-node1]root@solaris:~# metainit -s dataset1 d30 1 1 /dev/did/dsk/d23s0
[qfs2rac-node1]root@solaris:~# metainit -s dataset1 d31 1 1 /dev/did/dsk/d16s0
[qfs2rac-node1]root@solaris:~# metainit -s dataset1 d40 1 1 /dev/did/dsk/d15s0
[qfs2rac-node1]root@solaris:~# metainit -s dataset1 d41 1 1 /dev/did/dsk/d19s0
...
```

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`:

```
[qfs2rac-node1]root@solaris:~# metainit -s dataset1 d1 -m d10
```

```
[qfs2rac-node1]root@solaris:~# metattach -s dataset1 d11 d1
```

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:

```
[qfs2rac-node1]root@solaris:~# metattach -s dataset1 d11 d1
```

7. Create the remaining mirrors.

In the example, we create mirrors, d2, d3, d4, etc.

```
[qfs2rac-node1]root@solaris:~# metainit -s dataset1 d2 -m d20
[qfs2rac-node1]root@solaris:~# metattach -s dataset1 d21 d2
[qfs2rac-node1]root@solaris:~# metainit -s dataset2 d3 -m d30
[qfs2rac-node1]root@solaris:~# metattach -s dataset2 d31 d3
[qfs2rac-node1]root@solaris:~# metainit -s dataset2 d4 -m d40
[qfs2rac-node1]root@solaris:~# metattach -s dataset2 d41 d4
...
```

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:
  - 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:

```
[qfs2rac-node1]root@solaris:~# metainit -s dataset1 d53 -p d1 500g
[qfs2rac-node1]root@solaris:~# 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 a QFS Shared File System Hosts File on All SC-RAC Cluster Nodes](#)" on page 9-34. 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:

```
[qfs2rac-node1]root@solaris:~#
```

- 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.

```
[qfs2rac-node1]root@solaris:~# 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
[qfs2rac-node1]root@solaris:~#
```

- Check the `mcf` file for errors. Use the command `/opt/SUNWsamfs/sbin/sam-fsd`, and correct any errors found.

The `sam-fsd` command reads SAM-QFS 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`:

```
[qfs2rac-node1]root@solaris:~# sam-fsd
```

- 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.

```
[qfs2rac-node1]root@solaris:~# sammkfs -S qfs2rac
```

- 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:

```
[qfs2rac-node1]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 -
...
qfs2rac -
```

- In the third column of the `/etc/vfstab` file, enter the mount point of the file system relative to the cluster, the file system type (`samfs`), the `fsck` pass option (-),

and the mount-at-boot option (no). 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 sixth 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 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
[qfs2rac-node1]root@solaris:~#
```

9. Create the mount point for the high-availability shared file system.

```
[qfs2rac-node1]root@solaris:~# mkdir -p /global/sc-rac/qfs2rac
```

10. Mount the high-availability shared file system on the primary node.

```
[qfs2rac-node1]root@solaris:~# mount /global/sc-rac/qfs2rac
```

11. Set up the second node using the procedure "[Configure a Potential QFS Metadata Server on the Remaining SC-RAC Cluster Nodes](#)" on page 9-40.

12. Configure failover using the procedure "[Configure Failover of the SC-RAC Metadata Servers](#)" on page 9-41. Then return here.

You have successfully configured an SC-RAC configuration using Solaris Volume Manager mirrored volumes.

13. If you plan on using the sideband database feature, go to "[Configuring the SAM-QFS Reporting Database](#)" on page 10-1.
14. Otherwise, go to "[Configuring Notifications and Logging](#)" on page 11-1.

---

## Configuring the SAM-QFS Reporting Database

SAM-QFS 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 SAM-QFS 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 an existing or specially created recovery point file generated by the `samfsdump` command. SAM-QFS daemon processes then update the database automatically as the file systems 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 this procedure 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 SAM-QFS metadata server host or on an independent Solaris or Linux host.

In the example, we install MySQL on the Solaris host `samsql`:

```
[samsql]root@solaris:~#
```

3. Download and install the MySQL server software on a host, as directed in the *MySQL Reference Manual*. Enable automatic startup.
4. Connect to the MySQL server using the `mysql` client and the `root` user account. Use the command `mysql --user=root -p` and, when prompted, enter the password that you assigned to the `root` user during installation.

The `mysql` command shell starts:

```
[samsql]root@solaris:~# mysql --user=root -p
Enter Password:
mysql>
```

5. Create the SAM-QFS MySQL user. Use the command `CREATE USER 'user_name'@'host_name' IDENTIFIED BY 'user-password'`, where:
  - `user_name` is the name of the SAM-QFS MySQL user
  - `host_name` is the hostname or IP address of the remote SAM-QFS metadata server or localhost when MySQL is installed on the SAM-QFS metadata server host.
  - `user-password` is the password that you assign to the SAM-QFS MySQL user.

In the example, we create the user `samsql` on the SAM-QFS metadata server `samqfs1mds`. We set the user password `samsqluserpassw0rd` for demonstration purposes (it would not be a secure choice for production database use):

```
[samsql]root@solaris:~# mysql --user=root
Enter Password:
mysql> CREATE USER 'samsql'@'samqfs1mds' IDENTIFIED BY 'samsqluserpassw0rd'
mysql>
```

6. Grant the SAM-QFS 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`:

```
[samsql]root@solaris:~# 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>
```

7. Close the MySQL command interface, and return to the operating system command shell. Use the MySQL command `QUIT`.

```
[samsql]root@solaris:~# 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 SAM-QFS metadata server host as root.

In the example, we login to the host `samqfs1mds`:

```
[samqfs1mds]root@solaris:~#
```

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/samqfs1_recovery/140129`, that we scheduled when we configured the `samqfs1` file system (see "Configure File System Protection" on page 6-38). We send the output to the database load file `/root/samqfs1dataload`:

```
[samqfs1mds]root@solaris:~# samfsrestore -SZ /root/samqfs1dataload -f \
/zfs1/samqfs1_recovery/140129
```

3. If you do not have a current recovery point file, create a database load file directly. Change to the SAM-QFS file system's root directory, and use the command `samfsdump -SZ output-path-name`.

See the `samfsdump` man page for additional details. In the example, we change to the `/samqfs1` directory. We send the output to the database load file `/root/samqfs1dataload`:

```
[samqfs1mds]root@solaris:~# cd /samqfs1
[samqfs1mds]root@solaris:~# samfsdump -SZ /root/samqfs1dataload
```

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`:

```
[samqfs1mds]root@solaris:~#
```

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:

```
[samqfs1mds]root@solaris:~# 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`:

```
[samqfs1mds]root@solaris:~# 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 SAM-QFS metadata server host, `samqfs1mds`. So we enter the hostname `localhost`:

```
[samqfs1mds]root@solaris:~# 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 SAM-QFS 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:

```
[samqfs1mds]root@solaris:~# 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 SAM-QFS 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`:

```
[samqfs1mds]root@solaris:~# 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`:

```
[samqfs1mds]root@solaris:~# 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:

```
[samqfs1mds]root@solaris:~# 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):

```
[samqfs1mds]root@solaris:~# 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 SAM-QFS file system. Save the file and close the editor.

In the example, the file system is mounted at `/samqfs/samqfs1`:

```
[samqfs1mds]root@solaris:~# 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:/samqfs/samqfs1
:wq
[samqfs1mds]root@solaris:~#
```

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 SAM-QFS 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`.

```
[samqfs1mds]root@solaris:~# 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/samqfs1dataload`.

```
[samqfs1mds]root@solaris:~# samdb load samqfs1 /root/samqfs1dataload
```

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/samqfs1dataload`.

```
[samqfs1mds]root@solaris:~# samdb check samqfs1
```

14. Next, [Mount the SAM-QFS File System with Database Support Enabled](#).

## Mount the SAM-QFS File System with Database Support Enabled

1. Log in to the SAM-QFS metadata server host as `root`.

```
[samqfs1mds]root@solaris:~#
```

2. Back up the `/etc/vfstab` file.

```
[samqfs1mds]root@solaris:~# cp /etc/vfstab /etc/vfstab.backup
```



---

## Configuring Notifications and Logging

SAM-QFS 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 SAM-QFS Application Logging](#)
- [Configuring Device Logging](#)
- [Configuring Log Rotation](#)
- [Enabling Email Alerts.](#)

### Configuring Simple Network Management Protocol (SNMP)

Network management applications can monitor SAM-QFS 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 SAM-QFS 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 a SAM-QFS 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 SAM-QFS server as `root`.

In the example, the SAM-QFS server host is `samqfs1mds`:

```
[samqfs1mds]root@solaris:~#
```

2. Open the `/etc/hosts` file in a text editor, and 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:

```
[samqfs1mds]root@solaris:~# 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 add the missing management station, `management2`:

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

```
[samqfs1mds]root@solaris:~# 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
:wq
[samqfs1mds]root@solaris:~#
```

4. When the `/etc/hosts` file contains entries for all of your intended SNMP management station hosts, close the editor.

```
[samqfs1mds]root@solaris:~# vi /etc/hosts
...
10.0.0.20 management1
10.0.0.30 management2
:q
[samqfs1mds]root@solaris:~#
```

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 SAM-QFS server as `root`.

In the example, the SAM-QFS server host is `samqfs1mds`:

```
[samqfs1mds]root@solaris:~#
```

2. Open the `/etc/opt/SUNWsamfs/defaults.conf` file 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:

```
[samqfs1mds]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.
...
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.

```
[samqfs1mds]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.
...
alerts = on
:wq
root@solaris:~#
```

4. Tell the SAM-QFS service to reread the `defaults.conf` file and reconfigure itself accordingly. Use the command `samd config`.

```
[samqfs1mds]root@solaris:~# samd config
```

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 SAM-QFS server as `root`.

In the example, the SAM-QFS server host is `samqfs1mds`:

```
[samqfs1mds]root@solaris:~#
```

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:

```
[samqfs1mds]root@solaris:~# 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
[samqfs1mds]root@solaris:~#
```

6. Next, [Enable SAM-QFS 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 SAM-QFS server as `root`.

In the example, the SAM-QFS server host is `samqfs1mds`:

```
[samqfs1mds]root@solaris:~#
```

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:

```
[samqfs1mds]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.
...
#alerts = on
:wq
```

- ```
[samqfs1mds]root@solaris:~#
```
- 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.


```
[samqfs1mds]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.
...
alerts = off
:wq
[samqfs1mds]root@solaris:~#
```
 - Tell the SAM-QFS service to reread the `defaults.conf` file and reconfigure itself accordingly. Use the command `samd config`.


```
[samqfs1mds]root@solaris:~# samd config
```
 - Stop here. SNMP support is disabled.

Enabling SAM-QFS Application Logging

The `/var/adm/sam-log` file records status and error information for the SAM-QFS application and its component daemons and processes. To set up the logging process, proceed as follows:

Enable SAM-QFS Logging

- Log in to the SAM-QFS server as `root`.

In the example, the SAM-QFS server host is `samqfs1mds`:

```
[samqfs1mds]root@solaris:~#
```
- Open the `/etc/syslog.conf` file in a text editor.

In the example, we open the file in the `vi` editor:

```
[samqfs1mds]root@solaris:~# 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
...
```
- 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:

```
[samqfs1mds]root@solaris:~# 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
...
```

```
# SAM-QFS logging
local17.debug    /var/adm/sam-log
:wq
root@solaris:~#
```

4. Create the log file `/var/adm/sam-log`. Use the command `touch /var/adm/sam-log`.

```
[samqfs1mds]root@solaris:~# touch /var/adm/sam-log
```

5. Tell the Solaris `syslogd` daemon to re-read its configuration files and start SAM-QFS logging. Use the command `kill -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, SAM-QFS logging is enabled:

```
[samqfs1mds]root@solaris:~# kill -HUP syslogd
```

6. Go to [Configuring Device Logging](#).

Configuring Device Logging

The device logs 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 selectively enable logging of additional events on a per-device basis by using the `samset` command (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 SAM-QFS server as `root`.

In the example, the SAM-QFS server host is `samqfs1mds`:

```
[samqfs1mds]root@solaris:~#
```

2. Open the `/etc/opt/SUNWsamfs/defaults.conf` in a text editor.

In the example, we open the file in the `vi` editor:

```
[samqfs1mds]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.
...
```

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, limit the range of events that you enable in `defaults.conf` to the default selections. For diagnostic purposes, you can selectively enable additional events as needed using the `samset` command.

In the example, we enable device logging for all devices using the default logging level:

```
[samfs-mdsl]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-mdsl]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-mdsl]root@solaris:~#
```

5. Tell the SAM-QFS service to reread the `defaults.conf` file and reconfigure itself accordingly. Use the command `samd config`.

```
[samfs-mdsl]root@solaris:~# samd config
```

6. Next, [Set Up Automatic Rotation for SAM-QFS 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 SAM-QFS 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 SAM-QFS Log Files

Consider rotating the following logs:

- The SAM-QFS 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, because the log information is valuable for analytics and file-system recovery. For proper handling of archiver logs, see ["Configure File System Protection"](#) on page 6-38.

Then, once you have decided on the logs that should be rotated, proceed as follows for each log:

1. Log in to the SAM-QFS server as `root`.

In the example, the SAM-QFS server host is `samqfs1mds`:

```
[samqfs1mds]root@solaris:~#
```

2. Copy the script file from the uninstalled location, `/opt/SUNWsamfs/examples/log_rotate.sh`, to `/etc/opt/SUNWsamfs/scripts/log_rotate.sh`.

```
[samfs-mds1]root@solaris:~# cp /opt/SUNWsamfs/examples/log_rotate.sh \
/etc/opt/SUNWsamfs/scripts/
```

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
```

- Continuing on the same line, enter the path and name of the shell script file that rotates SAM-QFS logs, `/etc/opt/SUNWsamfs/scripts/log_rotate.sh`, followed by a space.

```
[samfs-mdsl]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
```

- 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 a SAM-QFS 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-mdsl]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
```

- Start a new line. Create a crontab entry that starts 10 minutes after the `log_rotate.sh` script and tells the Solaris `syslogd` daemon to close its old log file and resume logging to a new one. 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 SAM-QFS logging at 3:20 AM every Sunday:

```
[samfs-mdsl]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`
```

- Save the file, and close the editor.

```
[samfs-mdsl]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-mdsl]root@solaris:~#
```

9. Repeat this procedure until you have configured log rotation for all required logs.
10. Next, see [Enabling Email Alerts](#).

Enabling Email Alerts

Email alerts are best set up from the SAM-QFS 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 SAM-QFS system is now configured. But before you begin to use it, protect your work. For instructions, see "[Backing Up the SAM-QFS Configuration](#)" on page 13-1.

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 while insuring 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 to reflect the I/O request, 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, SAM-QFS 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 physical I/O efficiency.

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.

SAM-QFS 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 2-1, select your approach as follows:

- If your application consistently makes small, variably sized, and/or misaligned I/O requests, accept the SAM-QFS 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
```

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 SAM-QFS `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 to a point. So test the performance of the system after resetting the `readahead` size and then adjust it upwards until you see no more improvement in transfer rates.

Enable Switching Between Paged and Direct I/O

You can configure SAM-QFS 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
```

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 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 stripe=1,dio_rd_form_min=512

```

- 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      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

```

- 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

```

- 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
```

- 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
```

- 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
```

- 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
```

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 SAM-QFS `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
```

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
```

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](#)).

Backing Up the SAM-QFS Configuration

Once you have completed SAM-QFS configuration, protect your investment by backing up configuration files and related information. Carry out the following tasks:

- [Create a Backup Location for Your SAM-QFS Configuration](#)
- [Run `samexplorer` and Safely Store the Report](#)
- [Manually Backup the SAM-QFS Configuration](#)

Create a Backup Location for Your SAM-QFS Configuration

Proceed as follows:

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

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

2. Select a storage location for backup copies of your SAM-QFS 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@solaris:~# 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 SAM-QFS software and file systems. Oracle support services personnel use the output when troubleshooting. So creating a baseline `samexplorer` report whenever you configure or reconfigure SAM-QFS 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:

```
[samqfs1mds]root@solaris:~#
```

2. In the directory that holds your backup configuration information, create a subdirectory for sameexplorer 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:

```
[samqfs1mds]root@solaris:~# mkdir /zfs1/sam_config/explorer
```

3. Create the sameexplorer report in the selected directory. Use the command `sameexplorer path/hostname.YYYYMMDD.hhmmz.tar.gz`, where *path* is the path to the chosen directory, *hostname* is the name of the SAM-QFS file system host, and *YYYYMMDD.hhmmz* is a date and time stamp.

By default, the command creates the file:

`/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/:
```

```
[samqfs1mds]root@solaris:~# sameexplorer \
/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
```

4. Repeat this procedure whenever you significantly reconfigure your file systems.
5. Next, [Manually Backup the SAM-QFS Configuration](#).

Manually Backup the SAM-QFS Configuration

While the sameexplorer utility captures much of your SAM-QFS 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:

```
[samqfs1mds]root@solaris:~#
```

2. In the directory that holds your backup configuration information, create a subdirectory for manual backup copies of your SAM-QFS 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 /samqfs1. So we have created the directory /zfs1/sam_config/samconfig:

```
[samqfs1mds]root@solaris:~# mkdir /zfs1/sam_config/samconfig
```

3. In the chosen directory, create a subdirectory for the current SAM-QFS 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:

```
samqfs1mdsroot@solaris:~# mkdir /zfs1/sam_config/samconfig/20140127
```

4. Copy the configuration files to another file system.

```
/etc/opt/SUNWsamfs/
  mcf
  archiver.cmd
  defaults.conf
  diskvols.conf
  hosts.family-set-name
  hosts.family-set-name.local
  preview.cmd
  recycler.cmd
  releaser.cmd
  rft.cmd
  samfs.cmd
  stager.cmd
  inquiry.conf
  samremote                # SAM-Remote server configuration file
  family-set-name          # SAM-Remote client configuration file
  network-attached-library # Parameters file
  scripts/*                # Back up all locally modified files
/var/opt/SUNWsamfs/
```

5. Back up all library catalog data, including that maintained by the historian. For each catalog, use the command `/opt/SUNWsamfs/sbin/dump_cat -V catalog-file`, where *catalog-file* is the path and name of the catalog file. Redirect the output to *dump-file*, in a new location.

In the example, we dump the catalog data for library1 to the file library1cat.dump in a directory on the independent NFS-mounted file system zfs1:

```
samqfs1mdsroot@solaris:~# dump_cat -V /var/opt/SUNWsamfs/catalog/library1cat \
> /zfs1/sam_config/20140513/catalogs/library1cat.dump
```

6. Copy system configuration files that were modified during SAM-QFS 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 SAM-QFS 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 SAM-QFS, 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 SAM-QFS, 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.

Glossary of Equipment Types

The value of the `Equipment Type` field of the Master Configuration File (`mcf`) identifies devices and device configurations within the SAM-QFS software using 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 three 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 and the SAM-QFS 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, SAM-QFS can automatically set the correct type based on SCSI vendor codes.

gXXX

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 SAM-QFS 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 `gXXX` 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 SAM-QFS file system that maintains file-system metadata on the same devices that store file data.

od

Any SCSI-attached optical disk. SAM-QFS sets the appropriate equipment type automatically using the SCSI vendor code.

rb

Any SCSI-attached tape library. SAM-QFS 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.

tp

Any SCSI-attached tape drive. SAM-QFS sets the appropriate equipment type automatically using the SCSI vendor code.

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 SAM-QFS 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 not 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.

Mount Options in a Shared File System

The SAM-QFS 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 *StorageTek Storage Archive Manager and StorageTek QFS samu Command Reference*.

For more information about any of these mount options, see the `mount_samfs` 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 a SAM-QFS Shared File System

The `shared` mount option declares a file system to be a SAM-QFS shared file system. This option must be specified in the `/etc/vfstab` file in order for the file system to be mounted as a SAM-QFS 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.

minallopsz and maxallopsz: Tuning Allocation Sizes

The `minallopsz` and `maxallopsz` 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 minallopsz=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 a SAM-QFS Shared File System

By default, when hosts share files, the SAM-QFS 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 SAM-QFS 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 SAM-QFS 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 SAM-QFS 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, SAM-QFS 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 SAM-QFS 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 because, even 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 SAM-QFS 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 the SAM-QFS shared file system's 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 SAM-QFS 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 the SAM-QFS 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 *StorageTek Storage Archive Manager and StorageTek QFS Installation and Configuration Guide* (SAM-QFS Customer Documentation Library, docs.oracle.com).

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 SAM-QFS 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 SAM-QFS 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=MyNdOhPm`.

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.

See the *StorageTek Storage Archive Manager and StorageTek QFS Installation and Configuration Guide* (SAM-QFS Customer Documentation Library, docs.oracle.com.) for more information.

Configuration Directives for Archiving

This appendix lists directives used in the SAM-QFS command (.cmd) files that configure archiving file systems and related software operations (for comprehensive descriptions, see the SAM-QFS man pages). Directives are divided into the following sections:

- [Archiving Directives](#)
- [Staging Directives](#)
- [Preview Request Directives](#)

You can configure SAM-QFS command files either from the command line, as described here, or by using the SAM-QFS Manager software. For information on SAM-QFS Manager, see the online help.

Archiving Directives

This section provides usage information for the *archiving directives* that make up the `archiver.cmd` file. Archiving directives define the archive sets that control how files are archived, the media used, and the overall behavior of the archiving software.

There are four basic types of archiving directives:

- [Global Archiving Directives](#)
- [File System Directives](#)
- [Copy Parameters](#)
- [Volume Serial Number \(VSN\) Association Directives](#)

Both global and file system directives control how files are archived. But the archiver evaluates the file system-specific directives before evaluating the global directives. So file system directives override global directives when there is a conflict. Similarly, within the file system directives, the first directive listed overrides any subsequent, conflicting directives.

Global Archiving Directives

Global directives control the overall archiver operation and enable you to optimize operations for all configured file systems. Global directives are specified by entering equal sign (=) in the second field or the absence of additional fields. Global directives start the `archiver.cmd` file and end at the first of the [File System Directives](#).

archivemeta: Controlling Whether Metadata Is Archived

The `archivemeta` directive controls whether file system metadata is archived. 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. By default, metadata is not archived.

This directive has the following format:

```
archivemeta=state
```

For *state*, specify either `on` or `off`. The default is `off`.

The archiving process for metadata depends on whether you are using a Version 1 or a Version 2 superblock, as follows:

- 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.

archmax: Controlling the Size of Archive Files

The `archmax` directive specifies the maximum size of an archive (`.tar`) file. After the *target-size* value is met, no more user files are added to the archive file. Large user files are written in a single archive file.

To change the defaults, use the following directive:

```
archmax=media target-size
```

where *media* is one of the media types defined in [Appendix A](#) and in the `mcf` man page and where *target-size* is the maximum size of the archive file. This value is media-dependent. By default, archive files written to optical discs are no larger than 5 megabytes. The default maximum archive file size for tapes is 512 megabytes.

Setting large or small sizes for archive files has advantages and disadvantages. For example, if you are archiving to tape and `archmax` is set to a large size, the tape drive stops and starts less often. However, when writing large archive files, a premature end-of-tape causes a large amount of tape to be wasted. As a best practice, do not set the `archmax` directive to be more than 5 percent of the media capacity.

The `archmax` directive can also be set for an individual archive set.

bufsize: Setting the Archiver Buffer Size

By default, a file being archived is copied to archive media using a memory buffer. You can use the `bufsize` directive to specify a non-default buffer size and, optionally, to lock the buffer. These actions can improve performance. You can experiment with different *buffer-size* values. This directive has the following format:

```
bufsize=media buffer-size [lock]
```

where:

- *media* is one of the media types defined in [Appendix A](#) and in the `mcf` man page
- *buffer-size* is a number in the range [2-1024]. The default is 4. This value is multiplied by the `dev_blksize` value for the media type, and the resulting buffer

size is used. The `dev_blksize` value is specified in the `defaults.conf` file. See the `defaults.conf` man page for details.

- `lock` indicates whether the archiver can use locked buffers when making archive copies.

If `lock` is specified, the archiver sets file locks on the archive buffer in memory for the duration of the `sam-arcopy` 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.

The `lock` argument must be specified only on large systems with large amounts of memory. Insufficient memory can cause an out-of-memory condition. The `lock` argument is effective only if direct I/O is enabled for the file being archived. By default, `lock` is not specified and the file system sets the locks on all direct I/O buffers, including those for archiving.

You can specify a buffer size and a lock for each archive set by using the archive set copy Parameters, `-bufsize` and `-lock`. For more information, see "[Archive Copy Directives](#)" on page C-8.

drives: Controlling the Number of Drives Used for Archiving

By default, the archiver uses all of the drives in an automated library for archiving. To limit the number of drives used, use the `drives` directive. This directive has the following format:

```
drives=auto-lib count
```

where `auto-lib` is the family set name of the automated library as defined in the `mcf` file and `count` is the number of drives to be used for archiving activities.

You can also use the archive set copy parameters `-drivemax`, `-drivemin`, and `-drives` for this purpose.

examine: Controlling Archive Scans

The `examine` directive sets the `method` that the archiver uses to identify files that are ready for archiving:

```
examine=method
```

where `method` is one of the following directives:

- `noscan`, the default, specifies *continuous archiving*. After the initial scan, directories are scanned only when the content changes and archiving is required. Directory and inode information is not scanned. This archiving method provides better performance than scan archiving, particularly for file systems with more than 1,000,000 files.
- `scan` specifies *scan archiving*. The initial file system scan is a directory scan. Subsequent scans are inode scans.
- `scandirs` specifies scan archiving on directories only. If the archiver finds a directory with the `no_archive` attribute set, the directory is not scanned. If you have files that do not change, place them in this type of directory to reduce the amount of time spent on archiving scans.
- `scaninodes` specifies scan archiving on inodes only.

interval: Specifying an Archive Interval

The archiver runs periodically to examine the status of all mounted archived-enabled file systems. The timing is controlled by the archive interval, which is the time between scan operations on each file system. To change the archive interval, use the `interval` directive.

The `interval` directive initiates full scans only when continuous archiving is not set and no `startage`, `startsize`, or `startcount` Parameters have been specified. If continuous archiving is set (`examine=noscan`), the `interval` directive acts as the default `startage` value. This directive has the following format:

```
interval=time
```

For *time*, specify the amount of time you want between scan operations on a file system. By default, *time* is interpreted in seconds and has a value of 600, which is 10 minutes. You can specify a different unit of time, such as minutes or hours.

If the archiver receives the `samu` utility's `arrun` command, it begins scanning all file systems immediately. If the `examine=scan` directive is also specified in the `archiver.cmd` file, a scan is performed after `arrun` or `arscan` is issued.

If the `hwm_archive` mount option is set for the file system, the archive interval can be shortened automatically. This mount option specifies that the archiver commences its scan when the file system is filling up and the high-water mark is crossed. The `high=percent` mount option sets the high-water mark for the file system.

For more information about specifying the archive interval, see the `archiver.cmd` and `mount_samfs` man pages.

logfile: Specifying an Archiver Log File

The archiver can produce a log file that contains information about each file that is archived, re-archived, or unarchived. The log file is a continuous record of archival action. By default, archiver log files are not enabled. To specify a log file, use the `logfile` directive. This directive has the following format:

```
logfile=pathname
```

For *pathname*, specify the absolute path and name of the log file. The `logfile` directive can also be set for an individual file system.

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 and back them up. See the *StorageTek Storage Archive Manager and StorageTek QFS Installation and Configuration Guide* (SAM-QFS Customer Documentation Library, docs.oracle.com.) for more information.

notify: Renaming the Event Notification Script

The `notify` directive sets the name of the archiver's event notification script file. This directive has the following format:

```
notify=filename
```

For *filename*, specify the name of the file containing the archiver event notification script or the full path to this file. The default file name is `/etc/opt/SUNWsamfs/scripts/archiver.sh`.

The archiver executes this script to process various events in a site-specific manner. The script is called with one of the following keywords for the first argument: `emerg`, `alert`, `crit`, `err`, `warning`, `notice`, `info`, and `debug`.

Additional arguments are described in the default script. For more information, see the `archiver.sh` man page.

ovflmin: Controlling Volume Overflow

When volume overflow is enabled, the archiver can create archived files that span multiple volumes. When a file size exceeds the specified minimum size, the archiver writes the remaining portion of this file to another volume of the same type. The portion of the file written to each volume is called a *section*. The `s1s` command lists the archive copy, showing each section of the file on each volume.

The archiver controls volume overflow through the `ovflmin` directive. By default, volume overflow is disabled. To enable volume overflow, use the `ovflmin` directive in the `archiver.cmd` file. This directive has the following format:

```
ovflmin = media minimum-file-size
```

where *media* is one of the media types defined in [Appendix A](#) and in the `mcf` man page and *minimum-file-size* is the size of the smallest file that you want to trigger the volume overflow. The `ovflmin` directive can also be set for an individual archive set.

Use volume overflow with caution after assessing its effects. Disaster recovery and recycling are much more difficult with files that span volumes.

Volume overflow files do not generate checksums. For more information on using checksums, see the `ssum` man page.

scanlist_squash: Controlling Scanlist Consolidation

The `scanlist_squash` parameter controls scanlist consolidation. The default setting is `off`. This parameter can be either global or file system-specific.

When this option is enabled, the scan list entries for files in two or more subdirectories with the same parent directory that need to be scanned by `sam-arfind` at a much later time are consolidated. These directories are combined upwards to the common parent, which results in a deep recursive scan of many subdirectories. This consolidation can cause a severe performance penalty if archiving on a file system that has a large number of changes to many subdirectories.

setarchdone: Controlling the Setting of the archdone Flag

The `setarchdone` Parameter is a global directive that controls the setting of the `archdone` flag when the file is examined by `sam-arfind`. This directive has the following format:

```
setarchdone=state
```

where *state* is either `on` or `off`.

When all archive copies for a file have been made, the `archdone` flag is set for that file to indicate that no further archive action is required. During directory scans, the `archdone` flag is also set for files that will never be archived. Because evaluating whether a file will ever be archived can affect performance, the `setarchdone` directive gives you control over this activity. This directive controls the setting of the `archdone` flag only on files that will never be archived. It does not affect the setting of the `archdone` flag after archive copies are made.

The default setting for the directive is `off` if the `examine` directive is set to `scandirs` or `noscan`.

wait: Delaying Archiver Startup

The `wait` directive causes the archiver to wait for a start signal from `samu` or SAM-QFS File System Manager. By default, the archiver begins archiving when started by `sam-fsd`. This directive has the following format:

```
wait
```

The `wait` directive can also be set for an individual file system.

File System Directives

File system directives define archiving behavior for specific file systems:

- [fs: Specifying a File System](#)
- [copy-number \[archive-age\]: Specifying Multiple Copies of File System Metadata](#)
- [interval, logfile, scanlist as File System Directives](#)

fs: Specifying a File System

Each `fs=file-system-name` directive introduces a sequence of archiving directives that apply only to the named file system, `file-system-name`. This directive has the following format:

```
fs=file-system-name
```

where `file-system-name` is the file system name defined in the `mcf` file.

General directives and archive set association directives that occur after an `fs=` directive apply only to the specified file system.

copy-number [archive-age]: Specifying Multiple Copies of File System Metadata

File system metadata includes path names in the file system. If you have frequent changes to directories, the new path names cause the creation of new archive copies and result in frequent loading of the volumes specified for metadata. If more than one copy of the metadata is required, place copy definitions in the `archiver.cmd` file immediately after the `fs=` directive.

```
copy-number [archive-age]
```

where time is expressed as one or more combinations of an integer and a unit of time. Units include `s` (seconds), `m` (minutes), `h` (hours), `d` (days), `w` (weeks), and `y` (years). By default, SAM-QFS makes only a single copy of the metadata.

In the example, copy 1 of the metadata for the `fs=samma1` file system is made after 4 hours (4h) and copy 2 is made after twelve hours (12h):

```
# General Directives
archivemeta = off
examine = noscan
# Archive Set Assignments
fs = samma1
1 4h
2 12h
```

interval, logfile, scanlist as File System Directives

Several directives can be specified both as global directives for all file systems and as directives specific to only one file system. These directives are described in the following sections:

- [interval: Specifying an Archive Interval](#)
- [logfile: Specifying an Archiver Log File](#)
- [scanlist_squash: Controlling Scanlist Consolidation](#)
- [wait: Delaying Archiver Startup](#)

archive-set-name: the Archive Set Assignment Directive

The archive set assignment directive specifies which files will be archived together. You can specify files very narrowly, using the wide range of selection criteria described below. But avoid doing so unless absolutely necessary. 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.

Each archive set assignment directive has the following format:

```
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 the administrator-defined name for the archive set.
Names can contain up to 29 characters in any combination of upper and/or lower case letters [A-Za-z], numerals [0-9], and underscore characters (_), as long as the first character is a letter. You cannot include any other characters, including spaces, and you cannot use the names of the SAM-QFS special archive sets `no_archive` and `allsets` for your own archive sets.
- *path* specifies the path relative to the mount point of the subdirectory where archiving starts within the file system. All files in the starting directory and its subdirectories are archived. To include all of the files in a file system, use the dot (.) character. A leading slash (/) is not allowed in the path.
- `-access` rearchives 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*.

Archive Copy Directives

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. To change the default behavior, use archive copy directives. Archive copy directives must appear immediately after the archive set assignment directive to which they pertain.

The archive copy directives begin with a *copy-number* value of 1, 2, 3, or 4. The digit is followed by one or more arguments that specify archive characteristics for that copy. Each archive copy directive has the following format:

```
copy-number [archive-age] [-release [attribute] [-norelease] [-stage[attribute] [unarchive-age]
```

where:

- 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).
- The optional `-release` parameter clears the SAM-QFS 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`. 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.
- The optional `-norelease` parameter does not clear the SAM-QFS releaser software to free the disk space used by files until all copies marked with `-norelease` have been made.
- `-release -norelease`, used together, require that the SAM-QFS software free the disk space used by files immediately after all copies that are flagged `-release -norelease` are made. SAM-QFS does not wait for the releaser process to run.
- The optional `-stage` parameter The optional release *attribute* is `-a`, `-c copy-number`, `-f`, `-I`, `-i input_file`, `-w`, `-n`, `-p`, `-V`, `-x`, `-r`, `-d`, where:
 - `-a` (**associative staging**) 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* 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).

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 = samma1
logfile = /var/adm/samma1.archive.log
allsamma1 .
    1 -norelease 5m
    2 -norelease 1h 7y6m
```

Copy Parameters

Copy parameters define how the copies specified by an archive set are created. The archive set copy parameters section of the `archiver.cmd` file begins with the `params` directive and ends with the `endparams` directive:

```
params
allsets -sort path -offline_copy stageahead
allsets.1 -startage 10m -startsize 10M -drives 10 -archmax 1G
allsets.2 -startage 1h -startsize 1G -drives 2 -archmax 10G -reserve set
endparams
```

Each copy parameter takes the following form:

```
archive-set-name[.copy-number][R] [-startage time] [-startcount count] [-startsize size] [-archmax maximum-size] [-bufsize=buffer-size] [-drivemax maximum-size] [-drivemin minimum-size] [-drives number] [-fillvsns] [-lock] [-offline_copy method] [-sort criterion] [-rsort criterion] [-recycle_dataquantity size] [-recycle_hwm percent] [-recycle_ignore] [-recycle_mailaddr mail-address] [-recycle_mingainpercentage] [-recycle_vsncount count] [-recycle_minobs percentage] [-unarchagetime_ref] [-tapenonstop] [-reserve keyword] [-priority multiplier ranking]
```

where:

- *archive-set-name* is the name of an archive set defined by an archive set assignment directive in the [File System Directives](#) or the special directive `allsets`, which applies the specified copy parameters to all defined archive sets. Set parameters for `allsets` first, before specifying parameters for individual archive sets. Otherwise, the parameters for individual archive sets will override the `allsets` specification, defeating its purpose.
- *.copy-number* limits the application of the specified copy parameters to the archive copy specified by *copy-number*, where *copy-number* is an integer in the range [1-4], and the optional *R* limits application of the parameters to rearchived 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, *count* 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` limits the size of an archive file to no more than *maximum-size*, where *maximum-size* is media-dependent. The default maximum archive file size for magnetic tape is 512 megabytes. Archive files written to optical discs are no larger than 5 megabytes.

See "[archmax: Controlling the Size of Archive Files](#)" on page C-2 for a description of the global archiving directive of the same name.

- `-bufsize=` *buffer-size* sets the size of the buffer that holds the archive file as it is being written out to archival media to *buffer-size***dev_blksize*, where *buffer-size* is an integer in the range [2-32] and *dev_blksize* is the block size specified for the media type in the `defaults.conf` file. The default is 4.
- `-drivemax` limits the amount of data archived using one drive to no more than *maximum-size* megabytes, where *maximum-size* is an integer. By default, *maximum-size* is not specified.

When multiple drives are specified using the `-drives` parameter, limiting the amount of data written to any one drive can improve drives can help to balance workloads and improve overall drive utilization.

- `-drivemin` *minimum-size* limits the amount of data archived using one drive to at least *minimum-size* megabytes, where *minimum-size* is an integer. The default is the value of `-archmax` (if specified) or the value listed for the media type in the `defaults.conf` file.

Setting a lower limit on the amount of data written to a drive can improve drive utilization and efficiency. Set *minimum-size* large enough that the transfer time significantly exceeds the time required to load, position, and unload media. When `-drivemin` is specified, multiple drives are only used when data transfers are large enough.

- `-drives` *number* limits the number of drives used for archiving to at most *number*, where *number* is an integer. The default is 1.

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.

- `-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 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. The specified *method* can be `direct`, `stageahead`, `stageall`, or `none`.

Files can be released as soon as a single archive copy is made, so the remaining copies must be made from an offline copy. A specified `-offline_copy` method lets you tailor the copy process to suit the number of drives that can be made available and the amount of space available in the disk cache.

`direct` copies files directly from the offline volume to the archive volume, using two drives. To insure adequate buffer space, increase the value set by the `stage_n_window` mount option when using this method.

`stageahead` stages the next archive file while writing an archive file to its destination.

`stageall` stages all files to disk cache before archiving, using one drive. Make sure that the disk cache is large to hold the files when using this method.

`none` (the default) stages files to the disk cache as needed before copying to the archive volume.

- `-sort` sorts files by *criterion* before archiving them, where *criterion* is `age`, `priority`, `size`, or `none`.

`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.

- `-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 need 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 (1 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.
- `-recycle_ignore` prevents actual recycling of any media in the archive set, while allowing recycling processes to run normally. Used for testing.
- `-recycle_mailaddr mail-address` directs informational recycler messages to *mail-address*. Mail is not sent by default.
- `-recycle_mingain` limits selection of volumes for recycling to those which would increase their free space by at least the specified *percentage*. The default is 50.

- `-recycle_vsncount` limits the number of volumes that recycler schedules for rearchiving to *count*. Note that the actual number of volumes selected for recycling may also be dependant on the `-recycle_dataquantity` parameter. The parameter is ignored for disk media. The default is 1.
- `-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 parameter is ignored for removable media (see `-recycle_hwm` above). The default is 50.
- `-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, SAM-QFS 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: `allsets -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`.

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.

- `-priority multiplier ranking` changes the archiving priority of files when used with the `sort priority` parameter listed above. The *ranking* is a real number in the range $[(-3.400000000E+38) - 3.400000000E+38]$ ($-3.402823466 \times 10^{38}$ to $3.402823466 \times 10^{38}$) and *multiplier* is the archive characteristic for which you are changing the relative *ranking*, selected from the following list: *age*, *archive_immediate*, *archive_overflow*, *archive_loaded*, *copies*, *copy1*, *copy2*, *copy3*, *copy4*, *offline*, *queuwait*, *rearchive*, *rerelease*, *size*, *stage_loaded*, and *stage_overflow*.

See the archiver and `archiver.cmd` man pages for more information on priorities.

Volume Serial Number (VSN) Pools Directives

The VSN pools section of the `archiver.cmd` file defines named collections of archival media volumes that can be specified as a unit in [Volume Serial Number \(VSN\) Association Directives](#).

The section starts with a `vsnpools` directive and ends either with an `endvsnpools` directive or with the end of the `archiver.cmd` file. The syntax of a VSN pool definition is as follows:

```
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, SAM-QFS 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.

The example defines four VSN pools: `users_pool`, `data_pool`, `proj_pool`, and `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
users_pool li ^VOL2[0-9][0-9]
data_pool li ^VOL3.*
scratch_pool li ^VOL4[0-9][0-9]
proj_pool li ^VOL[56].*
endvsnpools
```

Volume Serial Number (VSN) Association Directives

The VSN associations section of the `archiver.cmd` file assigns archival media volumes to archive sets. This section starts with a `vsns` directive and ends with an `endvsns` directive.

Volumes assignment directives take the following form:

```
archive-set-name.copy-number [media-type volume-specification] [-pool vsn-pool-name]
```

where:

- *archive-set-name* is the name that an archive set assignment directive assigned 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].
- *media-type* is one of the two-character, SAM-QFS 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.
- `-pool vsn-pool-name` is a previously specified, named collection of archival media volumes that can be specified as a unit. See [Volume Serial Number \(VSN\) Pools Directives](#).

The example illustrates various ways in which media can be associated with two lines of VSN specifications.

```
vsns
archiveset.1 lt VSN001 VSN002 VSN003 VSN004 VSN005
archiveset.2 lt VSN0[6-9] VSN10
archiveset.3 -pool data_pool
endvsns
```

Staging Directives

Staging is the process of copying file data from nearline or offline storage back to online storage.

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.

When an application requires an offline file, its archive copy is staged to disk cache unless the file was archived with the `-n` (*never stage*) option. To make the file available to an application immediately, the read operation tracks along directly behind the staging operation so that the access can begin before the entire file is staged.

Stage errors include media errors, unavailability of media, unavailability of an automated library, and others. If a stage error is returned, the SAM-QFS software attempts to find the next available copy of the file, if one exists and if there is a device available to read the archive copy's media.

The `stager.cmd` File

In the `stager.cmd` file, specify directives to override the default behaviors. You can configure the stager to stage files immediately, to never stage files, to staging partially, 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.

This section describes the stager directives. For additional information about stager directives, see the `stager.cmd` man page. If you are using the SAM-QFS File System 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
```

drives: Specifying the Number of Drives for Staging

By default, the stager uses all available drives when staging files. If the stager keeps all the drives busy, it can interfere with the archiver's activities. The `drives` directive specifies the number of drives available to the stager. This directive has the following format:

```
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. By default, this is the number of drives configured in the `mcf` file for this library.

The example specifies that only one drive from the `dog` family set's library is used for staging files:

```
drives = dog 1
```

bufsize: Setting the Stage Buffer Size

By default, a file being staged is read into memory in a buffer before being restored from the archive media to disk cache. Use the `bufsize` directive to specify a buffer size and, optionally, to lock the buffer. These actions can improve performance. You can experiment with various *buffer-size* values. The directive has the following format:

```
bufsize= media-type buffer-size [lock]
```

where:

- *media-type* is one of the two-character, SAM-QFS media type identifiers listed in [Appendix A](#) and in the `mcf` man page.
- *buffer-size* is an integer in the range [2-8192]. This value is multiplied by the *media-type_blksize* value specified in the `defaults.conf` file. The higher the number specified for *buffer-size*, the more memory is used. The default is 16.
- *lock* mandates the use of locked buffers for the duration of each staging operation. This avoids overhead associated with locking and unlocking the staging buffer for each I/O request and improves 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.

The *lock* argument is effective only if direct I/O is enabled for the staged file. For more information about enabling direct I/O, see the `setfa`, `sam_setfa`, and `mount_samfs` man pages.

logfile: Specifying a Staging Log File

You can request that the SAM-QFS software collect file-staging event information and write it to a log file. By default, no log file is written. The `logfile` directive specifies a log file to which the stager can write logging information. The stager writes one or more lines to the log file for each file staged. This line includes information such as the name of the file, the date and time of the stage, and the volume serial number (VSN). The directive has the following format:

```
logfile=filename [event-list]
```

where *filename* is the full path name for the log file and *event-list* is a space-delimited list of the event types that are to be logged:

- all logs all staging events.
- start logs when staging begins for a file.
- finish (default) logs when staging ends for a file.
- cancel (default) logs when the operator cancels a stage.
- error (default) logs staging errors.

The following directive creates a stage log in the `/var/adm/` directory:

```
logfile=/var/adm/stage.log
```

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, SAM-QFS 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 (-).

The example shows part of a typical stager log:

```
S 2014/02/16 14:06:27 dk disk01 e.76d 2557.1759 1743132 /sam1/testdir0/filebu 1 root other root 0 -
F 2014/02/16 14:06:27 dk disk01 e.76d 2557.1759 1743132 /sam1/testdir0/filebu 1 root other root 0 -
S 2014/02/16 14:06:27 dk disk02 4.a68 1218.1387 519464 /sam1/testdir1/fileaq 1 root other root 0 -
S 2014/02/16 14:06:43 dk disk01 13.ba5 3179.41 750880 /sam1/testdir0/filecl 1 root other root 0 -
F 2014/02/16 14:06:43 dk disk01 13.ba5 3179.41 750880 /sam1/testdir0/filecl 1 root other root 0 -
```

maxactive: Specifying the Number of Stage Requests

The `maxactive` directive enables you to specify the number of stage requests that can be active at any one time. The directive has the following format:

```
maxactive=number
```

where *number* is an integer in the range [1-500000]. The default is 4000.

The example specifies that no more than 500 stage requests can be in the queue simultaneously:

```
maxactive=500
```

copysel1: Specifying the Copy Selection Order During Staging

The staging directive `copysel` sets the stager copy selection sequence per file system.

```
copysel=selection-order
```

where *selection-order* is a colon-delimited list of copy numbers in first-to-last order. The default selection order is 1:2:3:4.

For more information, see the `stager.cmd` man page. 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 = hp30 1
fs = samfs1
copysel = 4:3:2:1
fs = samfs2
copysel = 3:1:4:2
```

Preview Request Directives

When a SAM-QFS 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 SAM-QFS library-control daemon (`sam-amld`) reads these directives when it starts uses them until it stops. You cannot change queue priorities dynamically.

There are two 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

The following sections describe how to edit the `preview.cmd` file to control the preview queue:

- [Global Directives](#)
- [Global and/or File System-Specific Directives](#)
- [Sample `preview.cmd` File](#)

Global Directives

The following are purely global directives:

- [vsnpriority: Adjusting Volume Priorities](#)

- [age_priority: Adjusting Priorities for Time Spent Waiting in the Queue](#)

vsn_priority: Adjusting Volume Priorities

The `vsn_priority` directive increases the priority of volumes (VSNs) that are flagged as high-priority volumes by a specified value. The directive takes the following form:

```
vsn_priority=value
```

where *value* is a real number. The default is 1000.0.

You set the high priority flag on volumes using the command

```
chmed +p media-type.volume-serial-number
```

where *media-type* is one of the two-character, SAM-QFS media types listed in [Appendix A](#) and on the `mcf` man page and where *volume-serial-number* is the alphanumeric string that uniquely identifies the high-priority volume in the library. See the `chmed` man page for full information.

age_priority: Adjusting Priorities for Time Spent Waiting in the Queue

The `age_priority` directive changes the relative priority given to the amount of time that a request spends in the queue so that, for example, you can keep older requests from being indefinitely superseded by newer, higher-priority, requests. The directive specifies a multiplier that changes the relative weighting of the time spent in the queue. It takes the following form:

```
age_priority=weighting-factor
```

where *weighting-factor* is a real number greater, less than, or equal to 1.0 and where:

- 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.

Global and/or File System-Specific Directives

The following directives can be applied either globally or on a per-file system basis:

- [hwm_priority: Adjusting Priorities When the Disk Cache is Nearly Full](#)
- [lwm_priority: Adjusting Priorities When the Disk Cache is Nearly Empty](#)
- [lhwm_priority: Adjusting Priorities as the Disk Cache Fills](#)
- [hlwm_priority: Adjusting Priorities as the Disk Cache Empties](#)

hwm_priority: Adjusting Priorities When the Disk Cache is Nearly Full

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 directive takes the following form:

```
hwm_priority=weighting-factor
```

where *weighting-factor* is a real number. The default is 0.0.

lwm_priority: Adjusting Priorities When the Disk Cache is Nearly Empty

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 directive takes the following form:

```
lwm_priority=weighting-factor
```

where *weighting-factor* is a real number. The default is 0.0.

lhwm_priority: Adjusting Priorities as the Disk Cache Fills

The `lhwm_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:

```
lhwm_priority=weighting-factor
```

where *weighting-factor* is a real number. The default is 0.0.

hlwm_priority: Adjusting Priorities as the Disk Cache Empties

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 directive takes the following form:

```
hlwm_priority=weighting-factor
```

where *weighting-factor* is a real number. The default is 0.0.

Sample 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`, `lhwm_priority`, or `hlwm_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:

```
# 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 $1000 + (-200) + (1 \times 100) = 900$
- A staging mount request for a priority volume has the aggregate priority $1000 + 0 + (1 \times 100) = 1100$
- A staging mount request for a non-priority volume has the aggregate priority $0 + 0 + (1 \times 100) = 100$

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 $1000 + 500 + (1 \times 100) = 1600$
- A staging mount request for a priority volume has the aggregate priority $1000 + 0 + (1 \times 100) = 1100$
- A staging mount request for a non-priority volume has the aggregate priority $0 + 0 + (1 \times 100) = 100$

D

Examples

The `/opt/SUNWsamfs/examples/` subdirectory contains sample SAM-QFS 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
archiver.sh
defaults.conf
dev_down.sh
dtrace/fs_mon
dtrace/ino_mon
hosts.shsam1
hosts.shsam1.local.client
hosts.shsam1.local.server
inquiry.conf
load_notify.sh
log_rotate.sh
```

metadata_config_samfs.xml
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

Product Accessibility Features

Users with low vision, blindness, color blindness, or other visual impairments can access the StorageTek Storage Archive Manager and StorageTek QFS Software (SAM-QFS) 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 SAM-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/>.

addressable storage

The storage space encompassing online, nearline, offsite, and offline storage that is user-referenced through a SAM-QFS file system.

admin set ID

A storage administrator-defined set of users and/or groups that share common characteristics. Admin sets are typically created to administer storage for projects that involve users from several groups and span multiple files and directories.

archival media

The media to which an archive file is written. Archival media includes both removable tape or magneto-optical cartridges and disk file systems configured for archiving.

archival storage

Data storage space created on archival media.

archive set

An archive set identifies a group of files to be archived, and the files share common criteria that pertain to the size, ownership, group, or directory location. Archive sets can be defined across any group of file systems.

archiver

The archive program that automatically controls the copying of files to removable cartridges.

associative staging

Staging a group of related files when any one member of the group is staged. When files inhabit the same directory and are frequently used together, file owners can associate them by setting the SAM-QFS associative-staging file attribute. Then if any files in the group are offline when one of them is accessed by an application, SAM-QFS stages the entire group from archival media to disk cache. This insures that all needed files re available at the same time.

audit (full)

The process of loading cartridges to verify their VSNs. For magneto-optical cartridges, the capacity and space information is determined and entered into the automated

library's catalog. See [volume serial number \(VSN\)](#).

automated library

A robotically controlled device designed to automatically load and unload removable media cartridges without operator intervention. An automated library contains one or more drives and a transport mechanism that moves cartridges to and from the storage slots and the drives.

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 hard disk or magnetic tape cartridge. On disk devices, this is equivalent to the *sector size*, which is typically 512 bytes.

cartridge

A container for data-storage media, such as magnetic tape or optical media. Also called a [volume](#), *tape*, or *piece of media*. See [volume serial number \(VSN\)](#).

catalog

A record of the removable media volumes in an automated library. There is one catalog for each automated library and, at a site, there is one historian for all automated libraries. Volumes are identified and tracked using a [volume serial number \(VSN\)](#).

client-server

The model of interaction in a distributed system in which a program at one site sends a request to a program at another site and awaits a response. The requesting program is called the client. The program satisfying the response is called the server.

connection

The path between two protocol modules that provides reliable stream delivery service. A TCP connection extends from a TCP module on one machine to a TCP module on the other.

data device

In a file system, a device or group of devices upon which file data is stored.

DAU

See [disk allocation unit \(DAU\)](#).

device logging

A configurable feature that provides specific error information for the hardware devices that support a SAM-QFS file system.

device scanner

Software that periodically monitors the presence of all manually mounted removable devices and that detects the presence of mounted cartridges that can be requested by a user or other process.

direct access

A file attribute (stage never) designating that a nearline file can be accessed directly from the archive media and need not be retrieved to disk cache.

direct attached library

An automated library connected directly to a server using a SCSI interface. A SCSI-attached library is controlled directly by the SAM-QFS software.

direct I/O

An attribute used for large block-aligned sequential I/O. The `setfa` command's `-D` option is the direct I/O option. It sets the direct I/O attribute for a file or directory. If applied to a directory, the direct I/O attribute is inherited.

directory

A file data structure that points to other files and directories within the file system.

disk allocation unit (DAU)

In SAM-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 minimum number of I/O operations needed when transferring a file of a given size. It should be a multiple of the **block size** of the disk device.

Disk allocation unit vary depending upon the SAM-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.

disk buffer

In a SAM-Remote configuration, the buffer on the server system that is used for archiving data from the client to the server.

disk cache

The disk-resident portion of the file system software, used to create and manage data files between online disk cache and archive media. 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.

disk striping

The process of recording a file across several disks, thereby improving access performance and increasing overall storage capacity. See also **striping**.

drive

A mechanism for transferring data to and from a removable media volume.

Ethernet

A packet-switched local-area network technology.

extent array

The array within a file's inode that defines the disk location of each data block assigned to the file.

family device set

See [family set](#).

family set

A storage device that is represented by a group of independent physical devices, such as a collection of disks or the drives within an automated library. See also [storage family set](#).

FDDI

Fiber-distributed data interface, a standard for data transmission in a local area network that can extend in range up to 200 km (124 miles). The FDDI protocol is based on the token ring protocol.

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 hierarchical collection of files and directories.

file-system-specific directives

Archiver and releaser directives that follow global directives in the `archiver.cmd` file, are specific to a particular file system, and begin with `fs =`. File-system-specific directives apply until the next `fs =` directive line or the end of file is encountered. If multiple directives affect a file system, the file-system-specific directives override the global directives.

ftp

File Transfer Protocol, a network protocol for transferring files between two hosts. For a more secure alternative, see [sftp](#).

global directives

Archiver and releaser directives that apply to all file systems and that appear before the first `fs=` line.

grace period

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.

hard limit

In a [quota](#), the absolute maximum quantity of storage resources that specified user, group, and/or [admin set IDs](#) can consume. See [soft limit](#).

high-water mark

1. In an archiving file system, the percentage disk-cache utilization at which SAM-QFS file systems start 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` and `mount_samfs` 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 that starts the recycler process. Recycling empties partially full volumes of current data so that they can be replaced by new media or relabeled.

historian

The SAM-QFS historian is a catalog of volumes that have been exported from automated media libraries that are defined in the `/etc/opt/SUNWsamfs/mcf` file. By default, it is located on the SAM-QFS file-system host at `/var/opt/SUNWsamfs/catalog/historian`. For details, see the SAM-QFS `historian` man page.

hosts file

The hosts file contains a list of all of the hosts in a shared file system. If you are initializing a file system as a SAM-QFS shared file system, the hosts file, `/etc/opt/SUNWsamfs/hosts.fs-name`, must be created before the file system is created. The `sammkfs` command uses the hosts file when it creates the file system. You can use the `samsharefs` command to replace or update the contents of the hosts file at a later date.

indirect block

A disk block that contains a list of storage blocks. File systems have 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

Index node. A data structure used by the file system to describe a file. An inode describes all the attributes associated with a file other than the name. The attributes include ownership, access, permission, size, and the file location on the disk system.

inode file

A special file (`.inodes`) on the file system that contains the inode structures for all files resident in the file system. Inodes are 512 bytes long. The inode file is a metadata file, which is separated from file data in the file system.

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.

LAN

Local area network.

lease

A function that grants a client host permission to perform an operation on a file for a specified period of time. The metadata server issues leases to each client host. The leases are renewed as necessary to permit continued file operations.

library

See [automated library](#).

library catalog

See [catalog](#).

local file system

A file system that is installed on one node of a Solaris Cluster system and is not made highly available to another node. Also, a file system that is installed on a server.

low-water mark

In an archiving file system, the percentage disk-cache utilization at which SAM-QFS file systems 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` and `mount_samfs` man pages. Compare [high-water mark](#).

LUN

Logical unit number.

mcf

Master Configuration File. The file that is read at initialization time that defines the relationships between the devices (the topology) in a file system environment.

media

Tape or optical disk cartridges.

media recycling

The process of recycling or reusing archive media with few active files.

metadata

Data about data. Metadata is the index information used to locate the exact data position of a file on a disk. It consists of information about files, directories, access control lists, symbolic links, removable media, segmented files, and the indexes of segmented files.

metadata device

A device (for example, a solid-state disk or mirrored device) upon which file system metadata is stored. Having file data and metadata on separate devices can increase performance. In the `mcf` file, a metadata device is declared as an `mm` device within an `ma` file system.

mirror writing

The process of maintaining two copies of a file on disjointed sets of disks to prevent loss from a single disk failure.

mount point

The directory on which a file system is mounted.

multireader file system

A single-writer, multireader capability that enables you to specify a file system that can be mounted on multiple hosts. Multiple hosts can read the file system, but only one host can write to the file system. Multiple readers are specified with the `-o reader` option with the `mount` command. The single-writer host is specified with the `-o writer` option with the `mount` command. For more information, see the `mount_samfs` man page.

name space

The metadata portion of a collection of files that identifies the file, its attributes, and its storage locations.

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.

network attached automated library

A library, such as those from StorageTek, ADIC/Grau, IBM, or Sony, that is controlled using a software package supplied by the vendor. The QFS file system interfaces with the vendor software using a SAM-QFS media changer daemon designed specifically for the automated library.

NFS

Network file system, a file system that provides transparent access to remote file systems on heterogeneous networks.

NIS

Network Information Service, a distributed network database containing key information about systems and users on the network. The NIS database is stored on the master server and all slave servers.

offline storage

Storage that requires operator intervention for loading.

offsite storage

Storage that is remote from the server and is used for disaster recovery.

online storage

Storage that is immediately available, such as disk cache storage.

partition

A portion of a device or a side of a magneto-optical cartridge.

preallocation

The process of reserving a contiguous amount of space on the disk cache for writing a file. Preallocation can be specified only for a file that is size zero. For more information, see the `setfa` man page.

pseudo device

A software subsystem or driver with no associated hardware.

QFS

The SAM-QFS QFS Software product, a high-performance, high-capacity, UNIX file system that can be used on its own or as an archiving file system controlled by SAM-QFS Storage Archive Manager.

qfsdump

See [samfsdump \(qfsdump\)](#).

qfsrestore

See [samfsrestore \(qfsrestore\)](#).

quota

The amount of storage resources that specified user, group, or [admin set IDs](#) 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. It can protect against data loss from a single disk failure, can provide a fault-tolerant disk environment, 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 SAM-QFS 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.

recycler

A SAM-QFS utility that reclaims space on cartridges that is occupied by expired archive copies.

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 SAM-QFS file-system operations, see the SAM-QFS Solaris [regex](#) and [regcmp](#) man pages.

release priority

The priority according to which a file in a file system is released after being archived. Release priority is calculated by multiplication of various weights of file properties and then summation of the results.

releaser

A SAM-QFS 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.

remote procedure call

See [RPC](#).

removable media file

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. Also used for writing archive and stage file data.

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. The underlying data exchange mechanism used by NFS to implement custom network data servers.

SAM

See [SAM-QFS](#).

SAM-Remote client

A SAM-QFS system with a client daemon that contains a number of pseudodevices, and can also have its own library devices. The client depends on a SAM-Remote server for archive media for one or more archive copies.

SAM-Remote server

Both a full-capacity SAM-QFS storage management server and a SAM-Remote server daemon that defines libraries to be shared among SAM-Remote clients.

SAM-QFS

1. A common abbreviation for the SAM-QFS product.
2. An adjective describing a [QFS](#) file system that is configured for archiving and managed by SAM-QFS Storage Archive Manager software.

samfsdump (qfsdump)

A program that creates a control structure dump and copies all the control structure information for a given group of files. It does not generally copy file data. With the `-U` option, the command also copies data files. If the Storage Archive Manager packages are not installed, the command is called `qfsdump`.

samfsrestore (qfsrestore)

A program that restores inode and directory information from a control structure dump. See also [samfsdump \(qfsdump\)](#).

SAN

Storage Area Network.

SCSI

Small Computer System Interface, an electrical communication specification commonly used for peripheral devices such as disk and tape drives and automated libraries.

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 [hard limit](#).

sftp

Secure File Transfer Protocol, a secure implementation of [ftp](#) based on [ssh](#).

ssh

Secure Shell, an encrypted network protocol that allows secure, remote command-line login and command execution.

staging

The process of copying a nearline or offline file from archive storage back to online storage.

storage family set

A set of disks that are collectively represented by a single disk family device.

storage slots

Locations inside an automated library in which cartridges are stored when not being used in a drive.

stripe size

The number of disk allocation units (DAUs) to be allocated before writing proceeds to the next device of a stripe. If the `stripe=0` mount option is used, the file system uses round-robin access, not striped access.

striped group

A collection of devices within a file system that is defined in the `mcf` file as one or more `gXXX` devices. Striped groups are treated as one logical device and are always striped with a size equal to the disk allocation unit (DAU).

striping

A data access method in which files are simultaneously written to logical disks in an interlaced fashion. SAM-QFS file systems provide two types of striping: "hard

striping," using stripe groups, and "soft striping," using the `stripe=x` mount parameter. Hard striping is enabled when a file system is set up, and requires the definition of stripe groups within the `mcf` file. Soft striping is enabled through the `stripe=x` mount parameter, and can be changed for the file system or for individual files. It is disabled by setting `stripe=0`. Hard and soft striping can both be used if a file system is composed of multiple stripe groups with the same number of elements. See also [round-robin](#).

SUNW.qfs

A Solaris Cluster resource type that supports SAM-QFS shared file systems. The `SUNW.qfs` resource type defines failover resources for the shared file system's metadata server (MDS)

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](#).
2. A magnetic tape [cartridge](#) that holds a single logical volume.
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 [volumes](#). 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**.

WORM

Write-Once-Read-Many. A storage classification for media that can be written only once but read many times.