Veritas Storage Foundation™ Cluster File System Administrator's Guide

AIX

5.1

Veritas Storage Foundation™ Cluster File System Administrator's Guide

Product version: 5.1

Document version: 5.1.0

Legal Notice

Symantec Corporation
350 Ellis Street
Mountain View, CA 94043

http://www.symantec.com

Technical Support

Symantec Technical Support maintains support centers globally. Technical Support's primary role is to respond to specific queries about product features and functionality. The Technical Support group also creates content for our online Knowledge Base. The Technical Support group works collaboratively with the other functional areas within Symantec to answer your questions in a timely fashion. For example, the Technical Support group works with Product Engineering and Symantec Security Response to provide alerting services and virus definition updates.

Symantec's maintenance offerings include the following:

- A range of support options that give you the flexibility to select the right amount of service for any size organization
- Telephone and Web-based support that provides rapid response and up-to-the-minute information
- Upgrade assurance that delivers automatic software upgrade protection
- Global support that is available 24 hours a day, 7 days a week
- Advanced features, including Account Management Services

For information about Symantec's Maintenance Programs, you can visit our Web site at the following URL:

www.symantec.com/business/support/index.jsp

Contacting Technical Support

Customers with a current maintenance agreement may access Technical Support information at the following URL:

www.symantec.com/business/support/contact_techsupp_static.jsp

Before contacting Technical Support, make sure you have satisfied the system requirements that are listed in your product documentation. Also, you should be at the computer on which the problem occurred, in case it is necessary to replicate the problem.

When you contact Technical Support, please have the following information available:

- Product release level
- Hardware information
- Available memory, disk space, and NIC information
- Operating system

- Version and patch level
- Network topology
- Router, gateway, and IP address information
- Problem description:
  - Error messages and log files
  - Troubleshooting that was performed before contacting Symantec
  - Recent software configuration changes and network changes

Licensing and registration

If your Symantec product requires registration or a license key, access our non-technical support Web page at the following URL:

customercare.symantec.com

Customer service

Customer Care information is available at the following URL:

www.symantec.com/customercare

Customer Service is available to assist with the following types of issues:

- Questions regarding product licensing or serialization
- Product registration updates, such as address or name changes
- General product information (features, language availability, local dealers)
- Latest information about product updates and upgrades
- Information about upgrade assurance and maintenance contracts
- Information about the Symantec Buying Programs
- Advice about Symantec's technical support options
- Nontechnical presales questions
- Issues that are related to CD-ROMs or manuals

Documentation feedback

Your feedback on product documentation is important to us. Send suggestions for improvements and reports on errors or omissions. Include the title and document version (located on the second page), and chapter and section titles of the text on which you are reporting. Send feedback to:

sfha_docs@symantec.com

Maintenance agreement resources

If you want to contact Symantec regarding an existing maintenance agreement, please contact the maintenance agreement administration team for your region as follows:

Asia-Pacific and Japan	customercare_apac@symantec.com
Europe, Middle-East, and Africa	semea@symantec.com
North America and Latin America	supportsolutions@symantec.com

Additional enterprise services

Symantec offers a comprehensive set of services that allow you to maximize your investment in Symantec products and to develop your knowledge, expertise, and global insight, which enable you to manage your business risks proactively.

Enterprise services that are available include the following:

Symantec Early Warning Solutions	These solutions provide early warning of cyber attacks, comprehensive threat analysis, and countermeasures to prevent attacks before they occur.
Managed Security Services	These services remove the burden of managing and monitoring security devices and events, ensuring rapid response to real threats.
Consulting Services	Symantec Consulting Services provide on-site technical expertise from Symantec and its trusted partners. Symantec Consulting Services offer a variety of prepackaged and customizable options that include assessment, design, implementation, monitoring, and management capabilities. Each is focused on establishing and maintaining the integrity and availability of your IT resources.
Educational Services	Educational Services provide a full array of technical training, security education, security certification, and awareness communication programs.

To access more information about Enterprise services, please visit our Web site at the following URL:

www.symantec.com

Select your country or language from the site index.

Contents

Chapter 1

Technical overview

This chapter includes the following topics:

Storage Foundation Cluster File System architecture

The Veritas Storage Foundation Cluster File System (SFCFS) allows clustered servers to mount and use a file system simultaneously as if all applications using the file system were running on the same server. The Veritas Volume Manager cluster functionality (CVM) makes logical volumes and raw device applications accessible throughout a cluster.

This section includes the following topics:

About the symmetric architecture

SFCFS uses a symmetric architecture in which all nodes in the cluster can simultaneously function as metadata servers. SFCFS still has some remnants of the old master/slave or primary/secondary concept. The first server to mount each cluster file system becomes its primary; all other nodes in the cluster become secondaries. Applications access the user data in files directly from the server on which they are running. Each SFCFS node has its own intent log. File system operations, such as allocating or deleting files, can originate from any node in the cluster.

About Storage Foundation Cluster File System primary/secondary failover

If the server on which the SFCFS primary is running fails, the remaining cluster nodes elect a new primary. The new primary reads the intent log of the old primary and completes any metadata updates that were in process at the time of the failure.

If a server on which an SFCFS secondary is running fails, the primary reads the intent log of the failed secondary and completes any metadata updates that were in process at the time of the failure.

See "When the Storage Foundation Cluster File System primary fails" on page 61.

About single-host file system semantics using Group Lock Manager

SFCFS uses the Veritas Group Lock Manager (GLM) to reproduce UNIX single-host file system semantics in clusters. This is most important in write behavior. UNIX file systems make writes appear to be atomic. This means that when an application writes a stream of data to a file, any subsequent application that reads from the same area of the file retrieves the new data, even if it has been cached by the file system and not yet written to disk. Applications can never retrieve stale data, or partial results from a previous write.

To reproduce single-host write semantics, system caches must be kept coherent and each must instantly reflect any updates to cached data, regardless of the cluster node from which they originate. GLM locks a file so that no other node in the cluster can update it simultaneously, or read it before the update is complete.

About Veritas File System features supported in cluster file systems

The Veritas Storage Foundation Cluster File System is based on the Veritas File System (VxFS).

Most of the major features of VxFS local file systems are available on cluster file systems, including the following features:

- Extent-based space management that maps files up to a terabyte in size
- Fast recovery from system crashes using the intent log to track recent file system metadata updates
- Online administration that allows file systems to be extended and defragmented while they are in use

The list of supported features and commands that operate on SFCFS. Every VxFS manual page has a section on Storage Foundation Cluster File System Issues with

information on whether the command functions on a cluster-mounted file system and indicates any difference in behavior from local mounted file systems.

See the *Veritas Storage Foundation Release Notes*.

Veritas File System features in cluster file systems

Table 1-1 describes the VxFS supported features and commands for SFCFS.

Table 1-1 Veritas File System features in cluster file systems

Features	Description
Quick I/O for databases	The Quick I/O for Databases feature, using clusterized Oracle Disk Manager (ODM), is supported on SFCFS.
Storage Checkpoints	Storage Checkpoints are supported on cluster file systems, but are licensed only with other Veritas products.
Snapshots	Snapshots are supported on cluster file systems.
Quotas	Quotas are supported on cluster file systems.
NFS mounts	You export the NFS file systems from the Cluster. You can NFS export CFS file systems in a distributed highly available way.
Nested Mounts	You can use a directory on a cluster mounted file system as a mount point for a local file system or another cluster file system.
Freeze and thaw	Synchronizing operations, which require freezing and thawing file systems, are done on a cluster-wide basis.
Memory mapping	Shared memory mapping established by the `mmap`() function is supported on SFCFS. See the `mmap`(2) manual page.
Disk layout versions	SFCFS supports only disk layout Version 6 and 7. Use the `fstyp -v` *`special_device`* command to ascertain the disk layout version of a VxFS file system. Use the `vxupgrade` command to update the disk layout version.

Table 1-1 Veritas File System features in cluster file systems *(continued)*

Features	Description
Locking	Advisory file and record locking are supported on SFCFS. For the F_GETLK command, if there is a process holding a conflicting lock, the `l_pid` field returns the process ID of the process holding the conflicting lock. The nodeid-to-node name translation can be done by examining the `/etc/llthosts` file or with the `fsclustadm` command. Mandatory locking, and deadlock detection supported by traditional `fcntl` locks, are not supported on SFCFS. See the `fcntl`(2) manual page.

Veritas File System features not in cluster file systems

Table 1-2 describes functionality as not supported and may not be expressly prevented from operating on cluster file systems, but the actual behavior is indeterminate.

It is not advisable to use unsupported functionality on SFCFS, or to alternate mounting file systems with these options as local and cluster mounts.

Table 1-2 Veritas File System features not in cluster file systems

Unsupported features	Comments
qlog	Quick log is not supported.
Swap files	Swap files are not supported on cluster mounted file system.
mknod	The `mknod` command cannot be used to create devices on a cluster mounted file system.
Cache advisories	Cache advisories are set with the mount command on individual file systems, but are not propagated to other nodes of a cluster.
Cached Quick I/O	This Quick I/O for Databases feature that caches data in the file system cache is not supported.
Commands that depend on file access times	File access times may appear different across nodes because the `atime` file attribute is not closely synchronized in a cluster file system. So utilities that depend on checking access times may not function reliably.

Storage Foundation Cluster File System benefits and applications

This section describes the SFCFS benefits and applications.

This section includes the following topics:

- How Storage Foundation Cluster File System works
- When to use Storage Foundation Cluster File System

How Storage Foundation Cluster File System works

SFCFS simplifies or eliminates system administration tasks that result from the following hardware limitations:

- The SFCFS single file system image administrative model simplifies administration by making all file system management operations can be performed from any node.
- Because all servers in a cluster have access to SFCFS cluster-shareable file systems, keeping data consistent across multiple servers is automatic. All cluster nodes have access to the same data, and all data is accessible by all servers using single server file system semantics.
- Because all files can be accessed by all servers, applications can be allocated to servers to balance load or meet other operational requirements. Similarly, failover becomes more flexible because it is not constrained by data accessibility.
- Because each SFCFS file system can be on any node in the cluster, the file system recovery portion of failover time in an *n*-node cluster can be reduced by a factor of *n* by distributing the file systems uniformly across cluster nodes.
- Enterprise RAID subsystems can be used more effectively because all of their capacity can be mounted by all servers, and allocated by using administrative operations instead of hardware reconfigurations.
- Larger volumes with wider striping improve application I/O load balancing. Not only is the I/O load of each server spread across storage resources, but with SFCFS shared file systems, the loads of all servers are balanced against each other.
- Extending clusters by adding servers is easier because each new server's storage configuration does not need to be set up—new servers simply adopt the cluster-wide volume and file system configuration.

- The clusterized Oracle Disk Manager (ODM) feature that makes file-based databases perform as well as raw partition-based databases is available to applications running in a cluster.

When to use Storage Foundation Cluster File System

You should use SFCFS for any application that requires the sharing of files, such as for home directories and boot server files, Web pages, and for cluster-ready applications. SFCFS is also applicable when you want highly available standby data, in predominantly read-only environments where you just need to access data, or when you do not want to rely on NFS for file sharing.

Almost all applications can benefit from SFCFS. Applications that are not "cluster-aware" can operate on and access data from anywhere in a cluster. If multiple cluster applications running on different servers are accessing data in a cluster file system, overall system I/O performance improves due to the load balancing effect of having one cluster file system on a separate underlying volume. This is automatic; no tuning or other administrative action is required.

Many applications consist of multiple concurrent threads of execution that could run on different servers if they had a way to coordinate their data accesses. SFCFS provides this coordination. Such applications can be made cluster-aware allowing their instances to co-operate to balance client and data access load, and thereby scale beyond the capacity of any single server. In such applications, SFCFS provides shared data access, enabling application-level load balancing across cluster nodes.

SFCFS provides the following features:

- For single-host applications that must be continuously available, SFCFS can reduce application failover time because it provides an already-running file system environment in which an application can restart after a server failure.
- For parallel applications, such as distributed database management systems and Web servers, SFCFS provides shared data to all application instances concurrently. SFCFS also allows these applications to grow by the addition of servers, and improves their availability by enabling them to redistribute load in the event of server failure simply by reassigning network addresses.
- For workflow applications, such as video production, in which very large files are passed from station to station, the SFCFS eliminates time consuming and error prone data copying by making files available at all stations.
- For backup, the SFCFS can reduce the impact on operations by running on a separate server, accessing data in cluster-shareable file systems.

The following are examples of applications and how they might work with SFCFS:

- Using Storage Foundation Cluster File System on file servers

Two or more servers connected in a cluster configuration (that is, connected to the same clients and the same storage) serve separate file systems. If one of the servers fails, the other recognizes the failure, recovers, assumes the primaryship, and begins responding to clients using the failed server's IP addresses.

- Using Storage Foundation Cluster File System on web servers
  Web servers are particularly suitable to shared clustering because their application is typically read-only. Moreover, with a client load balancing front end, a Web server cluster's capacity can be expanded by adding a server and another copy of the site. A SFCFS-based cluster greatly simplifies scaling and administration for this type of application.

Chapter 2

Storage Foundation Cluster File System architecture

This chapter includes the following topics:

About Veritas Cluster Server architecture

The Group Membership and Atomic Broadcast (GAB) and Low Latency Transport (LLT) are VCS-specific protocols implemented directly on Ethernet data link. They run on redundant data links that connect the nodes in a cluster. VCS requires redundant cluster communication links to avoid single points of failure.

GAB provides membership and messaging for the cluster and its applications. GAB membership also provides orderly startup and shutdown of a cluster. The `/etc/gabtab` file is used to configure GAB. This file contain the `gabconfig` command run by GAB on startup. For example, the `-n` option of the command specifies the number of nodes in the cluster. GAB is configured automatically when you run the SFCFS installation script, but you may have to reconfigure GAB when adding nodes to a cluster.

See the `gabconfig`(1M) manual page.

LLT provides kernel-to-kernel communications and monitors network communications. The LLT `/etc/llthosts` and `/etc/llttab` files are configured to set system IDs within a cluster, set cluster IDs for multiple clusters, and tune network parameters such as heartbeat frequency. LLT is implemented so that cluster membership changes are reflected quickly, which in turn enables fast responses.

As with GAB, LLT is configured automatically when you run the VCS installation script. The `/etc/llttab` and `/etc/llthosts` files contain information you provide during installation. You may also have to reconfigure LLT when adding nodes to a cluster.

See the `llttab`(4) and the `llthosts`(4) manual pages.

See the *Veritas Cluster Server User's Guide*.

Each component in SFCFS registers with a GAB membership port. The port membership identifies nodes that have formed a cluster for the individual components.

Table 2-1 describes the port memberships.

Table 2-1 Port memberships

Port	Description
port a	heartbeat membership
port b	I/O fencing membership
port d	Oracle Disk Manager (ODM) membership
port f	Cluster File system membership
port h	Veritas Cluster Server communication between GAB and High Availability Daemon (HAD)
port v	Cluster Volume Manager membership
port w	Cluster Volume Manager daemons on different nodes communicate with one another using this port, but receive cluster membership information through GAB (port v)

About Storage Foundation Cluster File System and the Group Lock Manager

SFCFS uses the Veritas Group Lock Manager (GLM) to reproduce UNIX single-host file system semantics in clusters. UNIX file systems make writes appear atomic. This means when an application writes a stream of data to a file, a subsequent application reading from the same area of the file retrieves the new data, even if it has been cached by the file system and not yet written to disk. Applications cannot retrieve stale data or partial results from a previous write.

To reproduce single-host write semantics, the file system must keep system caches coherent, and each must instantly reflect updates to cached data, regardless of the node from which the updates originate.

Storage Foundation Cluster File System namespace

The mount point name must remain the same for all nodes mounting the same cluster file system. This is required for the VCS mount agents (online, offline, and monitoring) to work correctly.

About asymmetric mounts

A VxFS file system mounted with the `mount -o cluster` option is a cluster, or shared mount, as opposed to a non-shared or local mount. A file system mounted in shared mode must be on a VxVM shared volume in a cluster environment. A local mount cannot be remounted in shared mode and a shared mount cannot be remounted in local mode when you use the `mount -o remount` option. A single clustered file system can be mounted with different read/writes options on different nodes. These are called asymmetric mounts.

Asymmetric mounts allow shared file systems to be mounted with different read/write capabilities. For example, one node in the cluster can mount read/write, while other nodes mount read-only.

When a primary mounts "ro", this means that neither this node nor any other node is allowed to write the file system. Secondaries can only mount "ro", if the primary mounts "ro". Otherwise, the primary mounts either "rw" or "ro,crw", and the secondaries have the same choice.

You can specify the cluster read-write (`crw`) option when you first mount the file system, or the options can be altered when doing a remount (`mount -o remount`).

See the `mount_vxfs`(1M) manual page.

Figure 2-1 describes the first column showing the mode in which the primary is mounted:

Figure 2-1 Primary and secondary mounts

Primary \ Secondary	ro	rw	ro, crw
ro	X		
rw		X	X
ro, crw		X	X

The check marks indicate the mode secondary mounts can use for a given mode of the primary.

Mounting the primary with only the `-o cluster,ro` option prevents the secondaries from mounting in a different mode; that is, read-write.

Note: `rw` implies read-write capability throughout the cluster.

Primary and secondary

A file system cluster consists of one primary, and up to 31 secondaries. The primary-secondary terminology applies to one file system, not to a specific node (or hardware platform). You can have the same cluster node be primary for one shared file system, while at the same time it is secondary for another shared file system. Such distribution of file system primaryship to balance the load on a cluster is a recommended administrative policy.

See "Distribute the load on a cluster" on page 61.

For CVM, a single cluster node is the master for all shared disk groups and shared volumes in the cluster.

See "About Veritas Volume Manager cluster functionality" on page 47.

Determining or moving primaryship

The first node of a cluster file system to mount is called the primary node. Other nodes are called secondary nodes. If a primary node fails, an internal election process determines which of the secondaries becomes the primary file system.

To determine primaryship

- To determine primaryship, type the following command:

```
# fsclustadm -v showprimary mount_point
```

To give primaryship to a node

- To give primaryship to a node, type the following command on the node:

```
# fsclustadm -v setprimary mount_point
```

Synchronize time on Cluster File Systems

SFCFS requires that the system clocks on all nodes are synchronized using some external component such as the Network Time Protocol (NTP) daemon. If the

nodes are not in sync, timestamps for inode (`ctime`) and data modification (`mtime`) may not be consistent with the sequence in which operations actually happened.

File system tuneables

Using the `tunefstab` file will update the tuneable parameters at the time of mount. The file system `tunefstab` parameters are set to be identical on all nodes by propagating the parameters to each cluster node. When the file system is mounted on the node, the `tunefstab` parameters of the primary node are used. Symantec recommends that this file be identical on each node.

Setting the number of parallel fsck threads

This section describes how to set the number of parallel fsck threads.

The number of parallel fsck threads that could be active during recovery was set to 4. For example, if a node failed over 12 file systems, log replay for the 12 file systems will not complete at the same time. The number was set to 4 since parallel replay of a large number of file systems would put memory pressure on systems with less memory. However, on larger systems the restriction of 4 parallel processes replaying is not necessary.

This value gets tuned in accordance with available physical memory in the system.

To set the number of parallel fsck threads

- On all nodes in the cluster, edit the `/opt/VRTSvcs/bin/CFSfsckd/CFSfsckd.env` file and set `FSCKD_OPTS="-n N"`.

  where *N* is the number of parallel fsck threads desired and value of *N* has to be between 4 and 128.

  Note: The default number of parallel fsck threads is determined by the amount of memory on the machine, unless overridden the user.

About Storage Checkpoints

Veritas File System (VxFS) provides a Storage Checkpoint feature that quickly creates a persistent image of a file system at an exact point in time. Storage Checkpoints significantly reduce I/O overhead by identifying and maintaining

only the file system blocks that have changed since the last Storage Checkpoint or backup via a copy-on-write technique.

Storage Checkpoints provide:

- Persistence through reboots and crashes.
- The ability for data to be immediately writeable by preserving the file system metadata, the directory hierarchy, and user data.

Storage Checkpoints are actually data objects that are managed and controlled by the file system. You can create, remove, and rename Storage Checkpoints because they are data objects with associated names.

Unlike a disk-based mirroring technology that requires a separate storage space, Storage Checkpoints minimize the use of disk space by using a Storage Checkpoint within the same free space available to the file system.

After you create a Storage Checkpoint of a mounted file system, you can also continue to create, remove, and update files on the file system without affecting the logical image of the Storage Checkpoint. A Storage Checkpoint preserves not only the name space (directory hierarchy) of the file system, but also the user data as it existed at the moment the file system image was captured.

You can use a Storage checkpoint in many ways. For example, you can use them to:

- Create a stable image of the file system that can be backed up to tape.
- Provide a mounted, on-disk backup of the file system so that end users can restore their own files in the event of accidental deletion. This is especially useful in a home directory, engineering, or email environment.
- Create a copy of an application's binaries before installing a patch to allow for rollback in case of problems.
- Create an on-disk backup of the file system in that can be used in addition to a traditional tape-based backup to provide faster backup and restore capabilities.
- Test new software on a point-in-time image of the primary fileset without jeopardizing the live data in the current primary fileset by mounting the Storage Checkpoints as writable.

Storage Foundation Cluster File System backup strategies

The same backup strategies used for standard VxFS can be used with SFCFS because the APIs and commands for accessing the namespace are the same. File

System checkpoints provide an on-disk, point-in-time copy of the file system. Because performance characteristics of a checkpointed file system are better in certain I/O patterns, they are recommended over file system snapshots (described below) for obtaining a frozen image of the cluster file system.

File System snapshots are another method of a file system on-disk frozen image. The frozen image is non-persistent, in contrast to the checkpoint feature. A snapshot can be accessed as a read-only mounted file system to perform efficient online backups of the file system. Snapshots implement "copy-on-write" semantics that incrementally copy data blocks when they are overwritten on the snapped file system. Snapshots for cluster file systems extend the same copy-on-write mechanism for the I/O originating from any cluster node.

Mounting a snapshot filesystem for backups increases the load on the system because of the resources used to perform copy-on-writes and to read data blocks from the snapshot. In this situation, cluster snapshots can be used to do off-host backups. Off-host backups reduce the load of a backup application from the primary server. Overhead from remote snapshots is small when compared to overall snapshot overhead. Therefore, running a backup application by mounting a snapshot from a relatively less loaded node is beneficial to overall cluster performance.

The following are several characteristics of a cluster snapshot:

- A snapshot for a cluster mounted file system can be mounted on any node in a cluster. The file system can be a primary, secondary, or secondary-only. A stable image of the file system is provided for writes from any node.
  See the `mount_vxfs` manual page for more information on secondary-only (seconly) is a CFS mount option.
- Multiple snapshots of a cluster file system can be mounted on the same or different cluster nodes.
- A snapshot is accessible only on the node mounting the snapshot. The snapshot device cannot be mounted on two nodes simultaneously.
- The device for mounting a snapshot can be a local disk or a shared volume. A shared volume is used exclusively by a snapshot mount and is not usable from other nodes as long as the snapshot is mounted on that device.
- On the node mounting a snapshot, the snapped file system cannot be unmounted while the snapshot is mounted.
- A SFCFS snapshot ceases to exist if it is unmounted or the node mounting the snapshot fails. However, a snapshot is not affected if another node leaves or joins the cluster.

- A snapshot of a read-only mounted file system cannot be taken. It is possible to mount a snapshot of a cluster file system only if the snapped cluster file system is mounted with the `crw` option.

In addition to frozen images of file systems, there are volume-level alternatives available for shared volumes using mirror split and rejoin. Features such as Fast Mirror Resync and Space Optimized snapshot are also available.

See the *Veritas Volume Manager System Administrator's Guide.*

Parallel I/O

Some distributed applications read and write to the same file concurrently from one or more nodes in the cluster; for example, any distributed application where one thread appends to a file and there are one or more threads reading from various regions in the file. Several high-performance compute (HPC) applications can also benefit from this feature, where concurrent I/O is performed on the same file. Applications do not require any changes to use parallel I/O.

Traditionally, the entire file is locked to perform I/O to a small region. To support parallel I/O, SFCFS locks ranges in a file that correspond to I/O requests. The granularity of the locked range is a page. Two I/O requests conflict if at least one is a write request, and the I/O range of the request overlaps the I/O range of the other.

The parallel I/O feature enables I/O to a file by multiple threads concurrently, as long as the requests do not conflict. Threads issuing concurrent I/O requests could be executing on the same node, or on different nodes in the cluster.

An I/O request that requires allocation is not executed concurrently with other I/O requests. Note that when a writer is extending the file and readers are lagging behind, block allocation is not necessarily done for each extending write.

Predetermine the file size and preallocate the file to avoid block allocations during I/O. This improves the concurrency of applications performing parallel I/O to the file. Parallel I/O also avoids unnecessary page cache flushes and invalidations using range locking, without compromising the cache coherency across the cluster.

For applications that update the same file from multiple nodes, the `-nomtime` mount option provides further concurrency. Modification and change times of the file are not synchronized across the cluster, which eliminates the overhead of increased I/O and locking. The timestamp seen for these files from a node may not have the time updates that happened in the last 60 seconds.

I/O error handling policy

I/O errors can occur for several reasons, including failures of Fibre Channel links, host-bus adapters, and disks. SFCFS disables the file system on the node encountering I/O errors. The file system remains available from other nodes.

After the hardware error is fixed (for example, the Fibre Channel link is reestablished), the file system can be force unmounted and the mount resource can be brought online from the disabled node to reinstate the file system.

Recovering for I/O failures

The disabled file system can be restored by a force unmount and the resource will be brought online without rebooting, which also brings the shared disk group resource online.

Note: If the jeopardy condition is not fixed, the nodes are susceptible to leaving the cluster again on subsequent node failure.

See the *Veritas Cluster Server User's Guide*.

Single network link and reliability

Certain environments may prefer using a single private link or a public network for connecting nodes in a cluster, despite the loss of redundancy for dealing with network failures. The benefits of this approach include simpler hardware topology and lower costs; however, there is obviously a tradeoff with high availability.

For the above environments, SFCFS provides the option of a single private link, or using the public network as the private link if I/O fencing is present. I/O fencing is used to handle split-brain scenarios. The option for single network is given during installation.

See "About preventing data corruption with I/O fencing" on page 33.

Configuring low priority a link

LLT can be configured to use a low-priority network link as a backup to normal heartbeat channels. Low-priority links are typically configured on the customer's public or administrative network. This typically results in a completely different network infrastructure than the cluster private interconnect, and reduces the chance of a single point of failure bringing down all links. The low-priority link is not used for cluster membership traffic until it is the only remaining link. In

normal operation, the low-priority link carries only heartbeat traffic for cluster membership and link state maintenance. The frequency of heartbeats drops 50 percent to reduce network overhead. When the low-priority link is the only remaining network link, LLT also switches over all cluster status traffic. Following repair of any configured private link, LLT returns cluster status traffic to the high-priority link.

LLT links can be added or removed while clients are connected. Shutting down GAB or the high-availability daemon, had, is not required.

To add a link

- To add a link, type the following command:

  ```
  # lltconfig -d device -t device_tag
  ```

  where *device_tag* is a tag to identify particular link in subsequent commands, and is displayed by `lltstat`(1M).

To remove a link

- To remove a link, type the following command:

  ```
  # lltconfig -u device_tag
  ```

See the `lltconfig`(1M) manual page.

Changes take effect immediately and are lost on the next reboot. For changes to span reboots you must also update the `/etc/lltab` file.

Note: LLT clients will not know how things are going until you only have one LLT link left and GAB declares jeopardy

---

Split-brain and jeopardy handling

A split-brain occurs when the cluster membership view differs among the cluster nodes, increasing the chance of data corruption. With I/O fencing, the potential for data corruption is eliminated. I/O fencing requires disks that support SCSI-3 PGR.

Jeopardy state

In the absence of I/O fencing, SFCFS installation requires two heartbeat links. When a node is down to a single heartbeat connection, SFCFS can no longer discriminate between loss of a system and loss of the final network connection. This state is defined as jeopardy.

SFCFS detects jeopardy and responds to it in ways that prevent data corruption in some split-brain situations. However, data corruption can still occur in other situations:

- All links go down simultaneously.
- A node hangs and is unable to respond to heartbeat messages.

To eliminate the chance of data corruption in these scenarios, I/O fencing is required. With I/O fencing, the jeopardy state does not require special handling by the SFCFS stack.

Jeopardy handling

For installations that do not support SCSI-3 PGR, jeopardy handling prevents some potential split-brain conditions. If any cluster node fails following a jeopardy state notification, all cluster file systems that were mounted on the failed node or nodes are disabled on all remaining nodes. If a leave reconfiguration happens after jeopardy state notification, then the nodes which have received the jeopardy state notification leave the cluster.

About I/O fencing

I/O fencing protects the data on shared disks when nodes in a cluster detect a change in the cluster membership that indicates a split-brain condition.

The fencing operation determines the following:

- The nodes that must retain access to the shared storage
- The nodes that must be ejected from the cluster

This decision prevents possible data corruption. The installsfcfs installs the SFCFS I/O fencing driver, VRTSvxfen. To protect data on shared disks, you must configure I/O fencing after you install and configure SFCFS.

See the *Storage Foundation Cluster File System Installation Guide*.

I/O fencing technology uses coordination points for arbitration in the event of a network partition.

You can configure I/O fencing to use one or both of the following components as coordination points:

Coordinator disk	I/O fencing that uses coordinator disks is referred to as disk-based I/O fencing. Disk-based I/O fencing ensures data integrity in a single cluster.

Coordination point server (CP server)	I/O fencing that uses at least one CP server system is referred to as server-based I/O fencing. Server-based I/O fencing ensures data integrity in multiple clusters.

See "About preventing data corruption with I/O fencing" on page 33.

About preventing data corruption with I/O fencing

I/O fencing is a feature that prevents data corruption in the event of a communication breakdown in a cluster.

To provide high availability, the cluster must be capable of taking corrective action when a node fails. In this situation, SFCFS configures its components to reflect the altered membership.

Problems arise when the mechanism that detects the failure breaks down because symptoms appear identical to those of a failed node. For example, if a system in a two-node cluster fails, the system stops sending heartbeats over the private interconnects. The remaining node then takes corrective action. The failure of the private interconnects, instead of the actual nodes, presents identical symptoms and causes each node to determine its peer has departed. This situation typically results in data corruption because both nodes try to take control of data storage in an uncoordinated manner.

In addition to a broken set of private networks, other scenarios can generate this situation. If a system is so busy that it appears to stop responding or "hang," the other nodes could declare it as dead. This declaration may also occur for the nodes that use the hardware that supports a "break" and "resume" function. When a node drops to PROM level with a break and subsequently resumes operations, the other nodes may declare the system dead. They can declare it dead even if the system later returns and begins write operations.

SFCFS uses I/O fencing to remove the risk that is associated with split-brain. I/O fencing allows write access for members of the active cluster. It blocks access to storage from non-members.

About SCSI-3 Persistent Reservations

SCSI-3 Persistent Reservations (SCSI-3 PR) are required for I/O fencing and resolve the issues of using SCSI reservations in a clustered SAN environment. SCSI-3 PR enables access for multiple nodes to a device and simultaneously blocks access for other nodes.

SCSI-3 reservations are persistent across SCSI bus resets and support multiple paths from a host to a disk. In contrast, only one host can use SCSI-2 reservations with one path. If the need arises to block access to a device because of data integrity concerns, only one host and one path remain active. The requirements for larger clusters, with multiple nodes reading and writing to storage in a controlled manner, make SCSI-2 reservations obsolete.

SCSI-3 PR uses a concept of registration and reservation. Each system registers its own "key" with a SCSI-3 device. Multiple systems registering keys form a membership and establish a reservation, typically set to "Write Exclusive Registrants Only." The WERO setting enables only registered systems to perform write operations. For a given disk, only one reservation can exist amidst numerous registrations.

With SCSI-3 PR technology, blocking write access is as easy as removing a registration from a device. Only registered members can "eject" the registration of another member. A member wishing to eject another member issues a "preempt and abort" command. Ejecting a node is final and atomic; an ejected node cannot eject another node. In SFCFS, a node registers the same key for all paths to the device. A single preempt and abort command ejects a node from all paths to the storage device.

About I/O fencing operations

I/O fencing, provided by the kernel-based fencing module (vxfen), performs identically on node failures and communications failures. When the fencing module on a node is informed of a change in cluster membership by the GAB module, it immediately begins the fencing operation. The node tries to eject the key for departed nodes from the coordinator disks using the preempt and abort command. When the node successfully ejects the departed nodes from the coordinator disks, it ejects the departed nodes from the data disks. In a split-brain scenario, both sides of the split would race for control of the coordinator disks. The side winning the majority of the coordinator disks wins the race and fences the loser. The loser then panics and restarts the system.

About I/O fencing components

The shared storage for SFCFS must support SCSI-3 persistent reservations to enable I/O fencing. SFCFS involves two types of shared storage:

- Data disks—Store shared data
  See “About data disks” on page 35.
- Coordination points—Act as a global lock during membership changes
  See “About coordination points” on page 35.

About data disks

Data disks are standard disk devices for data storage and are either physical disks or RAID Logical Units (LUNs). These disks must support SCSI-3 PR and are part of standard VxVM or CVM disk groups.

CVM is responsible for fencing data disks on a disk group basis. Disks that are added to a disk group and new paths that are discovered for a device are automatically fenced.

About coordination points

Coordination points provide a lock mechanism to determine which nodes get to fence off data drives from other nodes. A node must eject a peer from the coordination points before it can fence the peer from the data drives. Racing for control of the coordination points to fence data disks is the key to understand how fencing prevents split-brain.

The coordination points can either be disks or servers or both. Typically, a cluster must have three coordination points.

- Coordinator disks

  Disks that act as coordination points are called coordinator disks. Coordinator disks are three standard disks or LUNs set aside for I/O fencing during cluster reconfiguration. Coordinator disks do not serve any other storage purpose in the SFCFS configuration.

  You can configure coordinator disks to use Veritas Volume Manager Dynamic Multipathing (DMP) feature. Dynamic Multipathing (DMP) allows coordinator disks to take advantage of the path failover and the dynamic adding and removal capabilities of DMP. So, you can configure I/O fencing to use either DMP devices or the underlying raw character devices. I/O fencing uses SCSI-3 disk policy that is either raw or dmp based on the disk device that you use. The disk policy is dmp by default.

  See the *Veritas Volume Manager Administrator's Guide*.

- Coordination point servers

  The coordination point server (CP server) is a software solution which runs on a remote system or cluster. CP server provides arbitration functionality by allowing the SFCFS cluster nodes to perform the following tasks:

  - Self-register to become a member of an active SFCFS cluster (registered with CP server) with access to the data drives
  - Check which other nodes are registered as members of this activeSFCFS cluster
  - Self-unregister from this activeSFCFS cluster

- Forcefully unregister other nodes (preempt) as members of this active SFCFS cluster

In short, the CP server functions as another arbitration mechanism that integrates within the existing I/O fencing module.

Note: With the CP server, the fencing arbitration logic still remains on the SFCFS cluster.

Multiple SFCFS clusters running different operating systems can simultaneously access the CP server. TCP/IP based communication is used between the CP server and the SFCFS clusters.

About I/O fencing configuration files

Table 2-2 lists the I/O fencing configuration files.

Table 2-2 I/O fencing configuration files

File	Description
/etc/vxfendg	This file includes the coordinator disk group information. This file is not applicable for server-based fencing.

Table 2-2 I/O fencing configuration files *(continued)*

File	Description
/etc/vxfenmode	This file contains the following parameters: ■ vxfen_mode ■ scsi3—For disk-based fencing ■ customized—For server-based fencing ■ disabled—To run the I/O fencing driver but not do any fencing operations. ■ vxfen_mechanism This parameter is applicable only for server-based fencing. Set the value as cps. ■ scsi3_disk_policy ■ dmp—Configure the vxfen module to use DMP devices The disk policy is dmp by default. If you use iSCSI devices, you must set the disk policy as dmp. ■ raw—Configure the vxfen module to use the underlying raw character devices **Note:** You must use the same SCSI-3 disk policy on all the nodes. ■ security This parameter is applicable only for server-based fencing. 1—Indicates that Symantec Product Authentication Service is used for CP server communications. This setting is the default. 0—Indicates that communication with the CP server is in non-secure mode. **Note:** The CP server and the Storage Foundation Cluster File System HA clusters must have the same security setting. ■ List of coordination points This list is required only for server-based fencing configuration. Coordination points in a server-based fencing can include coordinator disks, CP servers, or a mix of both. If you use coordinator disks, you must create a coordinator disk group with the coordinator disk names. Refer to the sample file /etc/vxfen.d/vxfenmode_cps for more information on how to specify the coordination points.

Table 2-2 I/O fencing configuration files *(continued)*

File	Description

/etc/vxfentab	When I/O fencing starts, the vxfen startup script creates this /etc/vxfentab file on each node. The startup script uses the contents of the /etc/vxfendg and /etc/vxfenmode files. Any time a system is rebooted, the fencing driver reinitializes the vxfentab file with the current list of all the coordinator points.
	Note: The /etc/vxfentab file is a generated file; do not modify this file.
	For disk-based I/O fencing, the /etc/vxfentab file on each node contains a list of all paths to each coordinator disk. An example of the /etc/vxfentab file in a disk-based fencing configuration on one node resembles as follows:
	■ Raw disk:
	`/dev/hdisk75`
	`/dev/hdisk76`
	`/dev/hdisk77`
	■ DMP disk:
	`/dev/vx/rdmp/hdisk75`
	`/dev/vx/rdmp/hdisk76`
	`/dev/vx/rdmp/hdisk77`
	For server-based fencing, the /etc/vxfentab file also includes the security settings information.

How disk-based I/O fencing works in different event scenarios

Table 2-3 describes how I/O fencing works to prevent data corruption in different failure event scenarios. For each event, review the corrective operator actions.

Table 2-3 I/O fencing scenarios

Event	Node A: What happens?	Node B: What happens?	Operator action
Both private networks fail.	Node A races for majority of coordinator disks. If Node A wins race for coordinator disks, Node A ejects Node B from the shared disks and continues.	Node B races for majority of coordinator disks. If Node B loses the race for the coordinator disks, Node B panics and removes itself from the cluster.	When Node B is ejected from cluster, repair the private networks before attempting to bring Node B back.

Table 2-3 I/O fencing scenarios *(continued)*

Event	Node A: What happens?	Node B: What happens?	Operator action
Both private networks function again after event above.	Node A continues to work.	Node B has crashed. It cannot start the database since it is unable to write to the data disks.	Restart Node B after private networks are restored.
One private network fails.	Node A prints message about an IOFENCE on the console but continues.	Node B prints message about an IOFENCE on the console but continues.	Repair private network. After network is repaired, both nodes automatically use it.
Node A hangs.	Node A is extremely busy for some reason or is in the kernel debugger. When Node A is no longer hung or in the kernel debugger, any queued writes to the data disks fail because Node A is ejected. When Node A receives message from GAB about being ejected, it panics and removes itself from the cluster.	Node B loses heartbeats with Node A, and races for a majority of coordinator disks. Node B wins race for coordinator disks and ejects Node A from shared data disks.	Verify private networks function and restart Node A.

Table 2-3 I/O fencing scenarios *(continued)*

Event	Node A: What happens?	Node B: What happens?	Operator action
Nodes A and B and private networks lose power. Coordinator and data disks retain power. Power returns to nodes and they restart, but private networks still have no power.	Node A restarts and I/O fencing driver (vxfen) detects Node B is registered with coordinator disks. The driver does not see Node B listed as member of cluster because private networks are down. This causes the I/O fencing device driver to prevent Node A from joining the cluster. Node A console displays: `Potentially a preexisting split brain. Dropping out of the cluster. Refer to the user documentation for steps required to clear preexisting split brain.`	Node B restarts and I/O fencing driver (vxfen) detects Node A is registered with coordinator disks. The driver does not see Node A listed as member of cluster because private networks are down. This causes the I/O fencing device driver to prevent Node B from joining the cluster. Node B console displays: `Potentially a preexisting split brain. Dropping out of the cluster. Refer to the user documentation for steps required to clear preexisting split brain.`	Resolve preexisting split-brain condition. See "System panics to prevent potential data corruption" on page 173.

Table 2-3 I/O fencing scenarios *(continued)*

Event	Node A: What happens?	Node B: What happens?	Operator action
Node A crashes while Node B is down. Node B comes up and Node A is still down.	Node A is crashed.	Node B restarts and detects Node A is registered with the coordinator disks. The driver does not see Node A listed as member of the cluster. The I/O fencing device driver prints message on console: `Potentially a preexisting split brain. Dropping out of the cluster. Refer to the user documentation for steps required to clear preexisting split brain.`	Resolve preexisting split-brain condition. See "System panics to prevent potential data corruption" on page 173.
The disk array containing two of the three coordinator disks is powered off. Node B leaves the cluster and the disk array is still powered off.	Node A continues to operate as long as no nodes leave the cluster. Node A races for a majority of coordinator disks. Node A fails because only one of three coordinator disks is available. Node A panics and removes itself from the cluster.	Node B continues to operate as long as no nodes leave the cluster. Node B leaves the cluster.	Power on failed disk array and restart I/O fencing driver to enable Node A to register with all coordinator disks.

About secure communication between the SFCFS cluster and CP server

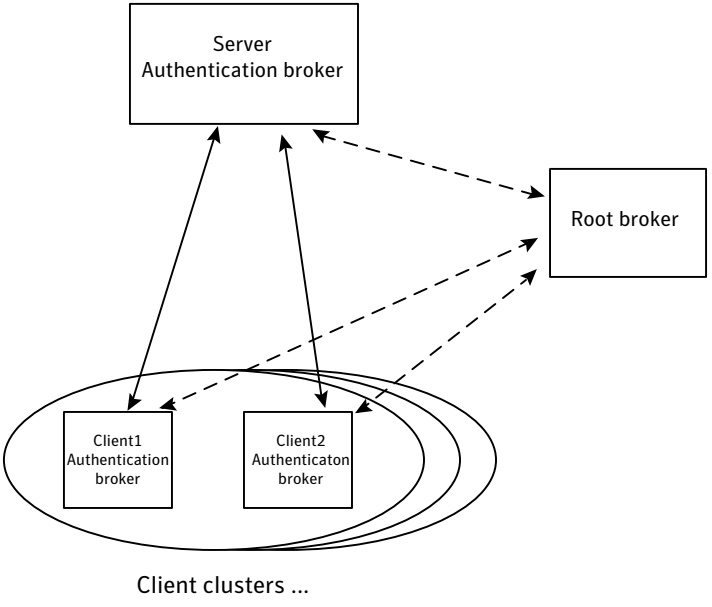
In a data center, TCP/IP communication between the SFCFS cluster and CP server must be made secure. The security of the communication channel involves encryption, authentication, and authorization.

The CP server node or cluster needs to confirm the authenticity of the SFCFS cluster nodes that communicate with it as a coordination point and only accept requests from known SFCFS cluster nodes. Requests from unknown clients are rejected as non-authenticated. Similarly, the fencing framework in SFCFS cluster must confirm that authentic users are conducting fencing operations with the CP server.

The encryption and authentication service for CP server is provided by Symantec™ Product Authentication Service. To enable Symantec™ Product Authentication Service, the VRTSat package is installed on the SFCFS clusters as well as CP server, as a part of VCS product installation.

Figure 2-2 displays a schematic of secure communication between the SFCFS cluster and CP server. An authentication broker is configured on CP server and each SFCFS cluster node which authenticates clients such as users or services, and grants them a product credential.

Figure 2-2 CP server and SFCFS clusters with authentication broker and root broker



Entities on behalf of which authentication is done, are referred to as principals. On the SFCFS cluster nodes, the current VCS installer creates the Authentication Server credentials on each node in the cluster, creates Web credentials for VCS users, and then sets up trust with the root broker. It also creates a VCS service group for the authentication broker. The installer then proceeds to start VCS in secure mode.

Typically, in an existing VCS cluster with security configured, a root broker would already have been configured and an authentication broker will be running on each cluster node.

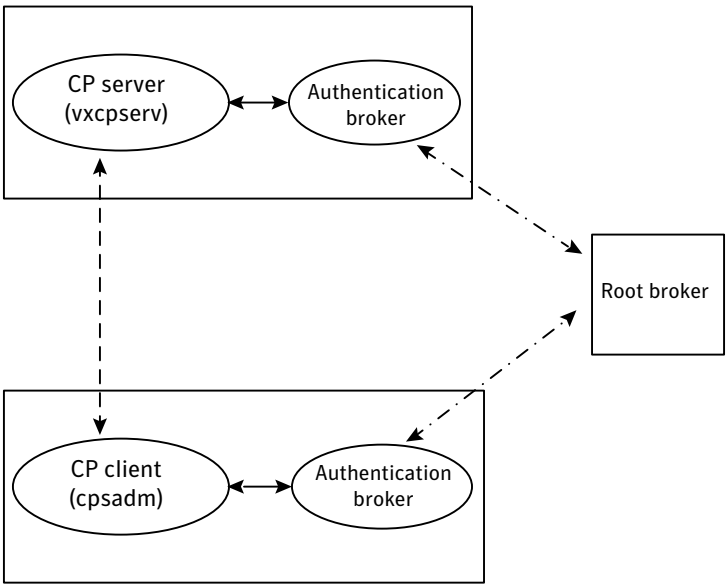
How secure communication between the CP servers and SFCFS clusters work

CP server and SFCFS cluster node communication involve the following entities:

- vxcpserv for the CP server
- cpsadm for the SFCFS cluster node

Figure 2-3 displays a schematic of the end-to-end communication flow with security enabled on CP server and SFCFS clusters.

Figure 2-3 End-To-end communication flow with security enabled on CP server and SFCFS clusters



Communication flow between CP server and SFCFS cluster nodes with security configured on them is as follows:

- Initial setup:
  Identities of authentication brokers configured on CP server, as well as SFCFS cluster nodes are configured in the root broker's authentication private domain repository.

  Note: If authentication brokers configured on CP server and SFCFS cluster nodes do not use the same root broker, then a trust should be established between the root brokers or authentication brokers, so that vxcpserv process can authenticate requests from theSFCFS cluster nodes.

  The `cpsadm` command gets the user name, domain type from the environment variables CPS_USERNAME, CPS_DOMAINTYPE. The user is expected to export these variables before running the cpsadm command manually. The customized fencing framework exports these environment variables internally before running the cpsadm commands.

The cp server process (vxcpserv) uses its own user (_CPS_SERVER_) which is added to the local authentication broker during server startup.

- Getting credentials from authentication broker:
  The cpsadm command tries to get the existing credentials from authentication broker running on the local node. If this fails, it tries to authenticate itself with the local authentication broker.
  The vxcpserv process tries to get the existing credentials from authentication broker running on the local node. If this fails, it tries to authenticate itself with the local authentication broker and creates a principal for itself .
- Communication between CP server and SFCFS cluster nodes:
  Once the CP server is up after establishing its credential, it becomes ready to receive data from the clients. Once authenticated with the local authentication broker, cpsadm connects to the CP server. Data is passed over to the CP server.
- Validation:
  On receiving data from a particular SFCFS cluster node, vxcpserv validates its credentials by consulting the local authentication broker. If validation fails, then the connection request data is rejected.

Security configuration details on CP server and SFCFS cluster

This section discusses security configuration details for the CP server and SFCFS cluster.

Settings in secure mode

If security is configured for communication between CP server and SFCFS cluster, the following are the settings:

- Settings on CP server:
  A user gets created in the local authentication broker during CP server startup with the following values:
  username: _CPS_SERVER_
  domainname: _CPS_SERVER_DOMAIN@*FQHN*
  domaintype: vx
  where, FQHN is Fully Qualified Host Name of the client node
  It can be verified using the following command on the CP server:

```
# /opt/VRTScps/bin/cpsat showcred
```

Note: The CP server configuration file (/etc/vxcps.conf) must not contain a line specifying **security=0**. If there is no line specifying "security" parameter or if there is a line specifying **security=1**, CP server with security is enabled (which is the default).

- Settings On SFCFS cluster node(s):
  On SFCFS cluster, a user gets created for each cluster node in the local authentication broker during VCS security configuration with the following values:
  username: _HA_VCS_hostname
  domainname: HA_SERVICES@*FQHN*
  domaintype: vx
  where, FQHN is Fully Qualified Host Name of the client node
  It can be verified using the following command on the SFCFS cluster node(s):

```
# /opt/VRTScps/bin/cpsat showcred
```

It is to be noted that the users mentioned above are used only for authentication for the communication between:

- CP server and authentication broker configured on it, and
- SFCFS cluster nodes and authentication brokers configured on them

For CP server's authorization, the following user gets created and used by customized fencing framework on the SFCFS cluster, if security is configured:

_HA_VCS_*hostname*@HA_SERVICES@*FQHN*

where, hostname is the client node name without qualification and FQHN is Fully Qualified Host Name of the client node.

For each SFCFS cluster node, this user must be registered on the CP server database before fencing starts on the SFCFS cluster node(s). This can be verified by issuing the following command:

```
# cpsadm -s cp_server -a list_users
```

The following is an example of the command output:

```
Username/Domain Type                             Cluster Name / UUID             Role
_HA_VCS_galaxy@HA_SERVICES@galaxy.symantec.com/vx  cluster1/ {f0735332-e3709c1c73b9} Operator
```

Note: The configuration file (/etc/vxfenmode) on each client node must not contain a line specifying **security=0**. If there is no line specifying "security" parameter or if there is a line specifying **security=1**, client node starts with security enabled (which is the default).

Settings in non-secure mode

In non-secure mode, only in-built authorization is provided on the CP server without asking for any passwords. No authentication and encryption are provided. User credentials of “cpsclient@hostname” of “vx” domaintype are used by the customized fencing framework for communication between CP server or SFCFS cluster node(s).

For each SFCFS cluster node, this user must be added on the CP server database before fencing starts on the SFCFS cluster node(s). The user can be verified by issuing the following command:

```
# cpsadm -s cpserver -a list_users
```

The following is an example of the command output:

```
Username/Domain Type  Cluster Name / UUID                 Role
cpsclient@galaxy/vx   cluster1 / {f0735332-e3709c1c73b9}  Operator
```

Note: In non-secure mode, CP server configuration file (/etc/vxcps.conf) should contain a line specifying **security=0**. Similarly, on each SFCFS cluster node the configuration file (/etc/vxfenmode) should contain a line specifying **security=0**.

About Veritas Volume Manager cluster functionality

Veritas Volume Manager cluster functionality (CVM) allows up to 32 nodes in a cluster to simultaneously access and manage a set of disks under VxVM control (VM disks). The same logical view of the disk configuration and any changes are available on each node. When the cluster functionality is enabled, all cluster nodes can share VxVM objects. Features provided by the base volume manager, such as mirroring, fast mirror resync and dirty region logging are also supported in the cluster environment.

To implement cluster functionality, VxVM works together with the cluster monitor daemon provided by the host operating system or by VCS. The cluster monitor informs VxVM of changes in cluster membership. Each node starts up independently and has its own cluster monitor, plus its own copies of the operating system and CVM. When a node joins a cluster it gains access to shared disks. When

a node leaves a cluster, it no longer has access to shared disks. A node joins a cluster when the cluster monitor is started on that node.

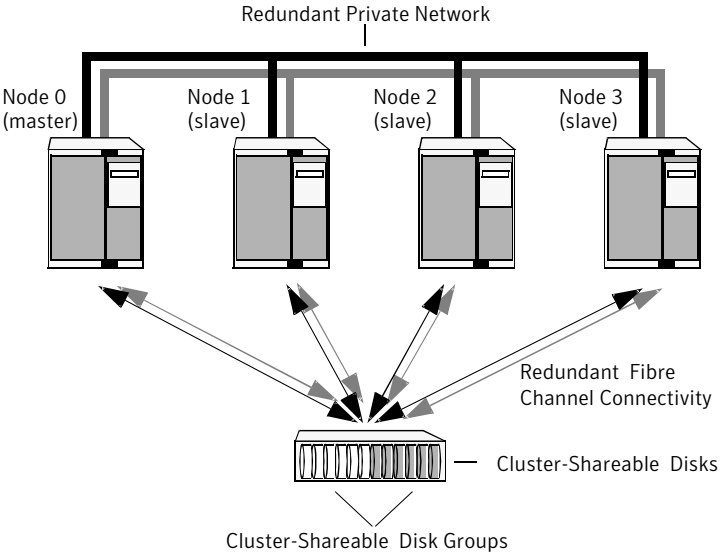
Note: RAID-5 volumes are not supported on a shared disk group.

Figure 2-4 illustrates a simple cluster arrangement consisting of four nodes with similar or identical hardware characteristics (CPUs, RAM and host adapters), and configured with identical software (including the operating system).

The nodes are fully connected by a private network and they are also separately connected to shared external storage (either disk arrays or JBODs: just a bunch of disks) via Fibre Channel. Each node has two independent paths to these disks, which are configured in one or more cluster-shareable disk groups.

The private network allows the nodes to share information about system resources and about each other's state. Using the private network, any node can recognize which nodes are currently active, which are joining or leaving the cluster, and which have failed. The private network requires at least two communication channels to provide redundancy against one of the channels failing. If only one channel were used, its failure would be indistinguishable from node failure—a condition known as network partitioning.

Figure 2-4 Example of a four node cluster



To the cluster monitor, all nodes are the same. VxVM objects configured within shared disk groups can potentially be accessed by all nodes that join the cluster.

However, the cluster functionality of VxVM requires one node to act as the master node; all other nodes in the cluster are slave nodes. Any node is capable of being the master node, which is responsible for coordinating certain VxVM activities.

Note: You must run commands that configure or reconfigure VxVM objects on the master node. Tasks that must be initiated from the master node include setting up shared disk groups and creating and reconfiguring volumes.

VxVM designates the first node to join a cluster the master node. If the master node leaves the cluster, one of the slave nodes is chosen to be the new master. In the preceding example, node 0 is the master node and nodes 1, 2 and 3 are slave nodes.

Shared disk groups overview

This section provides an overview of shared disk groups.

This section includes the following topics:

- Private and shared disk groups
- Activation modes of shared disk groups
- Connectivity policy of shared disk groups
- Limitations of shared disk groups

Private and shared disk groups

Table 2-4 describes the disk group types.

Table 2-4 Disk group types

Disk group	Description
Private	Belongs to only one node. A private disk group is only imported by one system. Disks in a private disk group may be physically accessible from one or more systems, but import is restricted to one system only. The root disk group is always a private disk group.
Shared	Is shared by all nodes. A shared (or cluster-shareable) disk group is imported by all cluster nodes. Disks in a shared disk group must be physically accessible from all systems that may join the cluster.

In a cluster, most disk groups are shared. Disks in a shared disk group are accessible from all nodes in a cluster, allowing applications on multiple cluster nodes to simultaneously access the same disk. A volume in a shared disk group

can be simultaneously accessed by more than one node in the cluster, subject to licensing and disk group activation mode restrictions.

You can use the `vxdg` command to designate a disk group as cluster-shareable. When a disk group is imported as cluster-shareable for one node, each disk header is marked with the cluster ID. As each node subsequently joins the cluster, it recognizes the disk group as being cluster-shareable and imports it. You can also import or deport a shared disk group at any time; the operation takes places in a distributed fashion on all nodes.

Each physical disk is marked with a unique disk ID. When cluster functionality for VxVM starts on the master, it imports all shared disk groups (except for any that have the `noautoimport` attribute set). When a slave tries to join a cluster, the master sends it a list of the disk IDs that it has imported, and the slave checks to see if it can access them all. If the slave cannot access one of the listed disks, it abandons its attempt to join the cluster. If it can access all of the listed disks, it imports the same shared disk groups as the master and joins the cluster. When a node leaves the cluster, it deports all its imported shared disk groups, but they remain imported on the surviving nodes.

Reconfiguring a shared disk group is performed with the co-operation of all nodes. Configuration changes to the disk group happen simultaneously on all nodes and the changes are identical. Such changes are atomic in nature, which means that they either occur simultaneously on all nodes or not at all.

Whether all members of the cluster have simultaneous read and write access to a cluster-shareable disk group depends on its activation mode setting.

The data contained in a cluster-shareable disk group is available as long as at least one node is active in the cluster. The failure of a cluster node does not affect access by the remaining active nodes. Regardless of which node accesses a cluster-shareable disk group, the configuration of the disk group looks the same.

Note: Applications running on each node can access the data on the VM disks simultaneously. VxVM does not protect against simultaneous writes to shared volumes by more than one node. It is assumed that applications control consistency (by using Veritas Storage Foundation Cluster File System or a distributed lock manager, for example).

Activation modes of shared disk groups

A shared disk group must be activated on a node in order for the volumes in the disk group to become accessible for application I/O from that node. The ability of applications to read from or to write to volumes is dictated by the activation mode

of a shared disk group. Valid activation modes for a shared disk group are `exclusivewrite`, `readonly`, `sharedread`, `sharedwrite`, and `off` (inactive).

Note: The default activation mode for shared disk groups is `sharedwrite`.

Special uses of clusters, such as high availability (HA) applications and off-host backup, can use disk group activation to explicitly control volume access from different nodes in the cluster.

Table 2-5 describes activation modes for shared disk groups.

Table 2-5 Activation modes for shared disk groups

Activation mode	Description
`exclusivewrite (ew)`	The node has exclusive write access to the disk group. No other node can activate the disk group for write access.
`readonly (ro)`	The node has read access to the disk group and denies write access for all other nodes in the cluster. The node has no write access to the disk group. Attempts to activate a disk group for either of the write modes on other nodes fail.
`sharedread (sr)`	The node has read access to the disk group. The node has no write access to the disk group, however other nodes can obtain write access.
`sharedwrite (sw)`	The node has write access to the disk group.
`off`	The node has neither read nor write access to the disk group. Query operations on the disk group are permitted.

Table 2-6 summarizes the allowed and conflicting activation modes for shared disk groups.

Table 2-6 Allowed and conflicting activation modes

Disk group activated in cluster as...	exclusive-write	readonly	sharedread	sharedwrite
exclusivewrite	Fails	Fails	Succeeds	Fails
readonly	Fails	Succeeds	Succeeds	Fails
sharedread	Succeeds	Succeeds	Succeeds	Succeeds
sharedwrite	Fails	Fails	Succeeds	Succeeds

To place activation modes under user control

- Create a `/etc/default/vxdg` file containing the following lines:

```
enable_activation=true
default_activation_mode=activation-mode
```

The *activation-mode* is one of `exclusivewrite, readonly, sharedread, sharedwrite`, or `off`.

When a shared disk group is created or imported, it is activated in the specified mode. When a node joins the cluster, all shared disk groups accessible from the node are activated in the specified mode.

The activation mode of a disk group controls volume I/O from different nodes in the cluster. It is not possible to activate a disk group on a given node if it is activated in a conflicting mode on another node in the cluster. When enabling activation using the defaults file, it is recommended that this file be made identical on all nodes in the cluster. Otherwise, the results of activation are unpredictable.

Note: If the `/etc/default/vxdg` file is edited while the `vxconfigd` daemon is already running, the `vxconfigd` process must be restarted for the changes in the defaults file to take effect.

If the default activation mode is anything other than `off`, an activation following a cluster join, or a disk group creation or import can fail if another node in the cluster has activated the disk group in a conflicting mode.

To display the activation mode for a shared disk group, use the `vxdg list` command.

You can also use the `vxdg` command to change the activation mode on a shared disk group.

See the *Veritas Volume Manager Administrator's Guide*.

Connectivity policy of shared disk groups

The nodes in a cluster must always agree on the status of a disk. In particular, if one node cannot write to a given disk, all nodes must stop accessing that disk before the results of the write operation are returned to the caller. Therefore, if a node cannot contact a disk, it should contact another node to check on the disk's status. If the disk fails, no node can access it and the nodes can agree to detach the disk. If the disk does not fail, but rather the access paths from some of the nodes fail, the nodes cannot agree on the status of the disk.

Table 2-7 describes the policies for resolving this type of discrepancy.

Table 2-7 Policies

Policy	Description
Global	The detach occurs cluster-wide (globally) if any node in the cluster reports a disk failure. This is the default policy.
Local	In the event of disks failing, the failures are confined to the particular nodes that saw the failure. However, this policy is not highly available because it fails the node even if one of the mirrors is available. Note that an attempt is made to communicate with all nodes in the cluster to ascertain the disks' usability. If all nodes report a problem with the disks, a cluster-wide detach occurs.

Limitations of shared disk groups

The cluster functionality of VxVM does not support RAID-5 volumes, or task monitoring for cluster-shareable disk groups. These features can, however, be used in private disk groups that are attached to specific nodes of a cluster. Online relayout is supported provided that it does not involve RAID-5 volumes.

The root disk group cannot be made cluster-shareable. It must be private.

Only raw device access may be performed via the cluster functionality of VxVM. It does not support shared access to file systems in shared volumes unless the appropriate software, such as Veritas Storage Foundation Cluster File System, is installed and configured.

If a shared disk group contains unsupported objects, deport it and then re-import the disk group as private on one of the cluster nodes. Reorganize the volumes into layouts that are supported for shared disk groups, and then deport and re-import the disk group as shared.

Storage Foundation Cluster File System and Veritas Volume Manager cluster functionality agents

Agents are VCS processes that manage predefined resource types. SFCFS and CVM require agents to interact with VCS. Agents bring resources online, take resources offline, monitor resources, and report any state changes to VCS. VCS bundled agents are part of VCS and are installed when VCS is installed. The SFCFS and CVM agents are add-on resources to VCS specifically for the Veritas File System and Veritas Volume Manager.

See "About agents for Storage Foundation Cluster File System" on page 129.

Veritas Volume Manager cluster functionality

The Veritas Volume Manager cluster functionality (CVM) makes logical volumes accessible throughout a cluster. CVM enables multiple hosts to concurrently access the logical volumes under its control. A VxVM cluster comprises nodes sharing a set of devices. The nodes are connected across a network. If one node fails, other nodes can access the devices. The VxVM cluster feature presents the same logical view of the device configurations, including changes, on all nodes. You configure CVM shared storage after VCS sets up a cluster configuration.

Chapter 3

Administering Storage Foundation Cluster File System and its components

This chapter includes the following topics:

About Storage Foundation Cluster File System administration

The Veritas Storage Foundation Cluster File System is a shared file system that enables multiple hosts to mount and perform file operations concurrently on the same file. To operate in a cluster configuration, SFCFS requires the integrated set of Veritas products included in the Veritas Storage Foundation Cluster File System.

To configure a cluster, SFCFS requires the Veritas Cluster Server (VCS). VCS supplies two major components integral to SFCFS. The LLT package provides node-to-node communications and monitors network communications. The GAB package provides cluster state, configuration, and membership service, and monitors the heartbeat links between systems to ensure that they are active. There

are several other packages supplied by VCS that provide application failover support when installing SFCFS HA.

See the *Veritas Storage Foundation Cluster File System Installation Guide*.

SFCFS also requires the cluster functionality (CVM) of the Veritas Volume Manager (VxVM) to create the shared volumes necessary for mounting cluster file systems.

For more information on these products, refer to the *Veritas Volume Manager* and *Veritas Cluster Server* documentation.

Veritas Enterprise Administrator Graphical User Interface

The Veritas Enterprise Administrator (VEA) console is no longer packaged with Storage Foundation products. Symantec recommends use of Storage Foundation Manager to manage, monitor and report on Storage Foundation product environments. You can download this utility at no charge at http://go.symantec.com/vom. If you wish to continue using VEA, a version is available for download from http://go.symantec.com/vom.

Administering CFS

This section describes some of the major aspects of cluster file system administration.

This section provides instructions for the following CFS administration tasks:

- Adding CFS file systems to VCS configuration
- Using cfsmount to mount CFS file systems
- Resizing CFS file systems
- Verifying the status of CFS file systems
- Verifying CFS port

If you encounter issues while administering CFS, refer to the troubleshooting section for assistance.

Adding CFS file systems to VCS configuration

To add a CFS file system to the VCS main.cf file without using an editor:

```
# cfsmntadm add oradatadg oradatavol \
/oradata1 cvm all=suid,rw

Mount Point is being added...
  /oradata1 added to the cluster-configuration
```

Using cfsmount to mount CFS file systems

To mount a CFS file system using cfsmount:

```
# cfsmount /oradata1
Mounting...
[/dev/vx/dsk/oradatadg/oradatavol]
mounted successfully at /oradata1 on system01
[/dev/vx/dsk/oradatadg/oradatavol]
mounted successfully at /oradata1 on system02
```

Resizing CFS file systems

If you see a message on the console indicating that a CFS file system is full, you may want to resize the file system. The `vxresize` command lets you resize a CFS file system. It extends the file system and the underlying volume.

See the `vxresize` (1M) manual page for information on various options.

The following command resizes an Oracle data CFS file system (the Oracle data volume is CFS mounted):

```
# vxresize -g oradatadg oradatavol +2G
```

Verifying the status of CFS file systems

Run the "cfscluster status" command to see the status of the nodes and their mount points:

```
# cfscluster status

  Node             :  system02
  Cluster Manager  :  not-running
  CVM state        :  not-running
  MOUNT POINT    SHARED VOLUME  DISK GROUP        STATUS

  /ocrvote       ocrvotevol     system01_ocr        NOT MOUNTED
  /oracle        ora_vol        system01_ora        NOT MOUNTED
  /crshome       ora_crs_vol    system01_crs        NOT MOUNTED
  /oradata1      ora_data1_vol  system01_data1      NOT MOUNTED
  /arch          archivol       system01_data1      NOT MOUNTED

  Node             :  system01
  Cluster Manager  :  running
  CVM state        :  running
  MOUNT POINT    SHARED VOLUME  DISK GROUP        STATUS
```

```
/ocrvote        ocrvotevol      system01_ocr        MOUNTED
/oracle         ora_vol         system01_ora        MOUNTED
/crshome        ora_crs_vol     system01_crs        MOUNTED
/oradata1       ora_data1_vol   system01_data1      MOUNTED
/arch           archivol        system01_data1      MOUNTED
```

Verifying CFS port

CFS uses port 'f' for communication between nodes. The CFS port state can be verified as follows:

```
# gabconfig -a | grep "Port f"
```

CFS agent log files

You can use the CFS agent log files that are located in the directory /var/VRTSvcs/log to debug CFS issues.

```
# cd /var/VRTSvcs/log
# ls
CFSMount_A.log
CFSfsckd_A.log
engine_A.log
```

The agent framework information is located in the engine_A.log file while the agent entry point information is located in the CFSMount_A.log and CFSfsckd_A.log files.

Storage Foundation Cluster File System commands

Table 3-1 describes the SFCFS commands.

Table 3-1 SFCFS commands

Commands	Description
cfscluster	Cluster configuration command
cfsmntadm	Adds, deletes, modifies, and sets policy on cluster mounted file systems
cfsdgadm	adds or deletes shared disk groups to and from a cluster configuration
cfsmount	mounts a cluster file system on a shared volume
cfsumount	unmounts a cluster file system on a shared volume

mount, fsclusteradm, and fsadm commands

The `mount` and `fsclustadm` commands are important for configuring cluster file systems.

mount

The `mount` command with the `-o cluster` option lets you access shared file systems.

See the `mount_vxfs`(1M) manual page.

fsclustadm

The `fsclustadm` command reports various attributes of a cluster file system. Using `fsclustadm` you can show and set the primary node in a cluster, translate node IDs to host names and vice versa, list all nodes that currently have a cluster mount of the specified file system mount point, and determine whether a mount is a local or cluster mount. The `fsclustadm` command operates from any node in a cluster on which the file system is mounted, and can control the location of the primary for a specified mount point.

See the `fsclustadm`(1M) manual page.

fsadm

The `fsadm` command can be invoked from the primary or secondary node.

See the `fsadm_vxfs`(1M) manual page.

Run commands safely in a cluster environment

Any UNIX command that can write to a raw device must be used carefully in a shared environment to prevent data from being corrupted. For shared VxVM volumes, SFCFS provides protection by reserving the volumes in a cluster to prevent VxFS commands, such as `fsck` and `mkfs`, from inadvertently damaging a mounted file system from another node in a cluster. However, commands such as `dd` execute without any reservation, and can damage a file system mounted from another node. Before running this kind of command on a file system, be sure the file system is not mounted on a cluster. You can run the `mount` command to see if a file system is a shared or local mount.

Time synchronization for Cluster File Systems

SFCFS requires that the system clocks on all nodes are synchronized using some external component such as the Network Time Protocol (NTP) daemon. If the nodes are not in sync, timestamps for creation (`ctime`) and modification (`mtime`) may not be consistent with the sequence in which operations actually happened.

Growing a Storage Foundation Cluster File System

There is a master node for CVM as well as a primary for SFCFS. When growing a file system, you grow the volume from the CVM master, and then grow the file system from any SFCFS node. The CVM master and the SFCFS node can be different nodes.

To determine the primary file system in a cluster

- To determine the primary file system in a cluster, type the following command:

```
# fsclustadm -v showprimary mount_point
```

To determine that the current node is the master CVM node

- To determine if the current node is the master CVM node, type the following comannd:

```
# vxdctl -c mode
```

To actually increase the size of the file system

1. On the master CVM node, type the following command:

```
# vxassist -g shared_disk_group growto volume_name newlength
```

2. On any SFCFS node, type the following command:

```
# fsadm -V vxfs -b newsize -r device_name mount_point
```

The /etc/filesystem file

In the `/etc/filesystems` file, do not specify any cluster file systems to mount-at-boot because mounts initiated from `filesystems` occur before cluster configuration begins. For cluster mounts, use the VCS configuration file to determine which file systems to enable following a reboot.

When the Storage Foundation Cluster File System primary fails

If the server on which the SFCFS primary is running fails, the remaining cluster nodes elect a new primary. The new primary reads the file system intent log and completes any metadata updates that were in process at the time of the failure. Application I/O from other nodes may block during this process and cause a delay. When the file system is again consistent, application processing resumes.

Because nodes using a cluster file system in secondary node do not update file system metadata directly, failure of a secondary node does not require metadata repair. SFCFS recovery from secondary node failure is therefore faster than from primary node failure.

Distribute the load on a cluster

Distributing the workload in a cluster provides performance and failover advantages.

For example, if you have eight file systems and four nodes, designating two file systems per node as the primary would be beneficial. Primaryship is determined by which node first mounts the file system. You can also use the `fsclustadm` to designate a SFCFS primary. The `fsclustadm setprimary` command can also define the order in which primaryship is assumed if the current primary fails. After setup, the policy is in effect as long as one or more nodes in the cluster have the file system mounted.

GUIs

You can use of Storage Foundation Manager to manage, monitor and report on Storage Foundation product environments. You can download this utility at no charge athttp://go.symantec.com/vom. If you wish to continue using VEA, a version is available for download from http://go.symantec.com/vom.

Storage Checkpoints on Storage Foundation Cluster File System

See the *Veritas Storage Foundation Advanced Features Administrator's Guide* for how to create and maintain Storage Checkpoints. The creation of Storage Checkpoints works the same on CFS file systems as they do on local mount file systems.

Snapshots on Storage Foundation Cluster File System

A snapshot provides a consistent point-in-time image of a VxFS file system. A snapshot can be accessed as a read-only mounted file system to perform efficient online backups of the file system. Snapshots implement copy-on-write semantics

that incrementally copy data blocks when they are overwritten on the snapped file system.

See the *Veritas Storage Foundation Advanced Features Administrator's Guide*.

Snapshots for cluster file systems extend the same copy-on-write mechanism for the I/O originating from any node in the cluster.

Cluster snapshot characteristics

A cluster snapshot has the following characteristics:

- A snapshot for a cluster mounted file system can be mounted on any node in a cluster. The file system can be a primary, secondary, or secondary-only. A stable image of the file system is provided for writes from any node.
- Multiple snapshots of a cluster file system can be mounted on the same or a different node in a cluster.
- A snapshot is accessible only on the node mounting a snapshot. The snapshot device cannot be mounted on two different nodes simultaneously.
- The device for mounting a snapshot can be a local disk or a shared volume. A shared volume is used exclusively by a snapshot mount and is not usable from other nodes in a cluster as long as the snapshot is active on that device.
- On the node mounting a snapshot, the snapped file system cannot be unmounted while the snapshot is mounted.
- A SFCFS snapshot ceases to exist if it is unmounted or the node mounting the snapshot fails. A snapshot, however, is not affected if any other node leaves or joins the cluster.
- A snapshot of a read-only mounted file system cannot be taken. It is possible to mount snapshot of a cluster file system only if the snapped cluster file system is mounted with the `crw` option.

Performance considerations

Mounting a snapshot file system for backup increases the load on the system because of the resources used to perform copy-on-writes and to read data blocks from the snapshot. In this situation, cluster snapshots can be used to do off-host backups. Off-host backups reduce the load of a backup application from the primary server. Overhead from remote snapshots is small when compared to overall snapshot overhead. Therefore, running a backup application by mounting a snapshot from a relatively less loaded node is beneficial to overall cluster performance.

Creating a snapshot on a Storage Foundation Cluster File System

To create and mount a snapshot on a two-node cluster using SFCFS administrative interface commands.

To create a snapshot on a cluster file system

1 To create a VxFS file system on a shared VxVM volume, type the following command:

```
# mkfs -V vxfs /dev/vx/rdsk/cfsdg/vol1

version 7 layout
104857600 sectors, 52428800 blocks of size 1024, log size
16384 blocks unlimited inodes, largefiles not supported
52428800 data blocks, 52399152 free data blocks 1600
allocation units of 32768 blocks, 32768 data blocks
```

2 To mount the file system on all nodes, type the following commands:

```
# cfsmntadm add cfsdg vol1 /mnt1 all=cluster
# cfsmount /mnt1
```

The `cfsmntadm` command adds an entry to the cluster manager configuration, then the `cfsmount` command mounts the file system on all nodes.

3 Add the snapshot on a previously created volume (`snapvol` in this example) to the cluster manager configuration. For example:

```
# cfsmntadm add snapshot cfsdg snapvol /mnt1 /mnt1snap system01=ro
```

The snapshot of a cluster file system is accessible only on the node where it is created; the snapshot file system itself cannot be cluster mounted.

4 Create and locally mount the snapshot file system on system01, type the following command:

```
# cfsmount /mnt1snap
```

5 A snapped file system cannot be unmounted until all of its snapshots are unmounted. Unmount and destroy the snapshot before trying to unmount the snapped cluster file system, type the following command:

```
# cfsumount /mnt1snap
```

Administering VCS

This section provides instructions for the following VCS administration tasks:

- Viewing available Veritas devices and drivers
- Verifying VCS configuration
- Starting and stopping VCS
- Rebooting nodes in a cluster

If you encounter issues while administering VCS, refer to the troubleshooting section for assistance.

Viewing available Veritas devices and drivers

To view the available Veritas devices:

```
# lsdev -C -c vxdrv

vxdmp     Available  Veritas VxDMP Device Driver
vxglm0    Available  N/A
vxgms0    Available  N/A
vxio      Available  Veritas VxIO Device Driver
vxportal0 Available  VERITAS VxPORTAL Device Driver
vxqio0    Defined    VERITAS VxQIO Device Driver
vxspec    Available  Veritas VxSPEC Device Driver
```

To view the devices that are loaded in memory, run the `genkex` command as shown in the following examples.

For example:

If you want to view whether or not the driver 'gab' is loaded in memory:

```
# genkex |grep gab

457f000    4ed20 /usr/lib/drivers/gab
```

If you want to view whether or not the 'vx' drivers are loaded in memory:

```
# genkex |grep vx

55de000     2000 /etc/vx/kernel/dmpjbod
55dc000     2000 /etc/vx/kernel/dmpap
55da000     2000 /etc/vx/kernel/dmpaa
55b5000    21000 /usr/lib/drivers/vxodm.ext_61
55b2000     3000 /usr/lib/drivers/vxspec
```

```
4de0000   7ce000 /usr/lib/drivers/vxio
4d7a000    62000 /usr/lib/drivers/vxdmp
4cee000    3d000 /usr/lib/drivers/vxgms.ext_61
4c7e000    4d000 /usr/lib/drivers/vxfen
4bf3000     e000 /usr/lib/drivers/vxqio.ext_61
4bef000     3000 /usr/lib/drivers/vxportal.ext_61
4969000   232000 /usr/lib/drivers/vxfs.ext_61
48fa000    3e000 /usr/lib/drivers/vxglm.ext
```

Configuring VCS to start Oracle with a specified Pfile

If you want to configure VCS such that Oracle starts with a specified Pfile, modify the main.cf file for the Oracle group as follows:

```
Oracle ora1 (
                Sid @system01 = vrts1
                Sid @system02 = vrts2
                Owner = oracle
                Home = "/app/oracle/orahome"
                StartUpOpt = SRVCTLSTART
                ShutDownOpt = SRVCTLSTOP
                pfile="/app/oracle/orahome/dbs/initprod1.ora"
                )
```

Verifying VCS configuration

To verify the VCS configuration:

```
# cd /etc/VRTSvcs/conf/config
# hacf -verify .
```

Starting and stopping VCS

To start VCS on each node:

```
# hastart
```

To stop VCS on each node:

```
# hastop -local
```

You can also use the command `hastop -all`; however, make sure that you wait for port 'h' to close before restarting VCS.

Rebooting nodes in a cluster

If you need to restart nodes in your cluster, use the `shutdown` command with the -r option. The `shutdown -r` command cleanly shuts down VCS and takes the service groups offline.

To restart nodes:

```
# shutdown -r
```

Do not use the `reboot` command. The `reboot` command causes issues with I/O fencing and Oracle CRS.

Administering CVM

Listing all the CVM shared disks

You can use the following command to list all the CVM shared disks:

```
# vxdisk -o alldgs list |grep shared
```

Establishing CVM cluster membership manually

In most cases you do not have to start CVM manually; it normally starts when VCS is started.

Run the following command to start CVM manually:

```
# vxclustadm -m vcs -t gab startnode

vxclustadm: initialization completed
```

Note that `vxclustadm` reads main.cf for cluster configuration information and is therefore not dependent upon VCS to be running. You do not need to run the `vxclustadm startnode` command as normally the `hastart` (VCS start) command starts CVM automatically.

To verify whether CVM is started properly:

```
# vxclustadm nidmap
Name          CVM Nid     CM Nid      State
system01        0           0         Joined: Master
system02        1           1         Joined: Slave
```

Manually importing a shared disk group

You can use the following command to manually import a shared disk group:

```
# vxdg -s import dg_name
```

Manually deporting a shared disk group

You can use the following command to manually deport a shared disk group:

```
# vxdg deport dg_name
```

Note that the deport of a shared disk group removes the SCSI-3 PGR keys on the disks. It also removes the 'shared' flag on the disks.

Manually starting shared volumes

Following a manual CVM shared disk group import, the volumes in the disk group need to be started manually, as follows:

```
# vxvol -g dg_name startall
```

To verify that the volumes are started, run the following command:

```
# vxprint -htrg dg_name | grep ^v
```

Evaluating the state of CVM ports

CVM kernel (vxio driver) uses port 'v' for kernel messaging and port 'w' for vxconfigd communication between the cluster nodes. The following command displays the state of CVM ports:

```
# gabconfig -a | egrep "Port [vw]"
```

Verifying if CVM is running in an SFCFS cluster

You can use the following options to verify whether CVM is up or not in an SFCFS cluster.

The following output is displayed on a node that is not a member of the cluster:

```
# vxdctl -c mode
mode: enabled: cluster inactive
# vxclustadm -v nodestate
state: out of cluster
```

On the master node, the following output is displayed:

```
# vxdctl -c mode

mode: enabled: cluster active - MASTER
master: system01
```

On the slave nodes, the following output is displayed:

```
# vxdctl -c mode

mode: enabled: cluster active - SLAVE
master: system02
```

The following command lets you view all the CVM nodes at the same time:

```
# vxclustadm nidmap

Name      CVM Nid    CM Nid     State
system01         0          0     Joined: Master
system02         1          1     Joined: Slave
```

Verifying CVM membership state

The state of CVM can be verified as follows:

```
# vxclustadm -v nodestate
state: joining
        nodeId=0
        masterId=0
        neighborId=1
        members=0x3
        joiners=0x0
        leavers=0x0
        reconfig_seqnum=0x72a10b
        vxfen=on
```

The state indicates that CVM has completed its kernel level join and is in the middle of vxconfigd level join.

The `vxdctl -c mode` command indicates whether a node is a CVM master or CVM slave.

Verifying the state of CVM shared disk groups

You can use the following command to list the shared disk groups currently imported in the SFCFS cluster:

```
# vxdg list |grep shared
```

```
oradatadg enabled,shared 1052685125.1485.system01
```

Verifying the activation mode

In an SFCFS cluster, the activation of shared disk group should be set to “shared-write” on each of the cluster nodes.

To verify whether the “shared-write”activation is set:

```
# vxdg list diskgroupname |grep activation

local-activation: shared-write
```

If "shared-write" activation is not set, run the following command:

```
# vxdg -g diskgroupname set activation=sw
```

CVM log files

The /var/VRTSvcs/log directory contains the agent log files.

```
# cd /var/VRTSvcs/log
# ls -l *CVM* engine_A.log
CVMCluster_A.log     # CVM Agent log
CVMVolDg_A.log       # CVM VolDg Agent log
CVMVxconfigd_A.log   # CVM vxconfigd Agent log
engine_A.log         # VCS log
```

You can use the cmdlog file to view the list of CVM commands that have been executed. The file is located at `/var/adm/vx/cmdlog`.

See the *Veritas Volume Manager Administrator's Guide* for more information.

Administering ODM

This section provides instructions for the following ODM administration tasks:

- Verifying the ODM port
- Starting ODM
- Stopping ODM

If you encounter issues while administering ODM, refer to the troubleshooting section for assistance.

Verifying the ODM port

It is recommended to enable ODM in SFCFS. Run the following command to verify that ODM is running:

```
# gabconfig -a | grep "Port d"
```

Starting ODM

The following procedure provides instructions for starting ODM.

To start ODM

- `# /etc/rc.d/rc2.d/S99odm start`

Stopping ODM

The following procedure provides instructions for stopping ODM.

To stop ODM

1. On all nodes, issue the following command:

   ```
   # hastop -local
   ```

2. Next, issue the following command:

   ```
   # /etc/rc.d/rc2.d/S99odm stop
   ```

Note: The administrator does not usually need to stop or start ODM. Normally, ODM is stopped during `shutdown -r`. ODM is started while rebooting, going to multi-user mode.

Administering I/O Fencing

See the *Veritas Cluster Server Administrator's Guide* for more information.

About administering I/O fencing

The I/O fencing feature provides the following utilities that are available through the VRTSvxfen package:

vxfentsthdw	Tests hardware for I/O fencing See "About vxfentsthdw utility" on page 71.
vxfenconfig	Configures and unconfigures I/O fencing Checks the list of coordinator disks used by the vxfen driver.
vxfenadm	Displays information on I/O fencing operations and manages SCSI-3 disk registrations and reservations for I/O fencing See "About vxfenadm utility" on page 79.
vxfenclearpre	Removes SCSI-3 registrations and reservations from disks See "About vxfenclearpre utility" on page 84.
vxfenswap	Replaces coordinator disks without stopping I/O fencing See "About vxfenswap utility" on page 86.
vxfendisk	Generates the list of paths of disks in the diskgroup. This utility requires that Veritas Volume Manager is installed and configured.

The I/O fencing commands reside in the /opt/VRTS/bin folder. Make sure you added this folder path to the PATH environment variable.

Refer to the corresponding manual page for more information on the commands.

About vxfentsthdw utility

You can use the vxfentsthdw utility to verify that shared storage arrays to be used for data support SCSI-3 persistent reservations and I/O fencing. During the I/O fencing configuration, the testing utility is used to test a single disk. The utility has other options that may be more suitable for testing storage devices in other configurations. You also need to test coordinator disk groups.

See *Storage Foundation Cluster File System Installation Guide* to set up I/O fencing.

The utility, which you can run from one system in the cluster, tests the storage used for data by setting and verifying SCSI-3 registrations on the disk or disks you specify, setting and verifying persistent reservations on the disks, writing data to the disks and reading it, and removing the registrations from the disks.

Refer also to the `vxfentsthdw`(1M) manual page.

About general guidelines for using vxfentsthdw utility

Review the following guidelines to use the vxfentsthdw utility:

- The utility requires two systems connected to the shared storage.

  Caution: The tests overwrite and destroy data on the disks, unless you use the `-r` option.

- The two nodes must have ssh (default) or rsh communication. If you use rsh, launch the vxfentsthdw utility with the `-n` option.
  After completing the testing process, you can remove permissions for communication and restore public network connections.
- To ensure both systems are connected to the same disk during the testing, you can use the `vxfenadm -i diskpath` command to verify a disk's serial number.
  See "Verifying that the nodes see the same disk" on page 83.
- For disk arrays with many disks, use the `-m` option to sample a few disks before creating a disk group and using the `-g` option to test them all.
- The utility indicates a disk can be used for I/O fencing with a message resembling:

  ```
  The disk /dev/hdisk75 is ready to be configured for
  I/O Fencing on node system01
  ```

  If the utility does not show a message stating a disk is ready, verification has failed.
- If the disk you intend to test has existing SCSI-3 registration keys, the test issues a warning before proceeding.

About the vxfentsthdw command options

Table 3-2 describes the methods that the utility provides to test storage devices.

Table 3-2 vxfentsthdw options

vxfentsthdw option	Description	When to use
-n	Utility uses rsh for communication.	Use when rsh is used for communication.

Table 3-2 vxfentsthdw options *(continued)*

vxfentsthdw option	Description	When to use
-r	Non-destructive testing. Testing of the disks for SCSI-3 persistent reservations occurs in a non-destructive way; that is, there is only testing for reads, not writes. May be used with -m, -f, or -g options.	Use during non-destructive testing. See "Performing non-destructive testing on the disks using the -r option" on page 76.
-t	Testing of the return value of SCSI TEST UNIT (TUR) command under SCSI-3 reservations. A warning is printed on failure of TUR testing.	When you want to perform TUR testing.
-d	Use DMP devices. May be used with -c or -g options.	By default, the script picks up the DMP paths for disks in the disk group. If you want the script to use the raw paths for disks in the disk group, use the -w option.
-w	Use raw devices. May be used with -c or -g options.	With the -w option, the script picks the operating system paths for disks in the disk group. By default, the script uses the -d option to pick up the DMP paths for disks in the disk group.
-c	Utility tests the coordinator disk group prompting for systems and devices, and reporting success or failure.	For testing disks in coordinator disk group. See "Testing the coordinator disk group using vxfentsthdw -c option" on page 74.
-m	Utility runs manually, in interactive mode, prompting for systems and devices, and reporting success or failure. May be used with -r and -t options. -m is the default option.	For testing a few disks or for sampling disks in larger arrays. See "Testing the shared disks using the vxfentsthdw -m option" on page 76.

Table 3-2 vxfentsthdw options *(continued)*

vxfentsthdw option	Description	When to use
-f *filename*	Utility tests system/device combinations listed in a text file. May be used with -r and -t options.	For testing several disks. See "Testing the shared disks listed in a file using the vxfentsthdw -f option" on page 78.
-g *disk_group*	Utility tests all disk devices in a specified disk group. May be used with -r and -t options.	For testing many disks and arrays of disks. Disk groups may be temporarily created for testing purposes and destroyed (ungrouped) after testing. See "Testing all the disks in a disk group using the vxfentsthdw -g option" on page 78.

Testing the coordinator disk group using vxfentsthdw -c option

Use the vxfentsthdw utility to verify disks are configured to support I/O fencing. In this procedure, the vxfentsthdw utility tests the three disks one disk at a time from each node.

The procedure in this section uses the following disks for example:

- From the node system01, the disks are /dev/rhdisk75, /dev/rhdisk76, and /dev/rhdisk77.
- From the node system02, the same disks are /dev/rhdisk80, /dev/rhdisk81, and /dev/rhdisk82.

Note: To test the coordinator disk group using the vxfentsthdw utility, the utility requires that the coordinator disk group, vxfencoorddg, be accessible from two nodes.

To test the coordinator disk group using vxfentsthdw -c

1 Use the vxfentsthdw command with the -c option. For example:

```
# vxfentsthdw -c vxfencoorddg
```

2 Enter the nodes you are using to test the coordinator disks:

```
Enter the first node of the cluster: system01
Enter the second node of the cluster: system02
```

3 Review the output of the testing process for both nodes for all disks in the coordinator disk group. Each disk should display output that resembles:

```
ALL tests on the disk /dev/rhdisk75 have PASSED.
The disk is now ready to be configured for I/O Fencing on node
system01 as a COORDINATOR DISK.

ALL tests on the disk /dev/rhdisk80 have PASSED.
The disk is now ready to be configured for I/O Fencing on node
system02 as a COORDINATOR DISK.
```

4 After you test all disks in the disk group, the vxfencoorddg disk group is ready for use.

Removing and replacing a failed disk

If a disk in the coordinator disk group fails verification, remove the failed disk or LUN from the vxfencoorddg disk group, replace it with another, and retest the disk group.

To remove and replace a failed disk

1 Use the vxdiskadm utility to remove the failed disk from the disk group.

Refer to the *Veritas Volume Manager Administrator's Guide*.

2 Add a new disk to the node, initialize it, and add it to the coordinator disk group.

See the *Storage Foundation Cluster File System Installation Guide* for instructions to initialize disks for I/O fencing and to set up coordinator disk groups.

If necessary, start the disk group.

See the *Veritas Volume Manager Administrator's Guide* for instructions to start the disk group.

3 Retest the disk group.

See "Testing the coordinator disk group using vxfentsthdw -c option" on page 74.

Performing non-destructive testing on the disks using the -r option

You can perform non-destructive testing on the disk devices when you want to preserve the data.

To perform non-destructive testing on disks

- To test disk devices containing data you want to preserve, you can use the -r option with the -m, -f, or -g options.

  For example, to use the -m option and the -r option, you can run the utility as follows:

  ```
  # vxfentsthdw -rm
  ```

  When invoked with the -r option, the utility does not use tests that write to the disks. Therefore, it does not test the disks for all of the usual conditions of use.

Testing the shared disks using the vxfentsthdw -m option

Review the procedure to test the shared disks. By default, the utility uses the -m option.

This procedure uses the /dev/hdisk75 disk in the steps.

If the utility does not show a message stating a disk is ready, verification has failed. Failure of verification can be the result of an improperly configured disk array. It can also be caused by a bad disk.

If the failure is due to a bad disk, remove and replace it. The vxfentsthdw utility indicates a disk can be used for I/O fencing with a message resembling:

```
The disk /dev/hdisk75 is ready to be configured for
I/O Fencing on node system01
```

Note: For A/P arrays, run the `vxfentsthdw` command only on secondary paths.

To test disks using vxfentsthdw script

1. Make sure system-to-system communication is functioning properly.
2. From one node, start the utility.

   ```
   # vxfentsthdw [-n]
   ```

3. After reviewing the overview and warning that the tests overwrite data on the disks, confirm to continue the process and enter the node names.

   ```
   ******** WARNING!!!!!!!! ********
   THIS UTILITY WILL DESTROY THE DATA ON THE DISK!!

   Do you still want to continue : [y/n] (default: n) y
   Enter the first node of the cluster: system01
   Enter the second node of the cluster: system02
   ```

4. Enter the names of the disks you are checking. For each node, the disk may be known by the same name:

   ```
   Enter the disk name to be checked for SCSI-3 PGR on node
   system01 in the format: /dev/rhdiskx
   /dev/rhdisk75
   Enter the disk name to be checked for SCSI-3 PGR on node
   system02 in the format: /dev/rhdiskx
   Make sure it's the same disk as seen by nodes system01 and system02
   /dev/rhdisk75
   ```

   If the serial numbers of the disks are not identical, then the test terminates.

5. Review the output as the utility performs the checks and report its activities.

6 If a disk is ready for I/O fencing on each node, the utility reports success:

```
ALL tests on the disk /dev/hdisk75 have PASSED
The disk is now ready to be configured for I/O Fencing on node
system01
...
Removing test keys and temporary files, if any ...
.
.
```

7 Run the vxfentsthdw utility for each disk you intend to verify.

Testing the shared disks listed in a file using the vxfentsthdw -f option

Use the -f option to test disks that are listed in a text file. Review the following example procedure.

To test the shared disks listed in a file

1 Create a text file disks_test to test two disks shared by systems system01 and system02 that might resemble:

```
system01 /dev/rhdisk75 system02 /dev/rhdisk77
system01 /dev/rhdisk76 system02 /dev/rhdisk78
```

Where the first disk is listed in the first line and is seen by system01 as /dev/rhdisk75 and by system02 as /dev/rhdisk77. The other disk, in the second line, is seen as /dev/rhdisk76 from system01 and /dev/rhdisk78 from system02. Typically, the list of disks could be extensive.

2 To test the disks, enter the following command:

```
# vxfentsthdw -f disks_test
```

The utility reports the test results one disk at a time, just as for the -m option.

Testing all the disks in a disk group using the vxfentsthdw -g option

Use the -g option to test all disks within a disk group. For example, you create a temporary disk group consisting of all disks in a disk array and test the group.

Note: Do not import the test disk group as shared; that is, do not use the -s option.

After testing, destroy the disk group and put the disks into disk groups as you need.

To test all the disks in a diskgroup

1. Create a diskgroup for the disks that you want to test.
2. Enter the following command to test the diskgroup test_disks_dg:

   ```
   # vxfentsthdw -g test_disks_dg
   ```

   The utility reports the test results one disk at a time.

Testing a disk with existing keys

If the utility detects that a coordinator disk has existing keys, you see a message that resembles:

```
There are Veritas I/O fencing keys on the disk. Please make sure
that I/O fencing is shut down on all nodes of the cluster before
continuing.

         ******** WARNING!!!!!!!! ********

THIS SCRIPT CAN ONLY BE USED IF THERE ARE NO OTHER ACTIVE NODES
IN THE CLUSTER!  VERIFY ALL OTHER NODES ARE POWERED OFF OR
INCAPABLE OF ACCESSING SHARED STORAGE.

If this is not the case, data corruption will result.

Do you still want to continue : [y/n] (default: n) y
```

The utility prompts you with a warning before proceeding. You may continue as long as I/O fencing is not yet configured.

About vxfenadm utility

Administrators can use the vxfenadm command to troubleshoot and test fencing configurations.

The command's options for use by administrators are as follows:

-s	read the keys on a disk and display the keys in numeric, character, and node format **Note:** The -g and -G options are deprecated. Use the -s option.
-i	read SCSI inquiry information from device

-m	register with disks
-n	make a reservation with disks
-p	remove registrations made by other systems
-r	read reservations
-x	remove registrations

Refer to the `vxfenadm`(1m) manual page for a complete list of the command options.

About the I/O fencing registration key format

The keys that the vxfen driver registers on the data disks and the coordinator disks consists of eight bytes. The key format is different for the coordinator disks and data disks.

The key format of the coordinator disks is as follows:

Byte	0	1	2	3	4	5	6	7
Value	V	F	cID 0x	cID 0x	cID 0x	cID 0x	nID 0x	nID 0x

where:

- VF is the unique identifier that carves out a namespace for the keys (consumes two bytes)
- cID 0x is the LLT cluster ID in hexadecimal (consumes four bytes)
- nID 0x is the LLT node ID in hexadecimal (consumes two bytes)

The vxfen driver uses this key format in both scsi3 mode and customized mode of I/O fencing.

The key format of the data disks that are configured as failover disk groups under VCS is as follows:

Byte	0	1	2	3	4	5	6	7
Value	A+nID	V	C	S				

where nID is the LLT node ID

For example: If the node ID is 1, then the first byte has the value as B ('A' + 1 = B).

The key format of the data disks configured as parallel disk groups under CVM is as follows:

Byte	0	1	2	3	4	5	6	7
Value	A+nID	P	G	R	DGcount	DGcount	DGcount	DGcount

where DGcount is the count of disk group in the configuration

Displaying the I/O fencing registration keys

You can display the keys that are currently assigned to the disks using the vxfenadm command.

To display the I/O fencing registration keys

1 To display the key for the disks, run the following command:

```
# vxfenadm -s disk_name
```

For example:

- To display the key for the coordinator disk /dev/hdisk75 from the system with node ID 1, enter the following command:

```
# vxfenadm -s /dev/hdisk75
key[1]:
  [Numeric Format]:  86,70,68,69,69,68,48,48
  [Character Format]: VFDEED00
* [Node Format]: Cluster ID: 57069 Node ID: 0 Node Name: system01
```

The -s option of vxfenadm displays all eight bytes of a key value in three formats. In the numeric format,

- The first two bytes, represent the identifier VF, contains the ASCII value 86, 70.
- The next four bytes contain the ASCII value of the cluster ID 57069 encoded in hex (0xDEED) which are 68, 69, 69, 68.
- The remaining bytes contain the ASCII value of the node ID 0 (0x00) which are 48, 48. Node ID 1 would be 01 and node ID 10 would be 0A.

An asterisk before the Node Format indicates that the vxfenadm command is run from the node of a cluster where LLT is configured and running.

- To display the keys on a CVM parallel disk group:

```
# vxfenadm -s /dev/vx/rdmp/disk_7

Reading SCSI Registration Keys...

Device Name: /dev/vx/rdmp/disk_7
```

```
Total Number Of Keys: 1
key[0]:
  [Numeric Format]:  66,80,71,82,48,48,48,48
  [Character Format]: BPGR0001
  [Node Format]: Cluster ID: unknown  Node ID: 1 Node Name: system02
```

- To display the keys on a VCS failover disk group:

```
# vxfenadm -s /dev/vx/rdmp/disk_8

Reading SCSI Registration Keys...

Device Name: /dev/vx/rdmp/disk_8
Total Number Of Keys: 1
key[0]:
  [Numeric Format]:  65,86,67,83,0,0,0,0
  [Character Format]: AVCS
  [Node Format]: Cluster ID: unknown  Node ID: 0 Node Name: system01
```

2 To display the keys that are registered in all the disks specified in a disk file:

```
# vxfenadm -s all -f disk_filename
```

For example:

To display all the keys on coordinator disks:

```
# vxfenadm -s all -f /etc/vxfentab

Device Name: /dev/vx/rdmp/disk_9
Total Number Of Keys: 2
key[0]:
  [Numeric Format]:  86,70,66,69,65,68,48,50
  [Character Format]: VFBEAD02
  [Node Format]: Cluster ID: 48813 Node ID: 2  Node Name: unknown
key[1]:
  [Numeric Format]:  86,70,68,69,69,68,48,48
  [Character Format]: VFDEED00
* [Node Format]: Cluster ID: 57069 Node ID: 0  Node Name: system01
```

You can verify the cluster ID using the lltstat -C command, and the node ID using the lltstat -N command. For example:

```
# lltstat -C
57069
```

If the disk has keys which do not belong to a specific cluster, then the vxfenadm command cannot look up the node name for the node ID and hence prints the node name as unknown. For example:

```
Device Name: /dev/vx/rdmp/disk_7
Total Number Of Keys: 1
key[0]:
  [Numeric Format]:  86,70,45,45,45,45,48,49
  [Character Format]: VF----01
  [Node Format]: Cluster ID: unknown  Node ID: 1  Node Name: system02
```

For disks with arbitrary format of keys, the vxfenadm command prints all the fields as unknown. For example:

```
  [Numeric Format]:  65,66,67,68,49,50,51,45
  [Character Format]: ABCD123-
  [Node Format]: Cluster ID: unknown Node ID: unknown
         Node Name: unknown
```

Verifying that the nodes see the same disk

To confirm whether a disk (or LUN) supports SCSI-3 persistent reservations, two nodes must simultaneously have access to the same disks. Because a shared disk is likely to have a different name on each node, check the serial number to verify the identity of the disk. Use the vxfenadm command with the -i option to verify that the same serial number for the LUN is returned on all paths to the LUN.

For example, an EMC disk is accessible by the /dev/rhdisk75 path on node A and the /dev/rhdisk76 path on node B.

To verify that the nodes see the same disks

1. Verify the connection of the shared storage for data to two of the nodes on which you installed SFCFS.
2. From node A, enter the following command:

```
# vxfenadm -i /dev/rhdisk75

Vendor id       : EMC
Product id      : SYMMETRIX
Revision        : 5567
Serial Number   : 42031000a
```

The same serial number information should appear when you enter the equivalent command on node B using the/dev/rhdisk76 path.

On a disk from another manufacturer, Hitachi Data Systems, the output is different and may resemble:

```
# vxfenadm -i /dev/hdisk76

Vendor id       : HITACHI
Product id      : OPEN-3
Revision        : 0117
Serial Number   : 0401EB6F0002
```

Refer to the `vxfenadm`(1M) manual page for more information.

About vxfenclearpre utility

You can use the vxfenclearpre utility to remove SCSI-3 registrations and reservations on the disks.

See "Removing preexisting keys" on page 84.

This utility currently does not support server-based fencing. You must manually resolve any preexisting split-brain with server-based fencing configuration.

See "Issues during server-based fencing start up on SFCFS cluster node" on page 180.

Removing preexisting keys

If you encountered a split-brain condition, use the vxfenclearpre utility to remove SCSI-3 registrations and reservations on the coordinator disks as well as on the data disks in all shared disk groups.

You can also use this procedure to remove the registration and reservation keys created by another node from a disk.

To clear keys after split-brain

1 Stop VCS on all nodes.

```
# hastop -all
```

2 Make sure that the port h is closed on all the nodes. Run the following command on each node to verify that the port h is closed:

```
# gabconfig -a
```

Port h must not appear in the output.

3 Stop I/O fencing on all nodes. Enter the following command on each node:

```
# /etc/init.d/vxfen.rc stop
```

4 If you have any applications that run outside of VCS control that have access to the shared storage, then shut down all other nodes in the cluster that have access to the shared storage. This prevents data corruption.

5 Start the vxfenclearpre script:

6 Read the script's introduction and warning. Then, you can choose to let the script run.

```
Do you still want to continue: [y/n] (default : n) y
```

The script cleans up the disks and displays the following status messages.

```
Cleaning up the coordinator disks...

Cleaning up the data disks for all shared disk groups...

Successfully removed SCSI-3 persistent registration and
reservations from the coordinator disks as well as the
shared data disks.

Reboot the server to proceed with normal cluster startup...
#
```

7 Restart all nodes in the cluster.

About vxfenswap utility

The vxfenswap utility allows you to replace coordinator disks in a cluster that is online. The utility verifies that the serial number of the new disks are identical on all the nodes and the new disks can support I/O fencing.

This utility also supports server-based fencing.

Refer to the `vxfenswap`(1M) manual page.

See the *Storage Foundation Cluster File System Installation Guide* for details on the coordinator disk requirements.

You can replace the coordinator disks without stopping I/O fencing in the following cases:

- The disk becomes defective or inoperable and you want to switch to a new diskgroup.
  See "Replacing I/O fencing coordinator disks when the cluster is online" on page 87.
  See "Replacing the coordinator diskgroup in a cluster that is online" on page 89.
  If you want to replace the coordinator disks when the cluster is offline, you cannot use the vxfenswap utility. You must manually perform the steps that the utility does to replace the coordinator disks.
  See "Replacing defective disks when the cluster is offline" on page 177.
- You want to switch the disk interface between raw devices and DMP devices.
  See "Changing the disk interaction policy in a cluster that is online" on page 92.
- The keys that are registered on the coordinator disks are lost.
  In such a case, the cluster might panic when a split-brain occurs. You can replace the coordinator disks with the same disks using the vxfenswap command. During the disk replacement, the missing keys register again without any risk of data corruption.
  See "Refreshing lost keys on coordinator disks" on page 95.

In server-based fencing configuration, you can use the vxfenswap utility to perform the following tasks:

- Perform a planned replacement of customized coordination points (CP servers or SCSI-3 disks).
  See " Replacing the coordination points for an online cluster" on page 101.
- Refresh the I/O fencing keys that are registered on the coordination points.
  See "Refreshing registration keys on the coordination points for server-based fencing" on page 104.

If the vxfenswap operation is unsuccessful, then you can use the `-a cancel` of the `vxfenswap` command to manually roll back the changes that the vxfenswap utility does.

- For disk-based fencing, use the `vxfenswap -g diskgroup -a cancel` command to cancel the vxfenswap operation.
  You must run this command if a node fails during the process of disk replacement, or if you aborted the disk replacement.
- For server-based fencing, use the `vxfenswap -a cancel` command to cancel the vxfenswap operation.

Replacing I/O fencing coordinator disks when the cluster is online

Review the procedures to add, remove, or replace one or more coordinator disks in a cluster that is operational.

Warning: The cluster might panic if any node leaves the cluster membership before the vxfenswap script replaces the set of coordinator disks.

To replace a disk in a coordinator diskgroup when the cluster is online

1. Make sure system-to-system communication is functioning properly.
2. Make sure that the cluster is online.

```
# vxfenadm -d

I/O Fencing Cluster Information:
================================
Fencing Protocol Version: 201
Fencing Mode: SCSI3
Fencing SCSI3 Disk Policy: dmp
Cluster Members:
  * 0 (system01)
  1 (system02)
RFSM State Information:
  node 0 in state 8 (running)
  node 1 in state 8 (running)
```

3 Import the coordinator disk group.

The file /etc/vxfendg includes the name of the disk group (typically, vxfencoorddg) that contains the coordinator disks, so use the command:

```
# vxdg -tfC import `cat /etc/vxfendg`
```

where:

-t specifies that the disk group is imported only until the node restarts.

-f specifies that the import is to be done forcibly, which is necessary if one or more disks is not accessible.

-C specifies that any import locks are removed.

4 Turn off the coordinator attribute value for the coordinator disk group.

```
# vxdg -g vxfencoorddg set coordinator=off
```

5 To remove disks from the coordinator disk group, use the VxVM disk administrator utility vxdiskadm.

6 Perform the following steps to add new disks to the coordinator disk group:

- Add new disks to the node.
- Initialize the new disks as VxVM disks.
- Check the disks for I/O fencing compliance.
- Add the new disks to the coordinator disk group and set the coordinator attribute value as "on" for the coordinator disk group.

See the *Storage Foundation Cluster File System Installation Guide* for detailed instructions.

Note that though the disk group content changes, the I/O fencing remains in the same state.

7 Make sure that the /etc/vxfenmode file is updated to specify the correct disk policy.

See the *Storage Foundation Cluster File System Installation Guide* for more information.

8 From one node, start the vxfenswap utility. You must specify the diskgroup to the utility.

The utility performs the following tasks:

- Backs up the existing /etc/vxfentab file.

- Creates a test file /etc/vxfentab.test for the diskgroup that is modified on each node.
- Reads the diskgroup you specified in the vxfenswap command and adds the diskgroup to the /etc/vxfentab.test file on each node.
- Verifies that the serial number of the new disks are identical on all the nodes. The script terminates if the check fails.
- Verifies that the new disks can support I/O fencing on each node.

9 If the disk verification passes, the utility reports success and asks if you want to commit the new set of coordinator disks.

10 Review the message that the utility displays and confirm that you want to commit the new set of coordinator disks. Else skip to step 11.

```
Do you wish to commit this change? [y/n] (default: n) y
```

If the utility successfully commits, the utility moves the /etc/vxfentab.test file to the /etc/vxfentab file.

11 If you do not want to commit the new set of coordinator disks, answer n.

The vxfenswap utility rolls back the disk replacement operation.

Replacing the coordinator diskgroup in a cluster that is online

You can also replace the coordinator diskgroup using the vxfenswap utility. The following example replaces the coordinator disk group vxfencoorddg with a new disk group vxfendg.

To replace the coordinator diskgroup

1. Make sure system-to-system communication is functioning properly.
2. Make sure that the cluster is online.

```
# vxfenadm -d


I/O Fencing Cluster Information:
================================
Fencing Protocol Version: 201
Fencing Mode: SCSI3
Fencing SCSI3 Disk Policy: dmp
Cluster Members:
  * 0 (system01)
  1 (system02)
RFSM State Information:
  node 0 in state 8 (running)
  node 1 in state 8 (running)
```

3. Find the name of the current coordinator diskgroup (typically vxfencoorddg) that is in the /etc/vxfendg file.

```
# cat /etc/vxfendg
vxfencoorddg
```

4. Find the alternative disk groups available to replace the current coordinator diskgroup.

```
# vxdisk -o alldgs list

DEVICE    TYPE    DISK  GROUP     STATUS
rhdisk64    auto:cdsdisk    -  (vxfendg)      online
rhdisk65    auto:cdsdisk    -  (vxfendg)      online
rhdisk66    auto:cdsdisk    -  (vxfendg)      online
rhdisk75    auto:cdsdisk    -  (vxfencoorddg) online
rhdisk76    auto:cdsdisk    -  (vxfencoorddg) online
rhdisk77    auto:cdsdisk    -  (vxfencoorddg) online
```

5 Validate the new disk group for I/O fencing compliance. Run the following command:

```
# vxfentsthdw -c vxfendg
```

See "Testing the coordinator disk group using vxfentsthdw -c option" on page 74.

6 If the new disk group is not already deported, run the following command to deport the disk group:

```
# vxdg deport vxfendg
```

7 Make sure that the /etc/vxfenmode file is updated to specify the correct disk policy.

See the *Storage Foundation Cluster File System Installation Guide* for more information.

8 From any node, start the vxfenswap utility. For example, if vxfendg is the new diskgroup that you want to use as the coordinator diskgroup:

```
# vxfenswap -g vxfendg [-n]
```

The utility performs the following tasks:

- Backs up the existing /etc/vxfentab file.
- Creates a test file /etc/vxfentab.test for the diskgroup that is modified on each node.
- Reads the diskgroup you specified in the vxfenswap command and adds the diskgroup to the /etc/vxfentab.test file on each node.
- Verifies that the serial number of the new disks are identical on all the nodes. The script terminates if the check fails.
- Verifies that the new disk group can support I/O fencing on each node.

9 If the disk verification passes, the utility reports success and asks if you want to replace the coordinator disk group.

10 Review the message that the utility displays and confirm that you want to replace the coordinator disk group. Else skip to step 13.

```
Do you wish to commit this change? [y/n] (default: n) y
```

If the utility successfully commits, the utility moves the /etc/vxfentab.test file to the /etc/vxfentab file.

The utility also updates the /etc/vxfendg file with this new diskgroup.

11 Set the coordinator attribute value as "on" for the new coordinator disk group.

```
# vxdg -g vxfendg set coordinator=on
```

Set the coordinator attribute value as "off" for the old disk group.

```
# vxdg -g vxfencoorddg set coordinator=off
```

12 Verify that the coordinator disk group has changed.

```
# cat /etc/vxfendg
vxfendg
```

The swap operation for the coordinator disk group is complete now.

13 If you do not want to replace the coordinator disk group, answer n at the prompt.

The vxfenswap utility rolls back any changes to the coordinator diskgroup.

Changing the disk interaction policy in a cluster that is online

In a cluster that is online, you can change the disk interaction policy from dmp to raw using the vxfenswap utility.

To change the disk interaction policy

1 Make sure system-to-system communication is functioning properly.

2 Make sure that the cluster is online.

```
# vxfenadm -d


I/O Fencing Cluster Information:
================================
Fencing Protocol Version: 201
Fencing Mode: SCSI3
Fencing SCSI3 Disk Policy: dmp
Cluster Members:
  * 0 (system01)
  1 (system02)
RFSM State Information:
  node 0 in state 8 (running)
  node 1 in state 8 (running)
```

3 On each node in the cluster, edit the /etc/vxfenmode file to change the disk policy.

4 From any node, start the vxfenswap utility:

```
# vxfenswap -g vxfencoordg [-n]
```

5 Verify the change in the disk policy.

```
# vxfenadm -d

I/O Fencing Cluster Information:
================================
Fencing Protocol Version: 201
Fencing Mode: SCSI3
Fencing SCSI3 Disk Policy: raw
```

Adding disks from a recovered site to the coordinator diskgroup

In a campus cluster environment, consider a case where the primary site goes down and the secondary site comes online with a limited set of disks. When the primary site restores, the primary site's disks are also available to act as coordinator disks. You can use the vxfenswap utility to add these disks to the coordinator diskgroup.

To add new disks from a recovered site to the coordinator diskgroup

1 Make sure system-to-system communication is functioning properly.

2 Make sure that the cluster is online.

```
# vxfenadm -d

I/O Fencing Cluster Information:
================================
Fencing Protocol Version: 201
Fencing Mode: SCSI3
Fencing SCSI3 Disk Policy: dmp
Cluster Members:
  * 0 (system01)
  1 (system02)
RFSM State Information:
  node 0 in state 8 (running)
  node 1 in state 8 (running)
```

3 Verify the name of the coordinator diskgroup.

```
# cat /etc/vxfendg
vxfencoorddg
```

4 Run the following command:

```
# vxdisk -o alldgs list

DEVICE    TYPE    DISK  GROUP    STATUS
rhdisk75    auto:cdsdisk    -  (vxfencoorddg)    online
rhdisk75    auto    -  -    offline
rhdisk75    auto    -  -    offline
```

5 Verify the number of disks used in the coordinator diskgroup.

```
# vxfenconfig -l
I/O Fencing Configuration Information:
======================================
Count                : 1
Disk List
Disk Name        Major  Minor  Serial Number     Policy

/dev/vx/rdmp/rhdisk75       32  48  R450 00013154 0312     dmp
```

6 When the primary site comes online, start the vxfenswap utility on any node in the cluster:

```
# vxfenswap -g vxfencoorddg [-n]
```

7 Verify the count of the coordinator disks.

```
# vxfenconfig -l
I/O Fencing Configuration Information:
======================================
Single Disk Flag      : 0
Count                : 3
Disk List
Disk Name        Major  Minor  Serial Number     Policy

/dev/vx/rdmp/rhdisk75       32  48  R450 00013154 0312     dmp
/dev/vx/rdmp/rhdisk76       32  32  R450 00013154 0313     dmp
/dev/vx/rdmp/rhdisk77       32  16  R450 00013154 0314     dmp
```

Refreshing lost keys on coordinator disks

If the coordinator disks lose the keys that are registered, the cluster might panic when a split-brain occurs.

You can use the vxfenswap utility to replace the coordinator disks with the same disks. The vxfenswap utility registers the missing keys during the disk replacement.

To refresh lost keys on coordinator disks

1 Make sure system-to-system communication is functioning properly.

2 Make sure that the cluster is online.

```
# vxfenadm -d


I/O Fencing Cluster Information:
================================
Fencing Protocol Version: 201
Fencing Mode: SCSI3
Fencing SCSI3 Disk Policy: dmp
Cluster Members:
  * 0 (system01)
  1 (system02)
RFSM State Information:
  node 0 in state 8 (running)
  node 1 in state 8 (running)
```

3 Run the following command to view the coordinator disks that do not have keys:

```
# vxfenadm -s all -f /etc/vxfentab


Device Name: /dev/vx/rdmp/hdisk75
Total Number of Keys: 0
No keys...
...
```

4 On any node, run the following command to start the vxfenswap utility:

```
# vxfenswap -g vxfencoorddg [-n]
```

5 Verify that the keys are atomically placed on the coordinator disks.

```
# vxfenadm -s all -f /etc/vxfentab
```

```
Device Name: /dev/vx/rdmp/hdisk75
Total Number of Keys: 4
...
```

Administering the CP server

This section describes how to perform CP server administrative and maintenance tasks.

cpsadm command environment variables

Table 3-3 describes the environment variables that are required for the `cpsadm` command. The `cpsadm` command detects these environment variables and uses their value when communicating with the CP server. They are used to authenticate and authorize the user.

Note: The environment variables are not required when the `cpsadm` command is run on the CP server, they are required when the `cpsadm` command is run on the SFCFS cluster nodes.

Table 3-3 cpsadm command environment variables

Environment variable	Description
CPS_USERNAME	This is the fully qualified username as configured in VxSS (vssat showcred).
CPS_DOMAINTYPE	One of the following values: ■ vx ■ unixpwd ■ nis ■ nisplus ■ ldap

The environment variables must be exported directly on the shell before running the `cpsadm` command. For example,

```
# export CPS_USERNAME=
# export CPS_DOMAINTYPE=
```

Additionally, the username and domaintype values are the same as those added onto the CP server. To view these values run the following command:

```
# cpsadm -s cp_server -a list_user
```

Adding and removing SFCFS cluster entries from the CP server database

- To add a SFCFS cluster to the CP server database
  Type the following command:

  ```
  # cpsadm -s cp_server -a add_clus -c cluster_name -u uuid
  ```

- To remove a SFCFS cluster from the CP server database
  Type the following command:

  ```
  # cpsadm -s cp_server -a rm_clus -u uuid
  ```

cp_server	The CP server's virtual IP address or virtual hostname.
cluster_name	The SFCFS cluster name.
uuid	The UUID (Universally Unique ID) of the SFCFS cluster.

Adding and removing a SFCFS cluster node from the CP server database

- To add a SFCFS cluster node from the CP server database
  Type the following command:

  ```
  # cpsadm -s cp_server -a add_node -u uuid -n nodeid
  -h host
  ```

- To remove a SFCFS cluster node from the CP server database
  Type the following command:

  ```
  # cpsadm -s cp_server -a rm_node -u uuid -n nodeid
  ```

cp_server	The CP server's virtual IP address or virtual hostname.
uuid	The UUID (Universally Unique ID) of the SFCFS cluster.
nodeid	The node id of the SFCFS cluster node.
host	Hostname

Adding or removing CP server users

- To add a user
  Type the following command:

  ```
  # cpsadm -s cp_server -a add_user -e user_name -f user_role
  -g domain_type -u uuid
  ```

- To remove a user
  Type the following command:

  ```
  # cpsadm -s cp_server -a rm_user -e user_name -g domain_type
  ```

cp_server	The CP server's virtual IP address or virtual hostname.
user_name	The user to be added to the CP server configuration.
user_role	The user role, either cps_admin or cps_operator.
domain_type	The domain type, for example vx, unixpwd, nis, etc.
uuid	The UUID (Universally Unique ID) of the SFCFS cluster.

Listing the CP server users

To list the CP server users

Type the following command:

```
# cpsadm -s cp_server -a list_users
```

Listing the nodes in all the SFCFS clusters

To list the nodes in all the SFCFS cluster

Type the following command:

```
# cpsadm -s cp_server -a list_nodes
```

Listing the membership of nodes in the SFCFS cluster

To list the membership of nodes in SFCFS cluster

Type the following command:

```
# cpsadm -s cp_server -a list_membership -c cluster_name
```

cp_server	The CP server's virtual IP address or virtual hostname.
cluster_name	The SFCFS cluster name.

Preempting a node

To preempt a node

Type the following command:

```
# cpsadm -s cp_server -a preempt_node -u uuid -n nodeid
-v victim_node id
```

cp_server	The CP server's virtual IP address or virtual hostname.
uuid	The UUID (Universally Unique ID) of the SFCFS cluster.
nodeid	The nodeid of the SFCFS cluster node.
victim_node id	The victim node's host name.

Registering and unregistering a node

- To register a node
  Type the following command:

  ```
  # cpsadm -s cp_server -a reg_node -u uuid -n nodeid
  ```

- To unregister a node
  Type the following command:

  ```
  # cpsadm -s cp_server -a unreg_node -u uuid -n nodeid
  ```

cp_server	The CP server's virtual IP address or virtual hostname.
uuid	The UUID (Universally Unique ID) of the SFCFS cluster.
nodeid	The nodeid of the SFCFS cluster node.

Enable and disable access for a user to a SFCFS cluster

- To enable access for a user to a SFCFS cluster
  Type the following command:

  ```
  # cpsadm -s cp_server -a add_clus_to_user -e user
  -f user_role -g domain_type -u uuid
  ```

- To disable access for a user to a SFCFS cluster
  Type the following command:

  ```
  # cpsadm -s cp_server -a rm_clus_from_user -e user_name
  -f user_role -g domain_type -u uuid
  ```

cp_server	The CP server's virtual IP address or virtual hostname.
user_name	The user name to be added to the CP server.
user_role	The user role, either cps_admin or cps_operator.
domain_type	The domain type, for example vx, unixpwd, nis, etc.
uuid	The UUID (Universally Unique ID) of the SFCFS cluster

Stopping the CP server

To stop the CP server

Type the following command:

```
# cpsadm -s cp_server -a halt_cps
```

Checking the connectivity of CP servers

To check the connectivity of a CP server

Type the following command:

```
# cpsadm -s cp_server -a ping_cps
```

Taking a CP server database snapshot

To take a CP server database snapshot

Type the following command:

```
# cpsadm -s cp_server -a db_snapshot
```

Coordination Point replacement for an online cluster

The following procedure can be used to perform a planned replacement of customized coordination points (CP servers or SCSI-3 disks) without incurring application downtime on an online SFCFS cluster.

The migration from SCSI-3 disk-based fencing with vxfen_mode=scsi3 to vxfen_mode=customized in an online cluster is currently not supported. Additionally, migration from vxfen_mode=customized to vxfen_mode=scsi3 in an online cluster is also currently not supported. The following procedure is for replacement of coordination points when fencing is running in customized mode in an online cluster, with vxfen_mechanism=cps.

If there is a requirement for migrating from SCSI-3 disk-based fencing to customized fencing with CP server, refer to the steps in the "Enable fencing in a SFCFS cluster with a new CP server" scenario for guidance:

See “CP server deployment and migration scenarios” on page 106.

In that scenario, the steps can be used as is except that when migrating from SCSI-3 disk-based fencing, the fencing is configured in scsi3 mode while in the "Enable fencing in a SFCFS cluster with a new CP server" scenario fencing is configured in disabled mode.

Note: If multiple clusters share the same CP server, all of clusters have to independently initiate this replacement procedure.

Replacing the coordination points for an online cluster

The following procedure describes how to migrate from SCSI-3 disks to new CP server coordination points.

To replace coordination points for an online cluster

1 Ensure that the SFCFS cluster nodes and users have been added to the new CP server(s).

This can be checked by running the following commands:

```
# cpsadm -s cp_server  -a list_nodes

# cpsadm -s cp_server -a list_users
```

If the SFCFS cluster nodes are not present here, prepare the new CP server(s) for use by the SFCFS cluster:

2 Ensure that fencing is running on the cluster using the old set of coordination points and in customized mode.

For example, enter the following command:

```
# vxfenadm -d

I/O Fencing Cluster Information:
================================
Fencing Protocol Version: 201
Fencing Mode: CUSTOMIZED
Cluster Members:
* 0 (galaxy)
  1 (nebula)
RFSM State Information:
node 0 in state 8 (running)
node 1 in state 8 (running)
```

This procedure is only applicable when fencing is already running with vxfen_mode=customized and mechanism=cps in the /etc/vxfenmode file.

3 Back up the /etc/vxfenmode file on each of the SFCFS cluster nodes.

4 Use a text editor to access /etc/vxfenmode and update the values to the new CP server (coordination points).

The values of the /etc/vxfenmode file has to be updated on all the nodes in the SFCFS cluster.

Review and if necessary, update the vxfenmode parameters for security, the coordination points, and if applicable to your configuration, vxfendg.

Refer to the text information within the vxfenmode file for additional information about these parameters and their new possible values.

5 Run vxfenswap utility from one of the nodes of the cluster.

The vxfenswap utility requires secure ssh connection to all the cluster nodes.

Use -n to use rsh instead of default ssh.

For example:

```
# /opt/VRTSvcs/vxfen/bin/vxfenswap -n

VERITAS vxfenswap version 5.1 Solaris

The logfile generated for vxfenswap is
/var/VRTSvcs/log/vxfen/vxfenswap.log.
19156

Please Wait...

VXFEN vxfenconfig NOTICE Driver will use customized fencing
- mechanism cps

Validation of coordination points change has succeeded on
all nodes.

You may commit the changes now.
WARNING: This may cause the whole cluster to panic
if a node leaves membership before the change is complete.
```

6 If validation of coordination points from all the nodes fails, the vxfenswap utility rollbacks the coordination point replacement operation. Proceed to restore /etc/vxfenmode with the backed up file on all the SFCFS cluster nodes.

7 You are then prompted to commit the change. Enter y for yes.

```
Do you wish to commit this change? [y/n] (default: n)
y

Successfully completed the online
coordination point replacement operation.
```

8 Confirm the successful execution of the vxfenswap utility by checking the coordination points currently used by the vxfen driver.

For example, run the following command:

```
# vxfenconfig -l
```

Note: A running online coordination point replacement operation can be canceled at any time using the command: `vxfenswap -a cancel`.

Refreshing registration keys on the coordination points for server-based fencing

Replacing keys on an online coordination point (CP server) involves refreshing that coordination point's registrations. You can perform a planned refresh of registrations on a CP server without incurring application downtime on the SFCFS cluster.

Note: Refresh registrations on a CP server, if the CP server agent issues an alert on the loss of such registrations on the CP server database.

The following procedure describes how to refresh the coordination point registrations.

Refreshing registration keys on the coordination points for server-based fencing

1 Ensure that the SFCFS cluster nodes and users have been added to the new CP server(s).

This can be checked by running the following commands:

```
# cpsadm -s cp_server -a list_nodes

# cpsadm -s cp_server -a list_users
```

2 Ensure that fencing is running on the cluster in customized mode using the coordination points mentioned in /etc/vxfenmode file.

For example, enter the following command:

```
# vxfenadm -d

================================
Fencing Protocol Version: 201
Fencing Mode: CUSTOMIZED
Cluster Members:
* 0 (galaxy)
1 (nebula)
RFSM State Information:
node 0 in state 8 (running)
node 1 in state 8 (running)

# vxfenconfig -l
```

This should show the coordination points currently used by I/O fencing.

3 Run the vxfenswap utility from one of the nodes of the cluster.

The vxfenswap utility requires secure ssh connection to all the cluster nodes. Use -n to use rsh instead of default ssh.

For example:

```
# /opt/VRTSvcs/vxfen/bin/vxfenswap -n


VERITAS vxfenswap version 5.1 Solaris
The logfile generated for vxfenswap is
/var/VRTSvcs/log/vxfen/vxfenswap.log.
19156
Please Wait...
VXFEN vxfenconfig NOTICE Driver will use customized fencing
- mechanism cps
Validation of coordination points change has succeeded on
all nodes.
You may commit the changes now.
WARNING: This may cause the whole cluster to panic
if a node leaves membership before the change is complete.
```

4 You are then prompted to commit the change. Enter y for yes.

```
Do you wish to commit this change? [y/n] (default: n)
y
Successfully completed the online
coordination point replacement operation.
```

5 Confirm the successful execution of the vxfenswap utility. If CP agent is configured, it should report ONLINE as it succeeds to find the registrations on coordination points. The registrations on the CP server and coordinator disks can be viewed using the cpsadm and vxfenadm utilities respectively.

Note that a running online coordination point refreshment operation can be canceled at any time using the command:

```
# vxfenswap -a cancel
```

CP server deployment and migration scenarios

Table 3-4 describes the supported deployment and migration scenarios, as well as the required procedures to be performed on the SFCFS cluster and CP server node or cluster.

Table 3-4 CP server deployment and migration scenarios

Scenario	CP server	SFCFS cluster	Action required
Setup of CP server for a SFCFS cluster for the first time	New CP server	New SFCFS cluster using CP server as coordination point	On the designated CP server, perform the following tasks: **1** Prepare to configure the new CP server. **2** Configure the new CP server. On the SFCFS cluster nodes, configure server-based I/O fencing. See the *Storage Foundation Cluster File System Installation Guide* for the procedures.
Add a new SFCFS cluster to an existing and operational CP server	Existing and operational CP server	New SFCFS cluster	On the SFCFS cluster nodes, configure server-based I/O fencing. See the *Storage Foundation Cluster File System Installation Guide* for the procedures.
Replace the coordination point from an existing CP server to a new CP server	New CP server	Existing SFCFS cluster using CP server as coordination point	On the designated CP server, perform the following tasks: **1** Prepare to configure the new CP server. **2** Configure the new CP server. **3** Prepare the new CP server for use by the SFCFS cluster See the *Storage Foundation Cluster File System Installation Guide* for the procedures. On the SFCFS cluster nodes run the `vxfenswap` command to move to replace the CP server: See " Coordination Point replacement for an online cluster" on page 101.

Table 3-4 CP server deployment and migration scenarios *(continued)*

Scenario	CP server	SFCFS cluster	Action required
Replace the coordination point from an existing CP server to an operational CP server coordination point	Operational CP server	Existing SFCFS cluster using CP server as coordination point	On the designated CP server, prepare to configure the new CP server manually. See the *Storage Foundation Cluster File System Installation Guide* for the procedures. On the SFCFS cluster run the `vxfenswap` command to move to replace the CP server: See " Coordination Point replacement for an online cluster" on page 101.

Table 3-4 CP server deployment and migration scenarios *(continued)*

Scenario	CP server	SFCFS cluster	Action required

Enabling fencing in a SFCFS cluster with a new CP server coordination point
Note: This procedure incurs application downtime on the SFCFS cluster.
	New CP server	Existing SFCFS cluster with fencing configured in disabled mode	On the designated CP server, perform the following tasks:
			1 Prepare to configure the new CP server.
			2 Configure the new CP server
			3 Prepare the new CP server for use by the SFCFS cluster
			See the *Storage Foundation Cluster File System Installation Guide* for the procedures.
			On the SFCFS cluster nodes, perform the following:
			1 Stop all applications, VCS, and fencing on the SFCFS cluster.
			2 To stop VCS, use the following command (to be run on all the SFCFS cluster nodes):
			`# hastop -local`
			3 Stop fencing using the following command:
			`# /etc/init.d/vxfen.rc stop`
			4 Re-configure fencing on the SFCFS cluster.
			See the *Storage Foundation Cluster File System Installation Guide* for the procedures.

Table 3-4 CP server deployment and migration scenarios *(continued)*

Scenario	CP server	SFCFS cluster	Action required

Enabling fencing in a SFCFS cluster with an operational CP server coordination point
Note: This procedure incurs application downtime.
	Operational CP server	Existing SFCFS cluster with fencing configured in disabled mode	On the designated CP server, prepare to configure the new CP server.
			See the *Storage Foundation Cluster File System Installation Guide* for this procedure.
			On the SFCFS cluster nodes, perform the following tasks:
			1 Stop all applications, VCS, and fencing on the SFCFS cluster.
			2 To stop VCS, use the following command (to be run on all the SFCFS cluster nodes):
			`# hastop -all`
			3 Stop fencing using the following command:
			`# /etc/init.d/vxfen.rc stop`
			4 Re-configure fencing on the SFCFS cluster.
			See the *Storage Foundation Cluster File System Installation Guide* for the procedures.

Refreshing registrations of SFCFS cluster nodes on coordination points (CP servers/ coordinator disks) without incurring application downtime	Operational CP server	Existing SFCFS cluster using the CP server as coordination point	On the SFCFS cluster run the `vxfenswap` command to refresh the keys on the CP server: See "Refreshing registration keys on the coordination points for server-based fencing" on page 104.

Migrating from non-secure to secure setup for CP server and SFCFS cluster communication

The following procedure describes how to migrate from a non-secure to secure set up for the CP server and SFCFS cluster.

To migrate from non-secure to secure setup for CP server and SFCFS cluster

1. Stop fencing on all the SFCFS cluster nodes of all the clusters (which are using the CP servers).

   For example:

   ```
   # /etc/init.d/vxfen.rc stop
   ```

2. Stop all the CP servers using the following command on each CP server:

   ```
   # hagrp -offline CPSSG -any
   ```

3. Ensure that security is configured for communication between CP servers and SFCFS cluster nodes.

   If security is not configured, please refer to:

   See "About secure communication between the SFCFS cluster and CP server" on page 42.

4. Modify /etc/vxcps.conf on each CP server to set security=1.

5. Start CP servers using the following command on all of them:

   ```
   # hagrp -online CPSSG -any
   ```

6. Add the following user for each client node on each CP server:

   _HA_VCS_*hostname*@HA_SERVICES@*FQHN*

   where, hostname is the client node name without qualification, and FQHN is Fully Qualified Host Name of the client node.

7 Add the users to the CP server database.

For example, issue the following commands on the CP server (CPS server.symantecexample.com):

```
# cpsadm -s CPS server.symantecexample.com -a add_user -e\
_HA_VCS_galaxy@HA_SERVICES@galaxy.symantec.com\
-f cps_operator -g vx

User _HA_VCS_galaxy@HA_SERVICESgalaxy.symantec.com successfully added

# cpsadm -s CPS server.symantecexample.com -a add_user -e\
_HA_VCS_nebula6@HA_SERVICES@nebula.symantec.com\
-f cps_operator -g vx

User _HA_VCS_nebula@HA_SERVICES@nebula.symantec.com successfully added
```

8 Authorize the user to administer the cluster.

For example, issue the following command on the CP server (CPS server.symantecexample.com):

```
# cpsadm -s CPS server.symantecexample.com -a\
add_clus_to_user -c cpcluster\
-u {f0735332-1dd1-11b2-a3cb-e3709c1c73b9}\
-e _HA_VCS_galaxy@HA_SERVICES@galaxy.symantec.com\
-f cps_operator -g vx

Cluster successfully added to user

_HA_VCS_galaxy@HA_SERVICES@galaxy.symantec.com privileges.
```

9 Modify /etc/vxfenmode file on each SFCFS cluster node to set security=1.

10 After modifying the /etc/vxfenmode file, run the vxfen init script on the SFCFS cluster node to start fencing.

For an example:

```
# /etc/init.d/vxfen.rc start
```

About VXFEN tunable parameters

The section describes the VXFEN tunable parameters and how to reconfigure the VXFEN module.

Table 3-5 describes the tunable parameters for the VXFEN driver.

Table 3-5 VXFEN tunable parameters

vxfen Parameter	Description and Values: Default, Minimum, and Maximum
vxfen_deblog_sz	Size of debug log in bytes ■ Values Default: 65536 Minimum: 65536 Maximum: 256K
vxfen_max_delay	Specifies the maximum number of seconds that the smaller sub-cluster waits before racing with larger sub-clusters for control of the coordinator disks when a split-brain occurs. This value must be greater than the vxfen_max_delay value. ■ Values Default: 60 Minimum: 1 Maximum: 600
vxfen_min_delay	Specifies the minimum number of seconds that the smaller sub-cluster waits before racing with larger sub-clusters for control of the coordinator disks when a split-brain occurs. This value must be smaller than the vxfen_max_delay value. ■ Values Default: 1 Minimum: 1 Maximum: 600
vxfen_vxfnd_tmt	Specifies the time in seconds that the I/O fencing driver VxFEN waits for the I/O fencing daemon VXFEND to return after completing a given task. ■ Values Default: 60 Minimum: 10 Maximum: 600

In the event of a network partition, the smaller sub-cluster delays before racing for the coordinator disks. The time delayed allows a larger sub-cluster to win the race for the coordinator disks. The vxfen_max_delay and vxfen_min_delay parameters define the delay in seconds.

Configuring the VXFEN module parameters

After adjusting the tunable kernel driver parameters, you must reconfigure the VXFEN module for the parameter changes to take effect.

The following example procedure changes the value of the vxfen_min_delay parameter.

Note: You must restart the VXFEN module to put any parameter change into effect.

To configure the VxFEN parameters and reconfigure the VxFEN module

1 Check the status of the driver and start the driver if necessary:

```
# /etc/methods/vxfenext -status
```

To start the driver:

```
# /etc/init.d/vxfen.rc start
```

2 List the current value of the tunable parameter:

For example:

```
# lsattr -El vxfen

vxfen_deblog_sz 65536 N/A True
vxfen_max_delay 3 N/A True
vxfen_min_delay 1 N/A True
vxfen_vxfnd_tmt 60 N/A True
```

The current value of the vxfen_min_delay parameter is 1 (the default).

3 Enter the following command to change the vxfen_min_delay parameter value:

```
# chdev -l vxfen -P -a vxfen_min_delay=30
```

4 Ensure that SFCFS is shut down. Unload and reload the driver after changing the value of the tunable:

```
# /etc/methods/vxfenext -stop

# /etc/methods/vxfenext -start
```

5 Repeat the above steps on each cluster node to change the parameter.

Chapter 4

Using Veritas Extension for Oracle Disk Manager

This chapter includes the following topics:

About Oracle Disk Manager

Veritas Extension for Oracle Disk Manager is specifically designed for Oracle10g or later to enhance file management and disk I/O throughput. The features of Oracle Disk Manager are best suited for databases that reside in a file system contained in Veritas File System. Oracle Disk Manager allows Oracle10g or later users to improve database throughput for I/O intensive workloads with special I/O optimization.

Veritas Extension for Oracle Disk Manager supports Oracle Resilvering. With Oracle Resilvering, the storage layer receives information from the Oracle database

as to which regions or blocks of a mirrored datafile to resync after a system crash. Oracle Resilvering avoids overhead from the VxVM DRL, which increases performance.

Oracle Disk Manager reduces administrative overhead by providing enhanced support for Oracle Managed Files. Veritas Extension for Oracle Disk Manager has Quick I/O-like capabilities, but is transparent to the user. Unlike Veritas Quick I/O, files managed using Veritas Extension for Oracle Disk Manager do not require special file naming conventions. The Oracle Disk Manager interface uses regular database files. If you are upgrading to Oracle10g or later, you should convert from Quick I/O to Oracle Disk Manager.

Database administrators can choose the datafile type used with the Oracle product. Historically, choosing between file system files and raw devices was based on manageability and performance. The exception to this is a database intended for use with Oracle Parallel Server, which requires raw devices on most platforms. If performance is not as important as administrative ease, file system files are typically the preferred file type. However, while an application may not have substantial I/O requirements when it is first implemented, I/O requirements may change. If an application becomes dependent upon I/O throughput, converting datafiles from file system to raw devices is often necessary.

Oracle Disk Manager was designed to work with Oracle10g or later to provide both performance and manageability. Oracle Disk Manager provides support for Oracle's file management and I/O calls for database storage on VxFS file systems and on raw volumes or partitions. This feature is provided as a dynamically-loaded shared library with which Oracle binds when it is loaded. The Oracle Disk Manager library works with an Oracle Disk Manager driver that is loaded in the kernel to perform its functions.

If you are upgrading to Oracle10g or later, you should convert from Quick I/O to Oracle Disk Manager.

The benefits of using Oracle Disk Manager are as follows:

- True kernel asynchronous I/O for files and raw devices
- Reduced system call overhead
- Improved file system layout by preallocating contiguous files on a VxFS file system
- Performance on file system files that is equivalent to raw devices
- Transparent to users
- Contiguous datafile allocation

How Oracle Disk Manager improves database performance

Oracle Disk Manager improves database I/O performance to VxFS file systems by:

- Supporting kernel asynchronous I/O
- Supporting direct I/O and avoiding double buffering
- Avoiding kernel write locks on database files
- Supporting many concurrent I/Os in one system call
- Avoiding duplicate opening of files per Oracle instance
- Allocating contiguous datafiles

About kernel asynchronous I/O support

Asynchronous I/O performs non-blocking system level reads and writes, allowing the system to perform multiple I/O requests simultaneously. Kernel asynchronous I/O is better than library asynchronous I/O because the I/O is queued to the disk device drivers in the kernel, minimizing context switches to accomplish the work.

About direct I/O support and avoiding double buffering

I/O on files using read() and write() system calls typically results in data being copied twice: once between the user and kernel space, and the other between kernel space and the disk. In contrast, I/O on raw devices is copied directly between user space and disk, saving one level of copying. As with I/O on raw devices, Oracle Disk Manager I/O avoids the extra copying. Oracle Disk Manager bypasses the system cache and accesses the files with the same efficiency as raw devices. Avoiding double buffering reduces the memory overhead on the system. Eliminating the copies from kernel to user address space significantly reduces kernel mode processor utilization freeing more processor cycles to execute the application code.

About avoiding kernel write locks on database files

When database I/O is performed by way of the write() system call, each system call acquires and releases a kernel write lock on the file. This lock prevents simultaneous write operations on the same file. Because database systems usually implement their own locks for managing concurrent access to files, write locks unnecessarily serialize I/O writes. Oracle Disk Manager bypasses file system locking and lets the database server control data access.

About supporting many concurrent I/Os in one system call

When performing asynchronous I/O, an Oracle process may try to issue additional I/O requests while collecting completed I/Os, or it may try to wait for particular I/O requests synchronously, as it can do no other work until the I/O is completed. The Oracle process may also try to issue requests to different files. All this activity can be accomplished with one system call when Oracle uses the Oracle Disk Manager I/O interface. This interface reduces the number of system calls performed to accomplish the same work, reducing the number of user space/kernel space context switches.

Note: With Oracle10g Release 2 (9.2) or later, you can use the `FILESYSTEMIO_OPTIONS` initialization parameter to enable or disable asynchronous I/O, direct I/O, or Concurrent I/O on file system files. This parameter is applicable to JFS and JFS2 files only. This parameter is not applicable to VxFS files, ODM files, or QIO files. For further information, please refer to Oracle Disk Manager documentation provided by Oracle.

About avoiding duplicate file opens

Oracle Disk Manager allows files to be opened once, providing a "file identifier." This is called "identifying" the files. The same file identifiers can be used by any other processes in the Oracle instance. The file status is maintained by the Oracle Disk Manager driver in the kernel. The reduction in file open calls reduces processing overhead at process initialization and termination, and it reduces the number of file status structures required in the kernel.

About allocating contiguous datafiles

Oracle Disk Manager can improve performance for queries, such as sort and parallel queries, that use temporary tablespaces. Without Oracle Disk Manager, Oracle does not initialize the datafiles for the temporary tablespaces. Therefore, the datafiles become sparse files and are generally fragmented. Sparse or fragmented files lead to poor query performance. When using Oracle Disk Manager, the datafiles are initialized for the temporary tablespaces and are allocated in a contiguous fashion, so that they are not sparse.

About Oracle Disk Manager and Storage Foundation Cluster File System

Oracle Disk Manager supports access to clustered files in the SFCFS environment. With a Veritas Storage Foundation Cluster File System license, ODM supports

SFCFS files in a serially-exclusive mode which allows access to each SFCFS file by one node at a time, but does not allow simultaneous access from multiple nodes.

See the `mount_odm`(1) man page for more information on its cluster support modes.

About Oracle Disk Manager and Oracle Managed Files

Oracle10g or later offers a feature known as Oracle Managed Files (OMF). OMF manages datafile attributes such as file names, file location, storage attributes, and whether or not the file is in use by the database. OMF is only supported for databases that reside in file systems. OMF functionality is greatly enhanced by Oracle Disk Manager.

The main requirement for OMF is that the database be placed in file system files. There are additional prerequisites imposed upon the file system itself.

OMF is a file management feature that:

- Eliminates the task of providing unique file names
- Offers dynamic space management by way of the tablespace auto-extend functionality of Oracle10g or later

OMF should only be used in file systems that reside within striped logical volumes, which support dynamic file system growth. File systems intended for OMF use must also support large, extensible files in order to facilitate tablespace auto-extension. Raw partitions cannot be used for OMF.

By default, OMF datafiles are created with auto-extend capability. This attribute reduces capacity planning associated with maintaining existing databases and implementing new applications. Due to disk fragmentation that occurs as the tablespace grows over time, database administrators have been somewhat cautious when considering auto-extensible tablespaces. Oracle Disk Manager eliminates this concern.

When Oracle Disk Manager is used in conjunction with OMF, special care is given within Veritas Extension for Disk Manager to ensure that contiguous disk space is allocated to datafiles, including space allocated to a tablespace when it is auto-extended. The table and index scan throughput does not decay as the tablespace grows.

How Oracle Disk Manager works with Oracle Managed Files

The following example illustrates the relationship between Oracle Disk Manager and Oracle Managed Files (OMF). The example shows the `init.ora` contents and the command for starting the database instance. To simplify Oracle UNDO

management, the new Oracle10g or later `init.ora` parameter `UNDO_MANAGEMENT` is set to `AUTO`. This is known as System-Managed Undo.

Note: Before building an OMF database, you need the appropriate `init.ora` default values. These values control the location of the `SYSTEM` tablespace, online redo logs, and control files after the `CREATE DATABASE` statement is executed.

```
$ cat initPROD.ora
UNDO_MANAGEMENT = AUTO
DB_CREATE_FILE_DEST = '/PROD'
DB_CREATE_ONLINE_LOG_DEST_1 = '/PROD'
db_block_size = 4096
db_name = PROD
$ sqlplus /nolog
SQL> connect / as sysdba
SQL> startup nomount pfile= initPROD.ora
```

The Oracle instance starts.

```
Total System Global Area 93094616 bytes
Fixed Size 279256 bytes
Variable Size 41943040 bytes
Database Buffers 50331648 bytes
Redo Buffers 540672 bytes
```

To implement a layout that places files associated with the `EMP_TABLE` tablespace in a directory separate from the `EMP_INDEX` tablespace, use the `ALTER SYSTEM` statement. This example shows how OMF handles file names and storage clauses and paths. The layout allows you to think of the tablespaces as objects in a file system as opposed to a collection of datafiles. Since OMF uses the Oracle Disk Manager file resize function, the tablespace files are initially created with the default size of 100MB and grow as needed. Use the `MAXSIZE` attribute to limit growth.

The following example shows the commands for creating an OMF database and for creating the `EMP_TABLE` and `EMP_INDEX` tablespaces in their own locale.

Note: The directory must exist for OMF to work, so the `SQL*Plus HOST` command is used to create the directories:

```
SQL> create database PROD;
```

The database is created.

```
SQL> HOST mkdir /PROD/EMP_TABLE;
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/PROD/EMP_TABLE';
```

The system is altered.

```
SQL> create tablespace EMP_TABLE DATAFILE AUTOEXTEND ON MAXSIZE \
500M;
```

A tablespace is created.

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/PROD/EMP_INDEX';
```

The system is altered.

```
SQL> create tablespace EMP_INDEX DATAFILE AUTOEXTEND ON MAXSIZE \
100M;
```

A tablespace is created.

Use the `ls` command to show the newly created database:

```
$ ls -lFR
total 638062
drwxr-xr-x 2 oracle10g dba 96 May  3 15:43 EMP_INDEX/
drwxr-xr-x 2 oracle10g dba 96 May  3 15:43 EMP_TABLE/
-rw-r--r-- 1 oracle10g dba 104858112 May 3 17:28 ora_1_BEhYgc0m.log
-rw-r--r-- 1 oracle10g dba 104858112 May 3 17:27 ora_2_BEhYu4NA.log
-rw-r--r-- 1 oracle10g dba 806912 May 3 15:43 ora_BEahlfUX.ctl
-rw-r--r-- 1 oracle10g dba 10489856 May 3 15:43 ora_sys_undo_BEajPSVq.dbf
-rw-r--r-- 1 oracle10g dba 104861696 May 3 15:4 ora_system_BEaiFE8v.dbf
-rw-r--r-- 1 oracle10g dba 186 May 3 15:03 PROD.ora

./EMP_INDEX:
total 204808
-rw-r--r-- 1 oracle10g dba 104861696 May 3 15:43
ora_emp_inde_BEakGfun.dbf

./EMP_TABLE:
total 204808
-rw-r--r-- 1 oracle10g dba 104861696 May 3 15:43
ora_emp_tabl_BEak1LqK.dbf
```

Setting up Veritas Extension for Oracle Disk Manager

Veritas Extension for Oracle Disk Manager is part of Veritas Storage Foundation Standard and Enterprise products. Veritas Extension for Oracle Disk Manager is enabled once your Veritas Storage Foundation Standard or Enterprise product and Oracle10g or later are installed. The Veritas Extension for Oracle Disk Manager library is linked to the library in the `{ORACLE_HOME}/lib` directory.

If you are performing a local Oracle installation, not on the SFCFS file system, then ODM linking needs to be performed on all nodes in the cluster.

Before setting up Veritas Extension for Oracle Disk Manager, the following conditions must be met:

Prerequisites

- Veritas Storage Foundation for cluster File System must be installed on your system.
- Oracle10g, or later, must be installed on your system.
- If Cached Quick I/O is available, do not enable Oracle Disk Manager when Cached Quick I/O is enabled for datafiles.

Usage Notes

- When the Quick I/O feature is available, Oracle Disk Manager uses the Quick I/O driver to perform asynchronous I/O. Do not turn off the Quick I/O mount option, which is the default.
- Oracle uses default file access methods if Oracle10g or later or Veritas Storage Foundation for Cluster File System is not installed, or VxFS 5.0 is not available in the kernel.

How to prepare existing database storage for Oracle Disk Manager

Non-Quick I/O files in a VxFS file system work with Oracle Disk Manager without any changes. The files are found and identified for Oracle Disk Manager I/O by default. To take full advantage of Oracle Disk Manager datafiles, files should not be fragmented.

If you are using Quick I/O files in a VxFS file system and you want to move to Oracle Disk Manager, convert the Quick I/O files to normal files using the `qio_convertdbfiles -u` command.

You must be running Oracle10g or later to use Oracle Disk Manager.

Converting Quick I/O files to Oracle Disk Manager files

If you plan to run the Veritas product with Oracle10g or later, and you have been using Quick I/O files, Symantec recommends that you convert your Quick I/O files to regular files. This should be done after you upgrade.

Note: If you are running an earlier version of Oracle (Oracle 8.x or lower), you should not convert your Quick I/O files because Oracle Disk Manager is for Oracle10g or later only.

The Oracle Disk Manager uses the Quick I/O driver to perform asynchronous I/O, do not turn off the Quick I/O mount option, which is the default.

To convert Quick I/O files to Oracle Disk Manager files

1. As Oracle DBA, run `qio_getdbfiles` to retrieve a list of all datafiles.

   ```
   $ /opt/VRTS/bin/qio_getdbfiles -T ora -a
   ```

   The list is compiled in a file named `mkqio.dat`.

2. Shutdown the database.
3. As Oracle DBA, run `qio_convertdbfiles` in the directory containing the `mkqio.dat` file. The `qio_convertdbfiles` script converts all Quick I/O files to ODM files.

   ```
   $ /opt/VRTS/bin/qio_convertdbfiles -T ora -u
   ```

4. Restart the database instance.

Verifying that Oracle Disk Manager is configured

Before verifying that Oracle Disk Manager is configured, make sure that the following conditions are met:

Prerequisites

- `/opt/VRTSodm/lib/libodm64.so` must exist.
- If you are using Oracle 10g, `$ORACLE_HOME/lib/libodm10.so` is linked to `/opt/VRTSodm/lib/libodm64.so`.
- If you are using Oracle 11g, $ORACLE_HOME/lib/libodm11.so is linked to /opt/VRTSodm/lib/libodm64.so.
- The `VRTSdbed` license must be valid.
- The `VRTSodm` package must be installed.

To verify that Oracle Disk Manager is configured

1 Verify that the ODM feature is included in the license:

```
# /opt/VRTS/bin/vxlicrep | grep ODM
```

The output verifies that ODM is enabled.

Note: Verify that the license key containing the ODM feature is not expired. If the license key has expired, you will not be able to use the ODM feature.

---

2 Check that the `VRTSodm` package is installed:

```
# lslpp -L VRTSodm
Fileset   Level   State  Type  Description (Uninstaller)
----------------------------------------------------------
VRTSodm   5.1.x.x   C F  Veritas Extension for
                         Oracle Disk Manager

State codes:
A -- Applied.
B -- Broken.
C -- Committed.
O -- Obsolete.  (partially migrated to newer version)
? -- Inconsistent State...Run lppchk -v.



Type codes:
F -- Installp Fileset
P -- Product
C -- Component
T -- Feature
R -- RPM Package
```

3 Check that `libodm.so` is present.

```
# ls -lL /opt/VRTSodm/lib/libodm64.so
-rw-r--r-- 1 root sys 14336 Apr 25 18:42
/opt/VRTSodm/lib/libodm.so
```

To verify that Oracle Disk Manager is running

1. Start the Oracle database.
2. Check that the instance is using the Oracle Disk Manager function:

```
# cat /dev/odm/stats
# echo $?
0
```

3. Verify that the Oracle Disk Manager is loaded:

   You can use the `genkld` or the `genkex` commands:

```
# genkld | grep odm
or
# genkex | grep odm
```

4. In the alert log, verify the Oracle instance is running. The log should contain output similar to the following:

```
Oracle instance running with ODM: Veritas 5.1.00.00 ODM Library,
Version 2.0
```

Disabling the Oracle Disk Manager feature

Since the Oracle Disk Manager feature uses regular files, you can access these files as regular VxFS files as soon as the feature is disabled.

Note: To convert to VxFS with Quick I/O, disable Oracle Disk Manager using the following procedure, then convert the files to Quick I/O files.

See "Converting Quick I/O files to Oracle Disk Manager files" on page 123.

Before disabling the Oracle Disk Manager feature, you may want to back up your files.

To disable the Oracle Disk Manager feature in an Oracle instance

1. Shut down the database instance.
2. Use the `rm` and `ln` commands to remove the link to the Oracle Disk Manager Library.

   For Oracle 11g, enter:

   ```
   # rm ${ORACLE_HOME}/lib/libodm11.so
   # ln -s ${ORACLE_HOME}/lib/libodm.so \
   ${ORACLE_HOME}/lib/libodm11.so
   ```

   For Oracle 10g, enter:

   ```
   # rm ${ORACLE_HOME}/lib/libodm10.so
   # ln -s ${ORACLE_HOME}/lib/libodm.so \
   ${ORACLE_HOME}/lib/libodm10.so
   ```

3. Restart the database instance.

About Cached ODM

ODM I/O normally bypasses the file system cache and directly reads from and writes to disk. Cached ODM enables some I/O to use caching and read ahead, which can improve ODM I/O performance. Cached ODM performs a conditional form of caching that is based on per-I/O hints from Oracle. The hints indicate what Oracle does with the data. ODM uses these hints to perform caching and read ahead for some reads, but ODM avoids caching other reads, even for the same file.

You can enable cached ODM only for local mount files. Cached ODM does not affect the performance of files and file systems for which you did not enable caching.

See "Enabling Cached ODM for file systems" on page 127.

Cached ODM can be configured in two ways. The primary configuration method is to turn caching on or off for all I/O on a per-file basis. The secondary configuration method is to adjust the ODM cachemap. The cachemap maps file type and I/O type combinations into caching advisories.

See "Tuning Cached ODM settings for individual files" on page 127.

Enabling Cached ODM for file systems

Cached ODM is initially disabled on a file system. You enable Cached ODM for a file system by setting the `odm_cache_enable` option of the `vxtunefs` command after the file system is mounted.

See the `vxtunefs`(1M) manual page.

Note: The `vxtunefs` command enables conditional caching for all of the ODM files on the file system.

To enable Cached ODM for a file system

1 Enable Cached ODM on the VxFS file system `/database01`:

```
# vxtunefs -s -o odm_cache_enable=1 /database01
```

2 Optionally, you can make this setting persistent across mounts by adding a file system entry in the file `/etc/vx/tunefstab`:

```
/dev/vx/dsk/datadg/database01 odm_cache_enable=1
```

See the `tunefstab`(4) manual page.

Tuning Cached ODM settings for individual files

You can use the `odmadm setcachefile` command to override the cachemap for a specific file so that ODM caches either all or none of the I/O to the file. The caching state can be ON, OFF, or DEF (default). The DEF caching state is conditional caching, meaning that for each I/O, ODM consults the cachemap and determines whether the specified file type and I/O type combination should be cached. The ON caching state causes the specified file always to be cached, while the OFF caching state causes the specified file never to be cached.

See the `odmadm`(1M) manual page.

Note: The cache advisories operate only if Cached ODM is enabled for the file system. If the `odm_cache_enable` flag is zero, Cached ODM is OFF for all of the files in that file system, even if the individual file cache advisory for a file is ON.

To enable unconditional caching on a file

- Enable unconditional caching on the file `/mnt1/file1`:

```
# odmadm setcachefile /mnt1/file1=on
```

  With this command, ODM caches all reads from `file1`.

To disable caching on a file

- Disable caching on the file `/mnt1/file1`:

```
# odmadm setcachefile /mnt1/file1=off
```

  With this command, ODM does not cache reads from `file1`.

To check on the current cache advisory settings for a file

- Check the current cache advisory settings of the files `/mnt1/file1` and `/mnt2/file2`:

```
# odmadm getcachefile /mnt1/file1 /mnt2/file2
/mnt1/file1,ON
/mnt2/file2,OFF
```

To reset all files to the default cache advisory

- Reset all files to the default cache advisory:

```
# odmadm resetcachefiles
```

Chapter 5

Agents for Storage Foundation Cluster File System

This chapter includes the following topics:

About agents for Storage Foundation Cluster File System

Agents are processes that manage predefined resource types. When an agent is started, it obtains configuration information from the Veritas Cluster Server (VCS). It then periodically monitors the resources and updates VCS with the resource status.

Agents typically do the following:

- Bring resources online
- Take resources offline
- Monitor resources and report any state changes to VCS

To use volumes as part of an Replicated Volume Group (RVG), configure the required RVG agents. The CVMVolDg resource does not support managing or monitoring volumes that are part of RVG.

For more information about RVG agents, see the *Veritas Cluster Server Agents for Veritas Volume Replicator Configuration Guide*.

VCS bundled agents are part of VCS and are installed when VCS is installed. The cluster functionality agents are add-on resources to VCS for the Veritas File System and Veritas Volume Manager (VxVM). Cluster functionality agents and resource types are part of the `VRTScavf` package and are configured when you run the `cfscluster config` command.

See the *Veritas Cluster Server Bundled Agents Reference Guide*.

This appendix includes the following topics:

Storage Foundation Cluster File System agents

SFCFS includes the following agents:

Veritas Cluster Server cluster components

Resources, attributes, and service groups are components integral to cluster functionality.

See the *Veritas Cluster Server User's Guide*.

Resources

Resources are hardware or software entities, such as disks, volumes, file system mount points, network interface cards (NICs), IP addresses, applications, and databases. Resources work together to provide a service to clients in a client/server environment. Resource types are defined in the `types.cf` file by a collection of attributes. The VCS configuration file, `main.cf`, contains the values for the attributes of the resources. The `main.cf` file incorporates the resources listed in the `types.cf` by way of an `include` directive.

Attributes

Attributes contain data regarding the cluster, nodes, service groups, resources, resource types, and agents. A specified value for a given attribute configures the resource to function in a specific way. By modifying the value of an attribute of a resource, you change the way the VCS agent manages the resource. Each attribute has a definition and a value. You define an attribute by specifying its data type and dimension. Attributes also have default values that are assigned when a value is not specified.

Service groups

Service groups are comprised of related resources. When a service group is brought online, all the resources within the group are brought online.

Modifying the agents and their resources

You can use the VCS Cluster Manager GUI, or enter VCS commands (the "ha" commands such as `hastatus` and `haconf`) from the command line, to modify the configuration of the resources managed by an agent. You can also edit the `main.cf` file directly, but you must reboot your system for the changes to take effect. An example `main.cf` file is located in the `/etc/VRTSvcs/conf/sample_cvm` directory.

It is advisable to use the Veritas Cluster Server GUI to administer your cluster file system resources.

See *Veritas Cluster Server Installation Guide*.

Resources and service groups for File System cluster functionality

Managing cluster mounts through VCS requires various resources types, resources, and service groups.

The following VCS resource types required for Veritas Volume Manager cluster functionality (or CVM) are:

- CVMCluster
- CVMVolDg

CVMCluster controls the overall operation of CVM. The agents of CVMCluster bring up the CVM cluster. Only one CVMCluster resource is required in a VCS environment. It is advisable to use the standard configuration procedure for CVM to add the CVMCluster resource. The procedure creates a service group named `cvm` and adds the resources to the configuration.

See "Storage Foundation Cluster File System administrative interface" on page 133.

The following VCS resource types required for SFCFS functionality are:

- CFSfsckd
- CFSMount

CFSfsckd is a mandatory resource type for SFCFS functionality. CFSfsckd agents start the cluster file system check (`fsck` command) daemon, `vxfsckd`, which must be running for a cluster mount to succeed. As with CVMCluster, only one resource instance is required for CFSfsckd. You add these resources using the SFCFS configuration process, which adds the resources to the `cvm` service group.

See "The cfscluster command" on page 134.

Each CVMVolDg resource controls one shared disk group, and one or more shared volumes of that disk group. CVMVolDg resources enable the disk group and set the disk group activation mode. Each CFSMount resource controls the cluster mount of a shared volume on a specified mount point. CFSMount resources also keep track of mount options associated with the mount points.

These resource instances are not added automatically during the SFCFS configuration; you must add them as required using the SFCFS cluster administration commands.

Note: That the CFSMount and CVMVolDg resources are not added to the `cvm` service group; those should be added to a different service group.

See "The cfsmntadm command" on page 134.

Resource and service group dependencies

Dependencies between resources and service groups specify the order in which the resource and service group are brought online and taken offline, which must be done in correct sequence.

The various resources and service groups required for SFCFS must follow these dependency (or link) rules:

- A CFSMount resource must depend on the corresponding CVMVolDg resource
- A service group containing the CVMVolDg resource must depend on the `cvm` service group

Storage Foundation Cluster File System administrative interface

The SFCFS administrative interface provides an easy and convenient way to create resources required for SFCFS with the correct attributes and the correct links between them.

Storage Foundation Cluster File System resource management commands

As many as five VCS agents are required to manage cluster file system functionality. Each of these resources has several attributes and dependencies between them. To make resources easier to manage, five SFCFS administrative commands are provided. It is advisable to use only these commands to manage cluster file systems.

Table 5-1 describes the SFCFS commands.

Table 5-1 SFCFS commands

Commands	Description
cfscluster	Cluster configuration command
cfsmntadm	Adds, deletes, modifies, and sets policy on cluster mounted file systems
cfsdgadm	adds or deletes shared disk groups to and from a cluster configuration
cfsmount	mounts a cluster file system on a shared volume
cfsumount	unmounts a cluster file system on a shared volume

The cfscluster command

The `cfscluster` command is used primarily to configure and unconfigure CVM and SFCFS, and can be run from any node in the cluster. VCS must be started before you can run the `cfscluster config` command. The `cfscluster config` command adds all the resource type definitions and adds resource instances, one each of type CVMCluster and CFSfsckd. The `cfscluster config` command also brings the resources online, and `cfscluster status` can be used to query the status of VCS.

See "Resources and service groups for File System cluster functionality" on page 131.

The `cfscluster unconfig` command takes resources offline (except CFSMount resources) and removes all the resources and service groups that were used to manage the cluster file system.

See the `cfscluster`(1M) manual page.

You must manually take CFSMount resources offline (using the `cfsumount` command) before executing the `cfscluster unconfig` command.

The cfsmntadm command

One CVMVolDg and one CFSMount resource is required to control each cluster mount. You can use the `cfsmntadm add` to add these resources. The `cfsmntadm` command takes mount points, shared volumes, and shared disk groups as arguments. You can optionally specify a service group name. If a service group name is specified, the `cfsmntadm` command creates a new service group (if the service group is not already present) and makes it dependent on the `cvm` service group. If no service group is specified, `cfsmntadm add` creates a new service group for each CFS mount added. The command next adds CVMVolDg to the specified service group and associates it with the specified disk group (if that kind of resource is not already present in the same service group). Subsequently, `cfsmntadm add` adds a CFSMount resource and links it with the CVMVolDg resource, then sets the appropriate values to the resource attributes. It is advisable to add all the mount points (that have their device in the same shared disk group) to the same service group.

See "Resources and service groups for File System cluster functionality" on page 131.

Using `cfsmntadm`, you can also add file system snapshots and Storage Checkpoints; delete, display, and modify resources; and set the primary election policy on a cluster mounted file system.

See the `cfsmntadm`(1M) manual page.

The cfsdgadm command

The `cfsdgadm` command is the administrative interface for shared disk groups. Using `cfsdgadm`, you can add a shared disk group to a cluster configuration, delete a shared disk group, modify the activation mode, or display the shared disk group's configuration information. A shared disk group must already exist before being specified with `cfsdgadm` command.

See the `cfsdgadm`(1M) manual page.

Note: Running the `cfsmntadm delete` command does not delete the disk group name from the `ActivationMode` attribute of the CFSfsckd resource. If no volumes or vsets in the disk group are in the VCS config, you must use the `cfsdgadm delete` to remove the disk group name from the `ActivationMode` attribute.

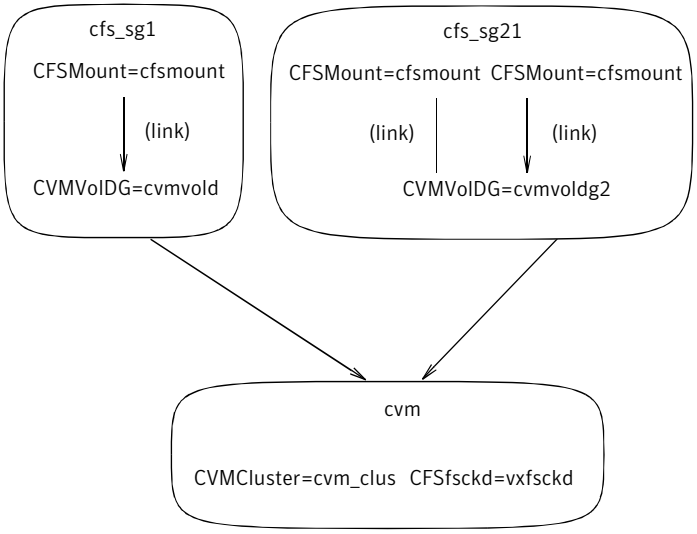
The cfsmount and cfsumount command

The `cfsmount` command mounts a cluster file system on a shared volume on one or more nodes. If no nodes are specified, the cluster file system is mounted on all associated nodes in the cluster. The cluster mount instance for the shared volume must be previously defined by the `cfsmntadm add` command before running `cfsmount`. The `cfsumount` command unmounts one or more shared volumes

See the `cfsmount`(1M) manual page.

Figure 5-1 describes the SFCFS service groups and resource dependencies.

Figure 5-1 SFCFS service groups and resource dependencies



Example main.cf file

This is a sample `main.cf` file:

```
include "types.cf"
include "CFSTypes.cf"
```

```
include "CVMTypes.cf"
cluster cfs_cluster (
    UserNames = { admin = HMNfMHmJNiNNlVNhMK }
    Administrators = { admin }
    CredRenewFrequency = 0
    HacliUserLevel = COMMANDROOT
    CounterInterval = 5
    )
system system01 (
    )
system system02 (
    )
group cvm (
    SystemList = { system01 = 0, system02 = 1 }
    AutoFailOver = 0
    Parallel = 1
    AutoStartList = { system01, system02 }
    )
    CFSfsckd vxfsckd (
        ActivationMode @system01 = { cfsdg = sw }
        ActivationMode @system02 = { cfsdg = sw }
        )
    CVMCluster cvm_clus (
        CVMClustName = cfs_cluster
        CVMNodeId = { system01 = 0, system02 = 1 }
        CVMTransport = gab
        CVMTimeout = 200
        )
    CVMVxconfigd cvm_vxconfigd (
        Critical = 0
        CVMVxconfigdArgs = { syslog }
        )
    cvm_clus requires cvm_vxconfigd
    vxfsckd requires cvm_clus
    // resource dependency tree
    //
    //  group cvm
    //  {
    //  CFSfsckd vxfsckd
    //      {
    //      CVMCluster cvm_clus
    //          {
    //          CVMVxconfigd cvm_vxconfigd
```

```
    //          }
    //      }
    //  }
group vrts_vea_cfs_int_cfsmount1 (
    SystemList = { system01 = 0, system02 = 1 }
    AutoFailOver = 0
    Parallel = 1
    AutoStartList = { system01, system02 }
    )
    CFSMount cfsmount1 (
        Critical = 0
        MountPoint = "/mnt0"
        BlockDevice = "/dev/vx/dsk/cfsdg/vol1"
        NodeList = { system01 , system02 }
        )
    CVMVolDg cvmvoldg1 (
        Critical = 0
        CVMDiskGroup = cfsdg
        CVMVolume = { vol1 }
        CVMActivation @system01 = sw
        CVMActivation @system02 = sw
        CVMVolumeIoTest = { vol}
        )
    requires group cvm online local firm
    cfsmount1 requires cvmvoldg1
    // resource dependency tree
    //
    //  group vrts_vea_cfs_int_cfsmount1
    //  {
    //  CFSMount cfsmount1
    //      {
    //      CVMVolDg cvmvoldg1
    //      }
    //  }
```

Example CVMTypes.cf file

This is a sample of the `CVMTypes.cf` file.

```
type CVMCluster (
    static int NumThreads = 1
    static int OnlineRetryLimit = 2
    static int OnlineTimeout = 400
```

```
    static str ArgList[] = { CVMTransport, CVMClustName, CVMNodeAddr,
 CVMNodeId, PortConfigd, PortKmsgd, CVMTimeout }
    str CVMClustName
    str CVMNodeAddr{}
    str CVMNodeId{}
    str CVMTransport
    int PortConfigd
    int PortKmsgd
    int CVMTimeout
)
type CVMVolDg (
    static keylist RegList = { CVMActivation, CVMVolume }
    static str ArgList[] = { CVMDiskGroup, CVMVolume, CVMActivation }
    static str ArgList[] = { CVMDiskGroup, CVMVolume, CVMActivation,
CVMVolumeIoTest, CVMDGAction }
    str CVMDiskGroup
    str CVMDGAction
    keylist CVMVolume
    str CVMActivation
    keylist CVMVolumeIoTest
    temp int voldg_stat
)
type CVMVxconfigd (
    static int FaultOnMonitorTimeouts = 2
    static int RestartLimit = 5
    static str ArgList[] = { CVMVxconfigdArgs }
    static str Operations = OnOnly
    keylist CVMVxconfigdArgs
)
```

Example CFSTypes.cf file

This is a sample of the `CFSTypes.cf` file.

```
type CFSMount (
    static keylist RegList = { MountOpt, Policy, NodeList, ForceOff,
 SetPrimary }
    static int FaultOnMonitorTimeouts = 1

    static int OnlineWaitLimit = 1
    static str ArgList[] = { MountPoint, BlockDevice, MountOpt, Primary }
    str MountPoint
    str MountType
    str BlockDevice
```

```
    str MountOpt
    keylist NodeList
    keylist Policy
    temp str Primary
    str SetPrimary
    temp str RemountRes
    str ForceOff
)
type CFSfsckd (
    static int RestartLimit = 1
    str ActivationMode{}
)
```

CFSMount agent

The CFSMount agent brings online, takes offline, and monitors a cluster file system mount point. The CFSMount agent executable is `/opt/VRTSvcs/bin/CFSMount/CFSMountAgent`. The type definition is in the `/etc/VRTSvcs/conf/config/CFSTypes.cf` file.

Table 5-2 describes the CFSMount agent entry points.

Table 5-2 CFSMount agent entry points

Entry Points	Description
Online	Mounts a block device or file system snapshot in cluster mode.
Offline	Unmounts the file system (doing a forced unmount if necessary).
Monitor	Determines if the file system is mounted. Checks mount status using the `fsclustadm` command.
Clean	A null operation for a cluster file system mount.
attr_change	Remounts file system with new mount option; sets new primary for file system; sets `fsclustadm` policy on file system.

Table 5-3 describes the CFSMount agent attributes.

Table 5-3 CFSMount agent attributes

Attributes	Type and Dimension	Definition
BlockDevice (required)	string-scalar	Block device for mount point.
MountPoint (required)	string-scalar	Directory for mount point.

Table 5-3 CFSMount agent attributes *(continued)*

Attributes	Type and Dimension	Definition
NodeList (required)	string-keylist	List of nodes on which to mount.
Policy (optional)	string-scalar	Node list for the primary file system selection policy.

MountOpt (optional)	string-scalar	Options for the `mount` command. To create a valid MountOpt attribute string:
		■ Use VxFS type-specific options only.
		■ Do not use the `-o` flag to specify the VxFS-specific options.
		■ Do not use the `-V vxfs` file system type option.
		■ The `cluster` option is not required.
		■ Specify options in a comma-separated list as in these examples:
		`ro`
		`ro,cluster`
		`blkclear,mincache=closesync`

MountType, Primary, SetPrimary, RemountRes, ForceOff (internal)		Not user configured used only by system.

CFSMount type definition

The CFSMount type definition:

```
type CFSMount (
        static keylist RegList = { MountOpt, Policy, NodeList, ForceOff,
SetPrimary }
        static int FaultOnMonitorTimeouts = 1
        static int OnlineWaitLimit = 1
        static str ArgList[] = { MountPoint, BlockDevice, MountOpt,
Primary }
        str MountPoint
        str MountType
        str BlockDevice
        str MountOpt
        keylist NodeList
```

```
        keylist Policy
        temp str Primary
        str SetPrimary
        temp str RemountRes
        str ForceOff
)
```

Sample of CFSMount configuration

This is a sample of CFSMount configuration:

```
CFSMount testdg_test01_fsetpri (
    Critical = 0
    mountPoint = "/mnt1"
    BlockDevice = "/dev/vx/dsk/testdg/test01"
    )
CFSMount testdg_test02_fsetpri (
    Critical = 0
    MountPoint = "/mnt2"
    BlockDevice = "/dev/vx/dsk/testdg/test02"
    MountOpt = "blkclear,mincache=closesync"
    )
```

CFSfsckd agent

The CFSfsckd agent starts, stops, and monitors the `vxfsckd` process. The CFSfsckd agent executable is `/opt/VRTSvcs/bin/CFSfsckd/CFSfsckdAgent`. The type definition is in the `/etc/VRTSvcs/conf/config/CFSTypes.cf` file. The configuration is added to the `main.cf` file after running the `cfscluster config` command.

Table 5-4 describes the CFSfsckd agent entry points.

Table 5-4 CFSfsckd agent entry points

Entry Points	Description
Online	Starts the vxfsckd process.
Offline	Kills the vxfsckd process.
Monitor	Checks whether the vxfsckd process is running.
Clean	A null operation for a cluster file system mount.

There are no required or optional attributes for the CFSfsckd agent.

CFSfsckd type definition

The CFSfsckd type definition:

```
type CFSfsckd (
        static int RestartLimit = 1
        str ActivationMode{}
)
```

Sample of CFSfsckd configuration

This is a sample of CFSfsckd configuration:

```
CFSfsckd vxfsckd (
)
```

CVMCluster agent

The CVMCluster agent controls node membership on the cluster port associated with CVM. The CVMCluster resource requires the CVMVxconfigd resource and must be configured to depend on CVMVxconfigd. The CVMCluster agent executable is `/opt/VRTSvcs/bin/CVMCluster/CVMClusterAgent`. The type definition is in the `/etc/VRTSvcs/conf/config/CVMTypes.cf` file. The configuration is added to the main.cf file after running the `cfscluster config` command.

Table 5-5 describes the CVMCluster agent entry points.

Table 5-5 CVMCluster agent entry points

Entry Points	Description
Online	Joins a node to the CVM cluster port.
Offline	Removes a node from the CVM cluster port.
Monitor	Monitors the node's CVM cluster membership state.
Clean	A null operation for a cluster file system mount.

Table 5-6 describes the CVMCluster agent attributes.

Table 5-6 CVMCluster agent attributes

Attributes	Type and Dimension	Definition
CVMClustName (required)	string-scalar	Name of the cluster.
CVMNodeAddr (required)	string-association	List of host names and IP addresses.
CVMNodeId (required)	string-association	List of host names and LLT node numbers.
CVMTransport (required)	string-association	The CVM transport mode, either `gab` or `udp`. For SFCFS, `gab` is the only valid transport mode.
PortConfigd (required)	integer-scalar	Port number used by CVM for vxconfigd-level communication.
PortKmsgd (required)	integer-scalar	Port number used by CVM for kernel-level communication.
CVMTimeout (required)	integer-scalar	Timeout used by CVM during cluster reconfigurations.

CVMCluster type definition

```
type CVMCluster (
        static int NumThreads = 1
        static int OnlineRetryLimit = 2
        static int OnlineTimeout = 400
        static str ArgList[] = { CVMTransport, CVMClustName, CVMNodeAddr,
CVMNodeId, PortConfigd, PortKmsgd, CVMTimeout }
        str CVMClustName
        str CVMNodeAddr{}
        str CVMNodeId{}
        str CVMTransport
        int PortConfigd
        int PortKmsgd
        int CVMTimeout
)
```

Sample of CVMCluster configuration

This is a sample of CVMCluster configuration:

```
CVMCluster cvm_clus (
        CVMClustName = cfs_cluster
        CVMNodeId = { system01 = 1, system02 = 2 }
        CVMTransport = gab
        CVMTimeout = 200
    )
```

CVMVolDg agent

The CVMVolDg agent brings online, takes offline, and monitors a VxVM shared volume in a disk group. The CVMVolDg agent executable is `/opt/VRTSvcs/bin/CVMVolDg/CVMVolDg`. The type definition is in the `/etc/VRTSvcs/conf/config/CVMTypes.cf` file.

Table 5-7 describes the CVMVolDg agent entry points.

Table 5-7 CVMVolDg agent entry points

Entry Points	Description
Online	Sets the activation mode of the shared disk group and brings volumes online.
Offline	Sets the activation mode of the shared disk group to “off.”
Monitor	Determines whether the disk group and volumes are online.
Clean	A null operation for a cluster file system mount.
attr_changed	Changes the activation mode of the shared disk groups specified.

Table 5-8 describes the CVMVolDg agent attributes.

Table 5-8 CVMVolDg agent attributes

Attributes	Type and Dimension	Definition
CVMDiskGroup (required)	string-scalar	Shared disk group name.
CVMVolume (required)	string-keylist	Shared Volume names. This list is used to check that the volumes are in the correct state before allowing the resource to come online, and that the volumes remain in an enabled state.

Table 5-8 CVMVolDg agent attributes *(continued)*

Attributes	Type and Dimension	Definition
CVMActivation (required)	string-scalar	Activation mode for the disk group. Must be set to shared-write (`sw`). This is a localized attribute.
CVMVolumeIoTest(optional)	string-keylist	List of volumes that will be periodically polled to test availability. The polling is in the form of a 1k read every monitor cycle to a maximum of 10 of the volumes in the list

CVMVolDg type definition

The CVMVolDg type definition:

```
type CVMVolDg (
        static keylist RegList = { CVMActivation, CVMVolume }
        static int OnlineRetryLimit = 2
        static int OnlineTimeout = 400
        static str ArgList[] = { CVMDiskGroup, CVMVolume, CVMActivation,
        CVMVolumeIoTest, CVMDGAction }
        str CVMDiskGroup
        str CVMDGAction
        keylist CVMVolume
        str CVMActivation
        keylist CVMVolume
        str CVMActivation
        keylist CVMVolumeIoTest
 temp int voldg_stat
)
```

Sample of CVMVolDg configuration

This is a sample of CVMVolDg configuration:

```
CVMVolDg cvmvoldg1 (
 Critical = 0
 CVMDiskgroup = testdg
 CVMVolume = { vol1, vol2, mvol1, mvol2, snapvol }
```

```
 CVMVolumeIoTest = { snapvol }
 CVMActivation @system1 = sw
 CVMActivation @system2 = sw
)
```

Chapter 6

Clustered NFS

This chapter includes the following topics:

About Clustered NFS

In previous releases, the SFCFS stack only allowed an active/passive setup for NFS serving due to the complexity of lock reclamation in an active/active configuration. This new Clustered NFS feature gracefully handles failure of any node and reclaim the locks in such a way as to not accidentally lose any existing lock grants without notification.

This release only supports NFS Version 3.

See "How to mount an NFS-exported file system on the NFS clients" on page 165.

Understanding how Clustered NFS works

This Clustered NFS feature allows the same file system mounted across multiple nodes using CFS to be shared over NFS from any combination of those nodes without any loss of functionality during failover. The failover of NFS lock servers

includes all the locks being released by the old node then reclaimed by clients talking to the new node during the grace period.

Basic design

The basic design is to have VCS manage Virtual IP (VIP) resources that can failover between nodes and to add extra code into the steps used to handle these resources to properly handle the NFS level operations. All other involved resources are active on all nodes participating. The lock data, which is saved into a shared area, is managed and used in lock-step with the Virtual IP resources to ensure that all locks are reclaimed properly by clients while preventing any inappropriate locks from being taken at the wrong time. Such interfering locks are prevented through a combination of stopping services and using new features of the lock handling inside the VxFS kernel code.

To communicate with the new code in the kernel, the `fsclustadm` command has been modified to add command line interfaces to the private ioctl calls.

Internal Clustered NFS functionality

This section describes the internal functionality of the triggers and actions scripts that are a part of the Clustered NFS solution.

preonline trigger

The `preonline` script copies the lock state files created by the node status monitor (normally called `statd` or `rpc.statd` daemon) during IP failovers and node failures.

The `preonline` script does the following on IP failover or node failure:

- Finds the IP and the node it was last online on.
- Finds the node on which the IP is next going to go online on.
- Calls `/opt/VRTS/bin/fsclustadm frlpause_enable` and `/opt/VRTS/bin/fsclustadm frlock_pause` to ensure that file system does not give out any new locks during the failover.
- Stops lock and status services on all nodes to prevent granting locks.
- Copies all the files from `/`*`locks`*`/sm/`*`lastonline`*`/sm/` to `/`*`locks`*`/sm/`*`nextonline`*`/sm/ directory.`
  where *locks* is the file system created for storing lock information.
  where *lastonline* is the node on which the VIP resource was previous online.
  where *nextonline* is the node on which the VIP resource will go online next.
- Calls `/opt/VRTS/bin/fsclustadm frlock_resume` to resume giving out locks.

Note: At the end of the preonline trigger all lock services have stopped on all nodes and no NFS locks can be requested until they are restarted.

postonline trigger

The `postonline` script for each VIP does the following during an IP failover or node failure:

- Starts lock services, triggers reclaim, and grace mode on all nodes.
- The restarting of status monitor scans all lock status files in the state directory and contacts all nodes to reclaim their locks. The state files get deleted after they are processed and reclaim messages sent appropriately.
- The lock server goes into grace mode and only allows clients to recover their locks during the grace period. It does not give out any new locks during the grace period.

postoffline trigger

The `postoffline` script does the following on IP failover:

- Calls `/opt/VRTS/bin/fsclustadm frlpause_disable` to reduce the internal usage counter.
- Each call to `/opt/VRTS/bin/fsclustadm frlpause_enable` needs to be matched with a call to `/opt/VRTS/bin/fsclustadm frlpause_disable` as the kernel keeps an internal counter to track the number of IP addresses active on a system. If there are no active IPs on a system, it will be in disabled mode.

Note: This trigger is only called for an administratively initiated failover. An actual system failure and reboot discards the local state being manipulated in this stage. This is the one trigger called on the node that was previously hosting the VIP, while the others are called on the server taking over.

Actions

- On each node, a `/opt/VRTSvcs/bin/IP/actions/nfscfs` file is installed. This file is used to start and stop the NFS locking daemons on a specified node. The `action` script is used instead of using `rsh`, `ssh` or `hacli` for remote command execution from the triggers.

- On each node, a `/opt/VRTSvcs/bin/Application/actions/nfscfsapp` file is installed. This file is used while configuring and unconfiguring the CNFS solution using `cfsshare config` and `cfsshare unconfig` commands.

cfsshare manual page

This Clustered NFS feature adds a new configuration utility called `cfsshare` to the VRTScavf package and several scripts that are added into the VCS configuration to manage parallel NFS server resources. The `cfsshare` command modifies VCS resources that were created by other utilities such as `cfsmntadm`.

See the `cfsshare`(1M) manual page.

Configure and unconfigure Clustered NFS

This section describes how to configure and unconfigure Clustered NFS.

Configure Clustered NFS

`cfsshare config [-n] shared_disk_group shared_volume mount_point`

The CNFS solution requires a shared file system such as `/locks` that is mounted on all cluster nodes.

The local state tracking directory contains a file for each NFS client that has a transaction with the NFS server. The local state tracking directory is:

`/var/statmon/sm`

This creates a symlink to `/locks/sm/nodename/sm` on all the cluster nodes. This allows the lock state files for any cluster node to be accessed by other nodes in the cluster, even when the node is down.

The `-n` option can be used if the user does not want `cfsshare` to create the symlink to `/locks/sm/nodename/sm`. If this option is used, then the user needs to manually create a symlink.

The `config` option adds this shared file system to VCS configuration; it creates the corresponding CFSMount resource in a special parallel service group called cfsnfssg. This also creates an NFS resource in the cfsnfssg service group. In addition to this, a separate resource of the new type ApplicationNone is created to monitor `lockd` and `statd` daemons.

If you run the cfsshare config -n option, you need to perform the following procedure:

1 On each node, create the following directory inside the locks directory:

```
# mkdir -p /locks/sm/nodename/sm
```

2 On each cluster node, create a symlink from `/locks/sm/nodename/sm` to `/var/statmon/sm`.

```
# ln -sf /locks/sm/nodename/sm /var/statmon
```

3 Run the following commands on any one cluster node to set the owner, group, and permissions of `/locks/sm` appropriately:

```
# chown -R daemon:sys /locks/sm
# chmod -R 755 /locks/sm
```

Service group cfsnfssg_dummy

As part of CNFS configuration, a service group called `cfsnfssg_dummy` gets created. This service group is mostly offline.

There is a limit on the number of service groups that can be created in VCS. If this limit is reached, then `cfsnfssg_dummy` serves as the service group in which resources get created during `cfsshare unshare` and `cfsshare delete` operations.

See the *Veritas Cluster Server Administrator's Guide* for information about the GroupLimit attribute.

Unconfigure Clustered NFS

```
cfsshare unconfig
```

This command is used to undo all the steps during the config phase.

Note: This command fails, if there are any CFS file systems still being shared.

Administering Clustered NFS

This section describes Clustered NFS scenarios.

See "Samples for configuring a Clustered NFS" on page 157.

See "Sample main.cf file" on page 161.

Displaying the NFS shared CFS file systems

```
cfsshare display
```

This command displays the CFS file systems that are currently being NFS shared by the cluster nodes.

Sharing a CFS file system previously added to VCS

```
cfsshare share mount_point [share_options]
```

Before running this command, the user should of have run `cfsmntadm` command to add the shared file system to VCS configuration and the `cfsmount` command to mount the shared file system at the *mount_point*. Once these commands have been executed, the CFSMount resource corresponding to the *mount_point* gets created in either a default service group (with a name similar to `vrts_vea_cfs_int_cfsmount`*number*) or in a separate service group, as specified by the user.

The `cfsshare share` command moves the CFSMount resource corresponding to the *mount_point* and the associated CVMVolDg resource to the cfsnfssg service group (that was created using the `config` option). In addition, this command also creates a share resource on top of the CFSMount resource in the same cfsnfssg service group.

Note: VCS does not have the functionality to move resources across service groups. The `cfsshare` command creates new CFSMount and CVMVolDg resources in the cfsnfssg service group and deletes the corresponding resources from the original service group.

The newly created resource names are different from the original resource names.

Unsharing the previous shared CFS file system

```
cfsshare unshare mount_point
```

Before running this command, the user is supposed to have run the `cfsshare share` command.

The `cfsshare unshare` command enables the user to stop sharing the file system mounted at the *mount_point*. This command moves the Share, CFSMount, and CVMVolDg resources corresponding to the *mount_point* from cfsnfssg service group to a newly created service group. The Share resource is taken offline and then deleted.

Note: VCS does not have the functionality to move resources across service groups. The `cfsshare` command creates new CFSMount and CVMVolDg resources in the newly created service group and deletes the corresponding resources from the original service group.

The newly created resource names are different from the original resource names.

Running the `cfsmntadm delete` command does not remove the `ActivationMode` attribute. If no volumes or vsets in the disk group are in the VCS config, you must use the `cfsdgadm delete` to remove this `ActivationMode` attribute.

Adding an NFS shared CFS file system to VCS

```
cfsshare add shared_disk_group shared_volume mount_point[share_options] \
node_name=[mount_options]...

cfsshare add shared_disk_group shared_volume mount_point[share_options] \
all=[mount_options]
```

This command adds the CFS file system to VCS configuration in the cfsnfssg service group, then mounts the file system at the *mount_point* and NFS shares the CFS file system.

Deleting the NFS shared CFS file system from VCS

```
cfsshare delete mount_point
```

Before running this command, the user is supposed to have run the `cfsshare add` command to create the required resources (Share, CFSMount, and CVMVolDg, if needed) in the cfsnfssg service group.

This command unshares the CFS file system mounted at the *mount_point*, unmounts the CFS file system, and removes the CFS file system from VCS configuration.

Adding a Virtual IP address to VCS

```
cfsshare addvip network_interface address netmask
```

This command is used to create a new non-parallel/failover service group that contains a NIC resource for the given network device and an IP resource for the Virtual IP address.

Deleting a Virtual IP address from VCS

```
cfsshare deletevip address
```

This command is used to delete the non-parallel/failover service group corresponding to the Virtual IP address.

Adding an IPv6 Virtual IP address to VCS

```
cfsshare addvipv6 network_interface ipv6_address prefixlen
```

This command is used to create a new non-parallel/failover service group which contains a NIC resource for the given network device and an IP resource for the IPv6 Virtual IP address.

Note: For this release the OS versions specified in the system requirements do not support NFS Version 3 over IPv6.

Deleting an IPv6 Virtual IP address from VCS

```
cfsshare deletevipv6 ipv6_address
```

This command is used to delete the non-parallel/failover service group corresponding to the IPv6 Virtual IP address.

Note: For this release the OS versions specified in the system requirements do not support NFS Version 3 over IPv6.

Changing the share options associated with an NFS share

This section describes how to change the share options associated with an NFS share.

To change the share options associated with an NFS share

1. On any node in the cluster, run `cfsshare unshare` to unshare the file system:

   ```
   # cfsshare unshare mount_point
   ```

2. On any node in the cluster, run `cfsshare share` to share the file system with the desired share options:

   ```
   # cfsshare share mount_point share_options
   ```

Note: The `cfsshare unshare` operation can affect NFS clients that might have mounted the *`mount_point`* file system.

Sharing a file system checkpoint

This section describes how to share a file system checkpoint.

To share a file system checkpoint

1 To add the checkpoint to the VCS configuration, enter:

```
# cfsmntadm add ckpt ckptname mntpt_of_fs mntpt_of_checkpoint \
all=cluster,rw
```

where *`cktpname`* is the checkpoint name.

where *`mntpt_of_fs`* is the name of the mount point of the file system.

where *`mntpt_of_checkpoint`* is the mount point for the checkpoint.

2 To mount the checkpoint, enter:

```
# cfsmount mntpt_of_checkpoint
```

3 Run the `cfsshare share` command to share this checkpoint:

```
# cfsshare share mntpt_of_checkpoint
```

Samples for configuring a Clustered NFS

There are two samples for configuring a Clustered NFS.

Sample 1

This sample is intended to use `cfsshare` command to config and control this feature.

To configure a Clustered NFS (Sample 1)

1 Configure a VCS configuration for CFS/CVM, enter:

```
# cfscluster config
```

2 Configure CNFS components, enter:

```
# cfsshare config shared_disk_group shared_volume \
mount_point
```

For example:

```
# cfsshare config cfsdg vollocks /locks
```

3 Add and mount the CFS file system to the VCS configuration, enter:

```
# cfsmntadm add shared_disk_group shared_volume mount_point \
[service_group] node_name=[mount_options] ...
# cfsmount mount_point
```

For example:

```
# cfsmntadm add vol1 /mnt1 all=
# cfsmount /mnt1
```

4 Share the CFS file system, enter:

```
# cfsshare share mount_point [share_options]
```

For example:

```
# cfsshare share /mnt1 rw
```

5 Add the Virtual IP addresses for users to access the shared CFS file systems, enter:

```
# cfsshare addvip network_interface address netmask
```

For example:

```
# cfsshare addvip en0 10.182.111.161 255.255.240.0
```

6 Delete a previously added Virtual IP address from the configuration, enter:

```
# cfsshare deletevip address
```

For example:

```
# cfsshare deletevip 10.182.111.161
```

7 Unshare CFS file system, enter:

```
# cfsshare unshare mount_point
```

For example:

```
# cfsshare unshare /mnt1
```

8 Unmount and remove the CFS file system from the VCS configuration, enter

```
# cfsumount mount_point
# cfsmntadm delete mount_point
```

For example:

```
# cfsumount /mnt1
# cfsmntadm delete /mnt1
```

9 Unconfigure NFS shared CFS file systems, enter:

```
# cfsshare unconfig
```

Sample 2

This sample is intended to use the `cfsshare` command to add a CFS file system to VCS configuration and mount them. Then share them through NFS, unshare, unmount, and remove the CFS file systems from VCS configuration.

To configure Clustered NFS (Sample 2)

1 Configure a VCS configuration for CFS/CVM, enter:

```
# cfscluster config
```

2 Configure the CNFS components, enter:

```
# cfsshare config shared_disk_group shared_volume mount_point
```

For example:

```
# cfsshare config cfsdg vollocks /locks
```

3 Add and mount the NFS shared CFS file system to the VCS configuration, enter:

```
# cfsshare add shared_disk_group shared_volume mount_point \
[share_options] all=[mount_options]
```

For example:

```
# cfsshare add cfsdg vol1 /mnt1 all=rw
```

4 Share the CFS file system, enter:

```
# cfsshare share mount_point [share_options]
```

For example:

```
# cfsshare share /mnt1 rw
```

5 Add the Virtual IP addresses for users to access the shared CFS file systems, enter:

```
# cfsshare addvip network_interface address netmask
```

For example:

```
# cfsshare addvip en0 10.182.111.161 255.255.240.0
```

6 Delete a previously added Virtual IP address from the configuration, enter:

```
# cfsshare deletevip address
```

For example:

```
# cfsshare deletevip 10.182.111.161
```

7 Unshare, unmount, and remove the CFS file system from the VCS configuration, enter:

```
# cfsshare delete mount_point
```

For example:

```
# cfsshare delete /mnt1
```

8 Unconfigure CNFS components, enter:

```
# cfsshare unconfig
```

Sample main.cf file

This is a sample `main.cf` file.

```
include "types.cf"
include "ApplicationNone.cf"
include "CFSTypes.cf"
include "CVMTypes.cf"

cluster cfs_cluster (
 UserNames = { admin = anoGniNkoJooMwoInl }
 Administrators = { admin }
 HacliUserLevel = COMMANDROOT
 )

system system01 (
 )

system system02 (
 )

group cfsnfssg (
 SystemList = { system01 = 0, system02 = 1 }
 AutoFailOver = 0
 Parallel = 1
 AutoStartList = { system01, system02 }
 )

 ApplicationNone app (
  MonitorProgram = "/opt/VRTSvcs/bin/ApplicationNone/lockdstatdmon"
  )
```

```
CFSMount cfsmount2 (
 Critical = 0
 MountPoint = "/fsqamnt2"
 BlockDevice = "/dev/vx/dsk/dg1/vol2"
 NodeList = { system01, system02 }
 )

CFSMount cfsnfs_locks (
 Critical = 0
 MountPoint = "/locks"
 BlockDevice = "/dev/vx/dsk/dg1/vollocks"
 NodeList = { system01, system02 }
 )

CVMVolDg cvmvoldg1 (
 Critical = 0
 CVMDiskGroup = dg1
 CVMActivation @system01 = sw
 CVMActivation @system02 = sw
 CVMVolume = { vol2 }
 CVMVolumeIoTest = { vol2 }
 )

NFS nfs (
 )

Share share1 (
 PathName = "/fsqamnt2"
 )

requires group cvm online local firm
cfsmount2 requires cvmvoldg1
cfsnfs_locks requires cvmvoldg1
share1 requires cfsmount2
share1 requires nfs


// resource dependency tree
//
// group cfsnfssg
// {
// ApplicationNone app
```

```
 // CFSMount cfsnfs_locks
 //      {
 //      CVMVolDg cvmvoldg1
 //      }
 // Share share1
 //      {
 //      NFS nfs
 //      CFSMount cfsmount2
 //          {
 //          CVMVolDg cvmvoldg1
 //          }
 //      }
 // }


group cfsnfssg_dummy (
 SystemList = { system01 = 0, system02 = 1 }
 AutoFailOver = 0
 Parallel = 1
 AutoStartList = { system01, system02 }
 )

 requires group cvm online local firm


 // resource dependency tree
 //
 // group cfsnfssg_dummy
 // {
 // }


group cvm (
 SystemList = { system01 = 0, system02 = 1 }
 AutoFailOver = 0
 Parallel = 1
 AutoStartList = { system01, system02 }
 )

 CFSfsckd vxfsckd (
  ActivationMode @system01 = { dg1 = sw }
  ActivationMode @system02 = { dg1 = sw }
  )
```

```
CVMCluster cvm_clus (
 CVMClustName = cfs_cluster
 CVMNodeId = { system01 = 0, system02 = 1 }
 CVMTransport = gab
 CVMTimeout = 200
 )

CVMVxconfigd cvm_vxconfigd (
 Critical = 0
 CVMVxconfigdArgs = { syslog }
 )

cvm_clus requires cvm_vxconfigd
vxfsckd requires cvm_clus


// resource dependency tree
//
// group cvm
// {
// CFSfsckd vxfsckd
//     {
//     CVMCluster cvm_clus
//         {
//         CVMVxconfigd cvm_vxconfigd
//         }
//     }
// }


group vip1 (
 SystemList = { system01 = 0, system02 = 1 }
 AutoStartList = { system01, system02 }
 PreOnline @system01 = 1
 PreOnline @system02 = 1
 )

 IP vip1 (
  Device = en0
  Address = "10.182.111.161"
  NetMask = "255.255.252.0"
  )
```

```
NIC nic1 (
 Device = en0
 )

requires group cfsnfssg online local firm
vip1 requires nic1


// resource dependency tree
//
// group vip1
// {
// IP vip1
//     {
//     NIC nic1
//     }
// }
```

How to mount an NFS-exported file system on the NFS clients

This section describes how to mount an NFS-exported file system on the NFS clients.

To mount an NFS-exported file system on the NFS clients

- Run the following command:

  ```
  # mount -V nfs -o vers=3 VIP_address:remote_filesystem mount_point
  ```

Debugging Clustered NFS

The `cfsshare` command logs error messages to the VCS logs in the `/var/VRTSvcs/log` directory. The `fsclustadm frlpause_query` command may be used to display the current local copy of the global state. This may be useful in debugging any issues with this Clustered NFS feature.

Chapter 7

Troubleshooting SFCFS

This chapter includes the following topics:

About troubleshooting SFCFS

SFCFS contains several component products, and as a result can be affected by any issue with component products. The first step in case of trouble should be to identify the source of the problem. It is rare to encounter problems in SFCFS itself; more commonly the problem can be traced to setup issues or problems in component products.

Use the information in this chapter to diagnose the source of problems. Indications may point to SFCFS set up or configuration issues, in which case solutions are provided wherever possible. In cases where indications point to a component product or to Oracle as the source of a problem, it may be necessary to refer to the appropriate documentation to resolve it.

Troubleshooting CFS

This section discusses troubleshooting CFS problems.

Incorrect order in root user's <library> path

An incorrect order in the root user's <library> path can cause the system to hang while changing the primary node in the Cluster File System or the RAC cluster.

If the <library> path of the root user contains an entry pointing to a Cluster File System (CFS) file system before the /usr/lib entry, the system may hang when trying to perform one of the following tasks:

- Changing the primary node for the CFS file system
- Unmounting the CFS files system on the primary node
- Stopping the cluster or the service group on the primary node

This configuration issue occurs primarily in a RAC environment with Oracle binaries installed on a shared CFS file system.

The following is an example of a <library path> that may cause the system to hang:

```
LIBPATH=/app/oracle/orahome/lib:/usr/lib:/usr/ccs/lib
```

In the above example, /app/oracle is a CFS file system, and if the user tries to change the primary node for this file system, the system will hang. The user is still able to ping and telnet to the system, but simple commands such as `ls` will not respond. One of the first steps required during the changing of the primary node is freezing the file system cluster wide, followed by a quick issuing of the `fsck` command to replay the intent log.

Since the initial entry in <library> path is pointing to the frozen file system itself, the `fsck` command goes into a deadlock situation. In fact, all commands (including `ls`) which rely on the <library> path will hang from now on.

The recommended procedure to correct for this problem is as follows: Move any entries pointing to a CFS file system in any user's (especially root) <library> path towards the end of the list after the entry for /usr/lib

Therefore, the above example of a <library path> would be changed to the following:

```
LIBPATH=/usr/lib:/usr/ccs/lib:/app/oracle/orahome/lib
```

Troubleshooting fenced configurations

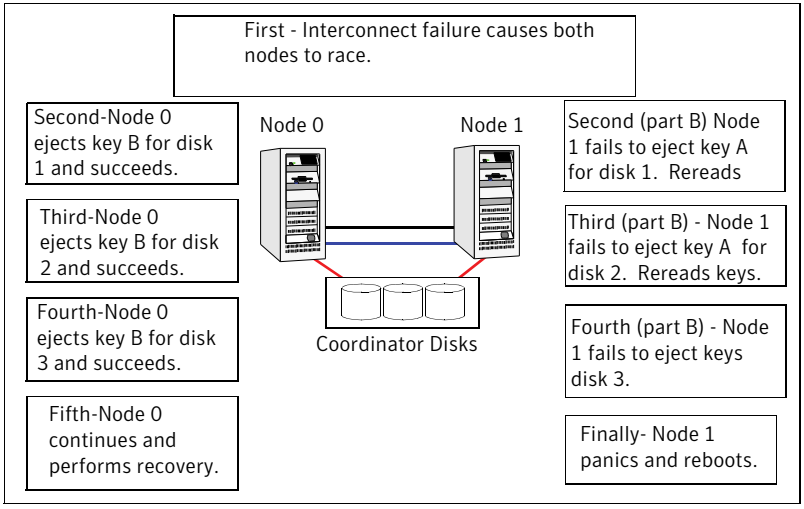
The following information describes network partitioning in a fenced environment.

See the *Veritas Cluster Server User's Guide*.

Example of a preexisting network partition (split-brain)

Figure 7-1 shows a two-node cluster in which the severed cluster interconnect poses a potential split-brain condition.

Figure 7-1 Preexisting network partition (split-brain)



Because the fencing module operates identically on each system, both nodes assume the other is failed, and carry out fencing operations to insure the other node is ejected. The VCS GAB module on each node determines the peer has failed due to loss of heartbeats and passes the membership change to the fencing module.

Each side “races” to gain control of the coordinator disks. Only a registered node can eject the registration of another node, so only one side successfully completes the command on each disk.

The side that successfully ejects the peer from a majority of the coordinator disks wins. The fencing module on the winning side then passes the membership change up to VCS and other higher-level packages registered with the fencing module, allowing VCS to invoke recovery actions. The losing side forces a kernel panic and reboots.

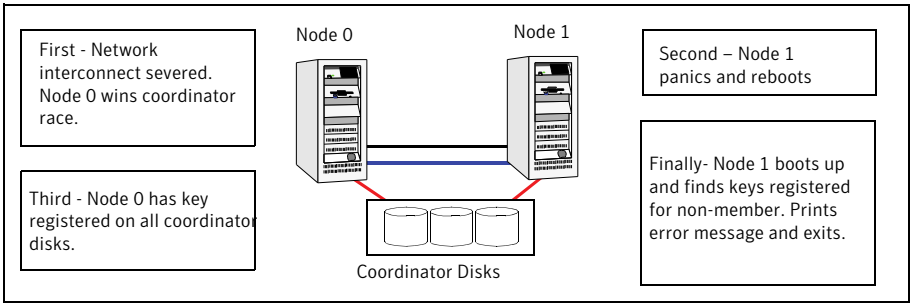
Recovering from a preexisting network partition (split-brain)

The fencing module `vxfen` prevents a node from starting up after a network partition and subsequent panic and reboot of a node.

Example Scenario I

Figure 7-2 scenario could cause similar symptoms on a two-node cluster with one node shut down for maintenance. During the outage, the private interconnect cables are disconnected.

Figure 7-2 Example scenario I



In example scenario I, the following occurs:

- Node 0 wins a coordinator race following to a network failure.
- Node 1 panics and reboots.
- Node 0 has keys registered on the coordinator disks. When Node 1 boots up, it sees the Node 0 keys, but cannot see Node 0 in the current GAB membership. It senses a potential preexisting split brain and causes the vxfen module to print an error message to the console. The vxfen module prevents fencing from starting, which, in turn, prevents VCS from coming online.
  Suggested solution: Shut down Node 1, reconnect the cables, and restart Node 1.

Example Scenario II

Similar to example scenario I, if private interconnect cables are disconnected in a two-node cluster, Node 1 is fenced out of the cluster, panics, and reboots. If before the private interconnect cables are fixed and Node 1 rejoins the cluster, Node 0 reboots and remote (or just reboots). No node can write to the data disks until the private networks are fixed. This is because GAB membership cannot be formed, therefore the cluster cannot be formed.

Suggested solution: Shut down both nodes, reconnect the cables, restart the nodes.

Example Scenario III

Similar to example scenario II, if private interconnect cables are disconnected in a two-node cluster, Node 1 is fenced out of the cluster, panics, and reboots. If

before the private interconnect cables are fixed and Node 1 rejoins the cluster, Node 0 panics due to hardware failure and cannot come back up, Node 1 cannot rejoin.

Suggested solution: Shut down Node 1, reconnect the cables, restart the node. You must then clear the registration of Node 0 from the coordinator disks.

To fix scenario III

1 On Node 1, type the following command:

```
# /opt/VRTSvcs/vxfen/bin/vxfenclearpre
```

2 Restart the node.

Troubleshooting I/O fencing

The following sections discuss troubleshooting the I/O fencing problems. Review the symptoms and recommended solutions.

SCSI reservation errors during bootup

When restarting a node of an SFCFS cluster, SCSI reservation errors may be observed such as:

```
date system name kernel: scsi3 (0,0,6) : RESERVATION CONFLICT
```

This message is printed for each disk that is a member of any shared disk group which is protected by SCSI-3 PR I/O fencing. This message may be safely ignored.

The vxfentsthdw utility fails when SCSI TEST UNIT READY command fails

While running the vxfentsthdw utility, you may see a message that resembles as follows:

```
Issuing SCSI TEST UNIT READY to disk reserved by other node
FAILED.
Contact the storage provider to have the hardware configuration
fixed.
```

The disk array does not support returning success for a `SCSI TEST UNIT READY` command when another host has the disk reserved using SCSI-3 persistent reservations. This happens with the Hitachi Data Systems 99XX arrays if bit 186 of the system mode option is not enabled.

The vxfentsthdw utility fails for Active/Passive arrays when you test disks in raw format

DMP opens the secondary (passive) paths with an exclusive flag in Active/Passive arrays. So, if you test the secondary (passive) raw paths of the disk, the vxfentsthdw command fails due to DMP's exclusive flag.

Recommended action: For Active/Passive arrays when you want to test the disks in raw format, you must use an active enabled path with the vxfentsthdw command. Run the `vxdmpadm getsubpaths dmpnodename=`*`enclosure-based_name`* command to list the active enabled paths.

Node is unable to join cluster while another node is being ejected

A cluster that is currently fencing out (ejecting) a node from the cluster prevents a new node from joining the cluster until the fencing operation is completed. The following are example messages that appear on the console for the new node:

```
...VxFEN ERROR V-11-1-25 ... Unable to join running cluster
since cluster is currently fencing
a node out of the cluster.
```

If you see these messages when the new node is booting, the vxfen startup script on the node makes up to five attempts to join the cluster.

To manually join the node to the cluster when I/O fencing attempts fail

- If the vxfen script fails in the attempts to allow the node to join the cluster, restart vxfen driver with the command:

  ```
  # /etc/init.d/vxfen.rc start
  ```

  If the command fails, restart the new node.

Ignore "DISK OPERATION ERROR" message during restart

When you restart the systems after you install SFCFS and configure I/O fencing, you may safely ignore error messages that resemble the following:

```
.
.
DCB47997   0506172404 T H hdisk13        DISK OPERATION ERROR
DCB47997   0506172404 T H hdisk13        DISK OPERATION ERROR
DCB47997   0506172404 T H hdisk17        DISK OPERATION ERROR
.
```

.
.

All SFCFS nodes come up in ADMIN_WAIT status

When you power back and reboot the SFCFS nodes after a cluster panic occurs from a power outage, I/O fencing may fail to start due to stale I/O fencing keys on the disk. This situation causes the nodes to come up in ADMIN_WAIT status.

To start the nodes that are in ADMIN_WAIT state

1 Verify the status of the nodes in the cluster.

```
# hastatus -summary
-- SYSTEM STATE
-- System State Frozen
A system01 ADMIN_WAIT 0
A system02 ADMIN_WAIT 0
```

2 Clear the stale I/O fencing keys using the `vxfenclearpre` command.

See "About vxfenclearpre utility" on page 84.

3 Start the I/O fencing driver:

```
# /etc/init.d/vxfen.rc start
```

4 Start VCS from any node in the cluster using the `hasys -force` command.

This command forces the VCS engine to use the configuration file from the node you specify in the command. For example:

```
# hasys -force system01
```

System panics to prevent potential data corruption

When a node experiences a split-brain condition and is ejected from the cluster, it panics and displays the following console message:

```
VXFEN:vxfen_plat_panic: Local cluster node ejected from cluster to
prevent potential data corruption.
```

See "How vxfen driver checks for preexisting split-brain condition" on page 174.

How vxfen driver checks for preexisting split-brain condition

The vxfen driver functions to prevent an ejected node from rejoining the cluster after the failure of the private network links and before the private network links are repaired.

For example, suppose the cluster of system 1 and system 2 is functioning normally when the private network links are broken. Also suppose system 1 is the ejected system. When system 1 restarts before the private network links are restored, its membership configuration does not show system 2; however, when it attempts to register with the coordinator disks, it discovers system 2 is registered with them. Given this conflicting information about system 2, system 1 does not join the cluster and returns an error from vxfenconfig that resembles:

```
vxfenconfig: ERROR: There exists the potential for a preexisting
split-brain. The coordinator disks list no nodes which are in
the current membership. However, they also list nodes which are
not in the current membership.

I/O Fencing Disabled!
```

Also, the following information is displayed on the console:

```
<date> <system name> vxfen: WARNING: Potentially a preexisting
<date> <system name> split-brain.
<date> <system name> Dropping out of cluster.
<date> <system name> Refer to user documentation for steps
<date> <system name> required to clear preexisting split-brain.
<date> <system name>
<date> <system name> I/O Fencing DISABLED!
<date> <system name>
<date> <system name> gab: GAB:20032: Port b closed
```

However, the same error can occur when the private network links are working and both systems go down, system 1 restarts, and system 2 fails to come back up. From the view of the cluster from system 1, system 2 may still have the registrations on the coordinator disks.

To resolve actual and apparent potential split-brain conditions

- Depending on the split-brain condition that you encountered, do the following:

Actual potential split-brain condition—system 2 is up and system 1 is ejected	1 Determine if system1 is up or not. 2 If system 1 is up and running, shut it down and repair the private network links to remove the split-brain condition. 3 Restart system 1.
Apparent potential split-brain condition—system 2 is down and system 1 is ejected	1 Physically verify that system 2 is down. Verify the systems currently registered with the coordinator disks. Use the following command: `# vxfenadm -g all -f /etc/vxfentab` The output of this command identifies the keys registered with the coordinator disks. 2 Clear the keys on the coordinator disks as well as the data disks using the `vxfenclearpre` command. See "Clearing keys after split-brain using vxfenclearpre command" on page 176. 3 Make any necessary repairs to system 2. 4 Restart system 2.

Cluster ID on the I/O fencing key of coordinator disk does not match the local cluster's ID

If you accidentally assign coordinator disks of a cluster to another cluster, then the fencing driver displays an error message similar to the following when you start I/O fencing:

```
000068 06:37:33 2bdd5845  0 ... 3066 0 VXFEN WARNING V-11-1-56
 Coordinator disk has key with cluster id 48813
 which does not match local cluster id 57069
```

The warning implies that the local cluster with the cluster ID 57069 has keys. However, the disk also has keys for cluster with ID 48813 which indicates that nodes from the cluster with cluster id 48813 potentially use the same coordinator disk.

You can run the following commands to verify whether these disks are used by another cluster. Run the following commands on one of the nodes in the local cluster. For example, on system01:

```
system01> # lltstat -C
57069

system01> # cat /etc/vxfentab
/dev/vx/rdmp/disk_7
/dev/vx/rdmp/disk_8
/dev/vx/rdmp/disk_9

system01> # vxfenadm -s /dev/vx/rdmp/disk_7
Reading SCSI Registration Keys...
Device Name: /dev/vx/rdmp/disk_7
Total Number Of Keys: 1
key[0]:
 [Character Format]: VFBEAD00
 [Node Format]: Cluster ID: 48813 Node ID: 0 Node Name: unknown
```

Recommended action: You must use a unique set of coordinator disks for each cluster. If the other cluster does not use these coordinator disks, then clear the keys using the vxfenclearpre command before you use them as coordinator disks in the local cluster.

See "About vxfenclearpre utility" on page 84.

Clearing keys after split-brain using vxfenclearpre command

If you have encountered a preexisting split-brain condition, use the vxfenclearpre command to remove SCSI-3 registrations and reservations on the coordinator disks as well as on the data disks in all shared disk groups.

See "About vxfenclearpre utility" on page 84.

Registered keys are lost on the coordinator disks

If the coordinator disks lose the keys that are registered, the cluster might panic when a cluster reconfiguration occurs.

To refresh the missing keys

- Use the vxfenswap utility to replace the coordinator disks with the same disks. The vxfenswap utility registers the missing keys during the disk replacement.

  See "Refreshing lost keys on coordinator disks" on page 95.

Replacing defective disks when the cluster is offline

If the disk becomes defective or inoperable and you want to switch to a new diskgroup in a cluster that is offline, then perform the following procedure.

In a cluster that is online, you can replace the disks using the vxfenswap utility.

See "About vxfenswap utility" on page 86.

Review the following information to replace coordinator disk in the coordinator disk group, or to destroy a coordinator disk group.

Note the following about the procedure:

- When you add a disk, add the disk to the disk group vxfencoorddg and retest the group for support of SCSI-3 persistent reservations.
- You can destroy the coordinator disk group such that no registration keys remain on the disks. The disks can then be used elsewhere.

To replace a disk in the coordinator disk group when the cluster is offline

1. Log in as superuser on one of the cluster nodes.
2. If VCS is running, shut it down:

   ```
   # hastop -all
   ```

   Make sure that the port h is closed on all the nodes. Run the following command to verify that the port h is closed:

   ```
   # gabconfig -a
   ```

3. Stop I/O fencing on each node:

   ```
   # /etc/init.d/vxfen.rc stop
   ```

   This removes any registration keys on the disks.
4. Import the coordinator disk group. The file /etc/vxfendg includes the name of the disk group (typically, vxfencoorddg) that contains the coordinator disks, so use the command:

   ```
   # vxdg -tfC import `cat /etc/vxfendg`
   ```

   where:

   -t specifies that the disk group is imported only until the node restarts.

   -f specifies that the import is to be done forcibly, which is necessary if one or more disks is not accessible.

   -C specifies that any import locks are removed.

5 To remove disks from the disk group, use the VxVM disk administrator utility, vxdiskadm.

You may also destroy the existing coordinator disk group. For example:

- Verify whether the coordinator attribute is set to on.

```
# vxdg list vxfencoorddg |  grep flags: | grep coordinator
```

- If the coordinator attribute value is set to on, you must turn off this attribute for the coordinator disk group.

```
# vxdg -g vxfencoorddg set coordinator=off
```

- Destroy the disk group.

```
# vxdg destroy vxfencoorddg
```

6 Add the new disk to the node, initialize it as a VxVM disk, and add it to the vxfencoorddg disk group.

See the *Storage Foundation Cluster File System Installation Guide* for detailed instructions.

7 Test the recreated disk group for SCSI-3 persistent reservations compliance.

See "Testing the coordinator disk group using vxfentsthdw -c option" on page 74.

8 After replacing disks in a coordinator disk group, deport the disk group:

```
# vxdg deport `cat /etc/vxfendg`
```

9 On each node, start the I/O fencing driver:

```
# /etc/init.d/vxfen.rc start
```

10 Verify that the I/O fencing module has started and is enabled.

```
# gabconfig -a
```

Make sure that port b membership exists in the output for all nodes in the cluster.

```
# vxfenadm -d
```

Make sure that I/O fencing mode is not disabled in the output.

11 If necessary, restart VCS on each node:

```
# hastart
```

The vxfenswap utility faults when echo or cat is used in .bashrc file

The vxfenswap utility faults when you use echo or cat to print messages in the .bashrc file for the nodes.

To recover the vxfenswap utility fault

- Verify whether the rcp or scp functions properly.

  If the vxfenswap operation is unsuccessful, use the `vxfenswap -cancel` command if required to roll back any changes that the utility made.

  See "About vxfenswap utility" on page 86.

Troubleshooting on the CP server

All the CP server operations and messages are logged in the /var/VRTScps/log directory in a detailed and easy to read format. The entries are sorted by date and time. The logs can be used for troubleshooting purposes or to review for any possible security issue on the single node VCS or SFHA cluster hosting the CP server.

The following files contain logs and text files that may be useful in understanding and troubleshooting a CP server:

- /var/VRTScps/log/cpserver_[ABC].log
- /var/VRTSat/vrtsat_broker.txt (Security related)

If the vxcpserv process fails on the CP server, then review the following diagnostic files:

- /var/VRTScps/diag/FFDC_CPS_<pid>_vxcpserv.log
- /var/VRTScps/diag/stack_<pid>_vxcpserv.txt

Note: If the vxcpserv process fails on the CP server, these files are present in addition to a core file. VCS restarts vxcpserv process automatically in such situations.

CP server service group issues

If you cannot bring up the CPSSG service group after the CP server configuration, verify that the CPSSG service group and its resources are valid and properly configured in the VCS configuration.

Check the VCS engine log to see if any of the CPSSG service group resources are FAULTED. The engine log is located in the following directory:

/var/VRTSvcs/log/engine_[ABC].log

The resources that are configured under the CPSSG service groups are displayed in the following figures:

- CPSSG group and dependency figure for CP server hosted on a single node VCS cluster:
- CPSSG group and dependency figure for CP server hosted on an SFHA cluster:

Note: For information about general VCS troubleshooting procedures, refer to the Veritas™ Cluster Server User's Guide, Version 5.1.

Troubleshooting server-based I/O fencing on the SFCFS cluster

The file /var/VRTSvcs/log/vxfen/vxfend_[ABC].log contains logs and text files that may be useful in understanding and/or troubleshooting fencing-related issues on a SFCFS cluster node.

Issues during server-based fencing start up on SFCFS cluster node

The following issues may occur during fencing start up on the SFCFS cluster node:

- `cpsadm` command on the SFCFS cluster gives connection error
- Authentication failure
- Authorization failure
- Preexisting split-brain

Testing the connectivity of the CP server

The connectivity of the CP server can be tested using the `cpsadm` command. The following `cpsadm` command tests whether a CP server is up and running at a process level:

```
# cpsadm -s cp_server -a ping_cps
```

where *cp_server* is the virtual IP address or virtual hostname on which the CP server is listening.

Issuing the command on the SFCFS cluster nodes requires the environment variables CPS_USERNAME and CPS_DOMAINTYPE to be set.

cpsadm command on the SFCFS cluster node gives connection error

If you receive a connection error message after issuing the `cpsadm` command on the SFCFS cluster, perform the following actions:

- Ensure that the CP server is reachable from all the SFCFS cluster nodes.
- Check that the correct CP server virtual IP/virtual hostname and port number are being used by the SFCFS cluster nodes.
  Check the /etc/vxfenmode file.
- Ensure that the running CP server is using the same virtual IP/virtual hostname and port number.

Authentication failure

If secure communication has been configured between the CP server and the SFCFS cluster nodes, authentication failure can occur due to the following causes:

- Symantec Product Authentication Services is not properly configured on the CP server and/or the SFCFS cluster.
- The CP server and the SFCFS cluster nodes use the same root broker but the certificate hash of the root broker is not same on the SFCFS cluster and the CP server. Run the following command on both the CP server and the SFCFS cluster to see the certificate hash:

  ```
  # cpsat showalltrustedcreds
  ```

- The CP server and the SFCFS cluster nodes use different root brokers, and trust is not established between the authentication brokers:
  See "About secure communication between the SFCFS cluster and CP server" on page 42.

- The CP server and SFCFS cluster do not have the same security setting.
  In order to configure secure communication, both the CP server and the SFCFS cluster must have same security setting.
  In order to have the same security setting, the security parameter must have same value in the /etc/vxcps.conf file on CP server and in the /etc/vxfenmode file on the SFCFS cluster nodes.

Authorization failure

Authorization failure occurs when the CP server's SFCFS cluster nodes or users are not added in the CP server configuration. Therefore, fencing on the SFCFS cluster node is not allowed to access the CP server and register itself on the CP server. Fencing fails to come up if it fails to register with a majority of the coordination points. To resolve this issue, add the SFCFS cluster node and user in the CP server configuration and restart fencing. Refer to the following section:

Preexisting split-brain

To illustrate preexisting split-brain, assume there are three CP servers acting as coordination points. One of the three CP servers then becomes inaccessible. While in this state, also one client node leaves the cluster. When the inaccessible CP server restarts, it has a stale registration from the node which left the SFCFS cluster. In this case, no new nodes can join the cluster. Each node that attempts to join the cluster gets a list of registrations from the CP server. One CP server includes an extra registration (of the node which left earlier). This makes the joiner node conclude that there exists a preexisting split-brain between the joiner node and the node which is represented by the stale registration. The situation is similar to that of preexisting split-brain, with coordinator disks, where the problem is solved by the administrator running the `vxfenclearpre` command. A similar solution is required using the `cpsadm` command.

The following `cpsadm` command can be used to clear a registration on a CP server:

```
# cpsadm -s cp_server -a unreg_node -c cluster_name -n nodeid
```

where *cp_server* is the virtual IP address or virtual hostname on which the CP server is listening, *cluster_name* is the VCS name for the SFCFS cluster, and *nodeid* specifies the node id of SFCFS cluster node.

After removing all stale registrations, the joiner node will be able to join the cluster.

Issues during online migration of coordination points

During online migration of coordination points using the vxfenswap utility, the operation is automatically rolled back if a failure is encountered during validation of coordination points from all the cluster nodes.

Validation failure of the new set of coordination points can occur in the following circumstances:

- The /etc/vxfenmode file is not updated on all the SFCFS cluster nodes, because new coordination points on the node were being picked up from an old /etc/vxfenmode file.
- The coordination points listed in the /etc/vxfenmode file on the different SFCFS cluster nodes are not the same. If different coordination points are listed in the /etc/vxfenmode file on the cluster nodes, then the operation fails due to failure during the coordination point snapshot check.
- There is no network connectivity from one or more SFCFS cluster nodes to the CP server(s).
- The cluster or nodes or users for the SFCFS cluster nodes have not been added on the new CP servers, thereby causing authorization failure.

Vxfen service group activity after issuing the vxfenswap command

After issuing the `vxfenswap` command, the Coordination Point agent reads the details of coordination points from the vxfenconfig -l output and starts monitoring the registrations on them.

During vxfenswap, when the vxfenmode file is being changed by the user, the Coordination Point agent does not move to FAULTED state but continues monitoring the old set of coordination points.

As long as the changes to vxfenmode file are not committed or the new set of coordination points are not re-elected in vxfenconfig -l output, the Coordination Point agent continues monitoring the old set of coordination points it read from vxfenconfig -l output in every monitor cycle.

The status of the Coordination Point agent (either ONLINE or FAULTED) depends upon the accessibility of the coordination points, the registrations on these coordination points, and the fault tolerance value.

When the changes to vxfenmode file are committed and reflected in the vxfenconfig -l output, then the Coordination Point agent reads the new set of coordination points and proceeds to monitor them in its new monitor cycle.

Troubleshooting server-based I/O fencing in mixed mode

The following procedure can be use to troubleshoot a mixed I/O fencing configuration (configuration using both coordinator disks and CP server for I/O fencing). This procedure involves using the following commands to obtain I/O fencing information:

- To obtain I/O fencing cluster information on the coordinator disks, run the following command on one of the cluster nodes:

  ```
  # vxfenadm -s diskname
  ```

  Any keys other than the valid keys used by the cluster nodes that appear in the command output are spurious keys.

- To obtain I/O fencing cluster information on the CP server, run the following command on one of the cluster nodes:

  ```
  # cpsadm -s cp_server -a list_membership -c cluster_name
  ```

  where *cp server* is the virtual IP address or virtual hostname on which the CP server is listening, and *cluster name* is the VCS name for the SFCFS cluster.
  Nodes which are not in GAB membership, but registered with CP server indicate a pre-existing network partition.
  Note that when running this command on the SFCFS cluster nodes, you need to first export the CPS_USERNAME and CPS_DOMAINTYPE variables.
  The CPS_USERNAME value is the user name which is added for this node on the CP server.

- To obtain the user name, run the following command on the CP server:

  ```
  # cpsadm -s cp_server -a list_users
  ```

  where *cp server* is the virtual IP address or virtual hostname on which the CP server is listening.
  The CPS_DOMAINTYPE value is vx.
  The following are export variable command examples:

```
# export CPS_USERNAME=_HA_VCS_test-system@HA_SERVICES@test-system.symantec.com
```

```
# export CPS_DOMAINTYPE=vx
```

Once a pre-existing network partition is detected using the above commands, all spurious keys on the coordinator disks or CP server must be removed by the administrator.

Troubleshooting mixed I/O fencing configuration (coordinator disks and CP server)

1 Review the current I/O fencing configuration by accessing and viewing the information in the vxfenmode file.

Enter the following command on one of the SFCFS cluster nodes:

```
# cat /etc/vxfenmode

vxfen_mode=customized
vxfen_mechanism=cps
scsi3_disk_policy=dmp
security=0
cps1=[10.140.94.101]:14250
vxfendg=vxfencoorddg
```

2 Review the I/O fencing cluster information.

Enter the `vxfenadm -d` command on one of the cluster nodes:

```
# vxfenadm -d

I/O Fencing Cluster Information:
================================

 Fencing Protocol Version: 201
 Fencing Mode: Customized
 Fencing Mechanism: cps
 Cluster Members:

        * 0 (system01)
          1 (system02)

 RFSM State Information:
        node   0 in state  8 (running)
        node   1 in state  8 (running)
```

3 Review the SCSI registration keys for the coordinator disks used in the I/O fencing configuration.

Enter the `vxfenadm -s` command on each of the SFCFS cluster nodes.

```
# vxfenadm -s /dev/vx/rdmp/3pardata0_190
```

```
Device Name: /dev/vx/rdmp/3pardata0_190
Total Number Of Keys: 2
key[0]:
         [Numeric Format]:  86,70,66,69,65,68,48,48
         [Character Format]: VFBEAD00
         [Node Format]: Cluster ID: 57069 Node ID: 0   Node Name: system01
key[1]:
         [Numeric Format]:  86,70,66,69,65,68,48,49
         [Character Format]: VFBEAD01
*        [Node Format]: Cluster ID: 57069 Node ID: 1   Node Name: system02
```

```
# vxfenadm -s /dev/vx/rdmp/3pardata0_191
```

```
Device Name: /dev/vx/rdmp/3pardata0_191
Total Number Of Keys: 2
key[0]:
         [Numeric Format]:  86,70,66,69,65,68,48,48
         [Character Format]: VFBEAD00
         [Node Format]: Cluster ID: 57069 Node ID: 0   Node Name: system01
key[1]:
         [Numeric Format]:  86,70,66,69,65,68,48,49
         [Character Format]: VFBEAD01
*        [Node Format]: Cluster ID: 57069 Node ID: 1   Node Name: system02
```

4 Review the CP server information about the cluster nodes.

On the CPS server, run the `cpsadm list nodes` command to review a list of nodes in the cluster.

The command syntax is as follows:

```
# cpsadm -s cp_server -a list_nodes
```

where *cp server* is the virtual IP address or virtual hostname on which the CP server is listening.

For example:

```
# /opt/VRTS/bin/cpsadm -s 10.140.94.101 -a list_nodes

ClusName        UUID                                    Hostname(Node ID) Registered
gl-rh2  {25aeb8c6-1dd2-11b2-95b5-a82227078d73}  node_101(0)         0
gl-rh2  {25aeb8c6-1dd2-11b2-95b5-a82227078d73}  node_102(1)         0
cpstest {a0cf10e8-1dd1-11b2-87dc-080020c8fa36}  node_220(0)         0
cpstest {a0cf10e8-1dd1-11b2-87dc-080020c8fa36}  node_240(1)         0
ictwo   {f766448a-1dd1-11b2-be46-5d1da09d0bb6}  node_330(0)         0
ictwo   {f766448a-1dd1-11b2-be46-5d1da09d0bb6}  sassette(1)         0
fencing {e5288862-1dd1-11b2-bc59-0021281194de}  CDC-SFLAB-CD-01(0)  0
fencing {e5288862-1dd1-11b2-bc59-0021281194de}  CDC-SFLAB-CD-02(1)  0
gl-su2  {8f0a63f4-1dd2-11b2-8258-d1bcc1356043}  gl-win03(0)         0
gl-su2  {8f0a63f4-1dd2-11b2-8258-d1bcc1356043}  gl-win04(1)         0
gl-su1  {2d2d172e-1dd2-11b2-bc31-045b4f6a9562}  gl-win01(0)         0
gl-su1  {2d2d172e-1dd2-11b2-bc31-045b4f6a9562}  gl-win02(1)         0
gl-ax4  {c17cf9fa-1dd1-11b2-a6f5-6dbd1c4b5676}  gl-ax06(0)          0
gl-ax4  {c17cf9fa-1dd1-11b2-a6f5-6dbd1c4b5676}  gl-ax07(1)          0
gl-ss2  {da2be862-1dd1-11b2-9fb9-0003bac43ced}  system01(0)         1
gl-ss2  {da2be862-1dd1-11b2-9fb9-0003bac43ced}  system02(1)         1
```

5 Review the CP server list membership.

On the CP server, run the following command to review the list membership. The command syntax is as follows:

```
# cpsadm -s cp_server -a list_membership -c cluster_name
```

where *cp_server* is the virtual IP address or virtual hostname on which the CP server is listening, and *cluster_name* is the VCS name for the SFCFS cluster.

For example:

```
# cpsadm -s 10.140.94.101 -a list_membership -c gl-ss2

List of registered nodes: 0 1
```

Checking keys on coordination points when vxfen_mechanism value is set to cps

When I/O fencing is configured in customized mode and the vxfen_mechanism value is set to cps, the recommended way of reading keys from the coordination points (coordinator disks and CP servers) is as follows:

- For coordinator disks, the disks can be put in a file and then information about them supplied to the `vxfenadm` command.
  For example:

  ```
  # vxfenadm -s all -f file_name
  ```

- For CP servers, the `cpsadm` command can be used to obtain the membership of the SFCFS cluster.
  For example:

  ```
  # cpsadm -s cp_server -a list_membership -c cluster_name
  ```

Where *cp_server* is the virtual IP address or virtual hostname on which CP server is configured, and *cluster_name* is the VCS name for the SFCFS cluster.

Troubleshooting CVM

This section discusses troubleshooting CVM problems.

CVM group is not online after adding a node to the cluster

The possible causes for the CVM group being offline after adding a node to the cluster are as follows:

- The cssd resource is configured as a critical resource in the cvm group.
- Other resources configured in the cvm group as critical resources are not online.

To resolve the issue if cssd is configured as a critical resource

1. Log onto one of the nodes in the existing cluster as the root user.
2. Configure the cssd resource as a non-critical resource in the cvm group:

```
# haconf -makerw
# hares -modify cssd Critical 0
# haconf -dump -makero
```

To resolve the issue if other resources in the group are not online

1. Log onto one of the nodes in the existing cluster as the root user.
2. Bring the resource online:

```
# hares -online resource_name -sys system_name
```

3. Verify the status of the resource:

```
# hastatus -resource  resource_name
```

4. If the resource is not online, configure the resource as a non-critical resource:

```
# haconf -makerw
# hares -modify resource_name Critical 0
# haconf -dump -makero
```

Shared disk group cannot be imported

If you see a message resembling:

```
vxvm:vxconfigd:ERROR:vold_pgr_register(/dev/vx/rdmp/disk_name):
local_node_id<0
Please make sure that CVM and vxfen are configured
and operating correctly
```

First, make sure that CVM is running. You can see the CVM nodes in the cluster by running the vxclustadm nidmap command.

```
# vxclustadm nidmap
Name          CVM Nid     CM Nid      State
system01         1           0           Joined: Master
system02         0           1           Joined: Slave
```

This above ouput shows that CVM is healthy, with system system01 as the CVM master. If CVM is functioning correctly, then the output above is displayed when CVM cannot retrieve the node ID of the local system from the `vxfen` driver. This usually happens when port b is not configured.

To verify vxfen driver is configured

- Check the GAB ports with the command:

  ```
  # gabconfig -a
  ```

Port b must exist on the local system.

Error importing shared disk groups

The following message may appear when importing shared disk group:

```
VxVM vxdg ERROR V-5-1-587 Disk group disk group name: import
failed: No valid disk found containing disk group
```

You may need to remove keys written to the disk.

For information about removing keys written to the disk:

See “Removing preexisting keys” on page 84.

Unable to start CVM

If you cannot start CVM, check the consistency between the `/etc/llthosts` and `main.cf` files for node IDs.

You may need to remove keys written to the disk.

For information about removing keys written to the disk:

See “Removing preexisting keys” on page 84.

CVMVolDg not online even though CVMCluster is online

When the CVMCluster resource goes online, then all shared disk groups that have the auto-import flag set are automatically imported. If the disk group import fails for some reason, the CVMVolDg resources fault. Clearing and taking the CVMVolDg type resources offline does not resolve the problem.

To resolve the resource issue

1. Fix the problem causing the import of the shared disk group to fail.
2. Offline the cvm group containing the resource of type CVMVolDg as well as the service group containing the CVMCluster resource type.
3. Bring the cvm group containing the CVMCluster resource online.
4. Bring the cvm group containing the CVMVolDg resource online.

Troubleshooting interconnects

This section discusses troubleshooting interconnect problems.

Restoring communication between host and disks after cable disconnection

If a fiber cable is inadvertently disconnected between the host and a disk, you can restore communication between the host and the disk without restarting.

You can also use this procedure if you are adding disks or disk arrays to an active/online SFCFS configuration.

To restore lost cable communication between host and disk

1. Reconnect the cable.
2. On all nodes, use the `cfgmgr` command to scan for new disks.

   It may take a few minutes before the host is capable of seeing the disk.
3. On all nodes, issue the following command to rescan the disks:

   ```
   # vxdisk scandisks
   ```
4. On the master node, reattach the disks to the disk group they were in and retain the same media name:

   ```
   # vxreattach
   ```

   This may take some time. For more details, see `vxreattach` (1M) manual page.

Appendix A

Creating a starter database

This appendix includes the following topics:

Creating a database for Oracle 10g or 11g

Create a database tablespace for Oracle 10g or 11g using one of the two options:

- Option 1: on shared raw VxVM volumes
- Option 2: on cluster file system (CFS)

Before you begin, take note of the following prerequisites:

- CRS daemons must be running. To verify the status of CRS, enter:

  ```
  # $CRS_HOME/bin/crs_stat
  ```

- Use the `ping` command to verify that all private IP addresses on each node are up.

Creating database tablespace on shared raw VxVM volumes (option 1)

This section describes how to create database tablespace on shared raw VxVM volumes (option 1).

To create database tablespace on shared raw VxVM volumes (option 1)

1 Log in as `root`.

2 Find out the CVM master, enter the following command on any node:

```
# vxdctl -c mode
mode: enabled: cluster active - MASTER
master: system01
```

The above sample output indicates that `system01` is the CVM master.

3 On the CVM master, find out the spare disks that can be used for creating shared disk group for Oracle database tablespaces, enter:

```
# vxdisk -o alldgs list
DEVICE TYPE          DISK   GROUP     STATUS
hdisk1 auto:none     -      -         online invalid
hdisk2 auto:none     -      -         online invalid
hdisk3 auto:cdsdisk  -      tempdg    online shared
hdisk4 auto:cfsdisk  -      ocrvotedg online shared
hdisk5 auto:cdsdisk  -      -         online shared
hdisk6 auto:cdsdisk  -      -         online shared
```

The above sample output indicates that shared disks `hdisk5` and `hdisk6` are free and can be used for Oracle database tablespaces.

4 On the CVM master node, create a shared disk group:

```
# vxdg -s init oradatadg hdisk5 hdisk6
```

5 Create a volume in the shared group for each of the required tablespaces.

See the *Oracle* documentation specific to the Oracle database release to determine the tablespace requirements.

For example, enter:

```
# vxassist -g oradatadg make VRT_system01 1000M
# vxassist -g oradatadg make VRT_system02 10M
.
.
.
```

6 Define the access mode and permissions for the volumes storing the Oracle data. For each volume listed in `$ORACLE_HOME/raw_config`, use the `vxedit` command:

```
# vxedit -g disk_group set group=group user=user mode=660 volume
```

See the `vxedit`(1M) manual page.

For example, enter:

```
# vxedit -g oradatadg set group=oinstall user=oracle mode=660 \
VRT_system01
```

In this example, `VRT_system01` is the name of one of the volumes. Repeat the command to define access mode and permissions for each volume in the `oradatadg`.

7 Create the database.

See the *Oracle* documentation.

Creating database tablespace on CFS (option 2)

This section describes how to create database tablespace on CFS (option 2). If you plan to use a cluster file system to store the Oracle database, use the following procedure to create the file system.

To creating database tablespace on CFS (option 2)

1 Log in as `root`.

2 Find out the CVM master, enter the following command on any node:

```
# vxdctl -c mode
mode: enabled: cluster active - MASTER
master: system01
```

The above sample output indicates that `system01` is the CVM master.

3 On the CVM master, find out the spare disks that can be used for creating shared disk group for Oracle database tablespaces, enter:

```
# vxdisk -o alldgs list
DEVICE TYPE          DISK   GROUP     STATUS
hdisk1 auto:none     -      -         online invalid
hdisk2 auto:none     -      -         online invalid
hdisk3 auto:cdsdisk  -      tempdg    online shared
hdisk4 auto:cfsdisk  -      ocrvotedg online shared
hdisk5 auto:cdsdisk  -      -         online shared
hdisk6 auto:cdsdisk  -      -         online shared
```

The above sample output indicates that shared disks `hdisk5` and `hdisk6` are free and can be used for Oracle database tablespaces.

4 Create a shared disk group. For example, enter:

```
# vxdg -s init oradatadg hdisk5
```

5 Create a single shared volume that is large enough to contain a file system for all tablespaces.

See the *Oracle* documentation specific to the Oracle database release for tablespace sizes.

Assuming 6.8GB are required for the tablespaces, enter:

```
# vxassist -g oradatadg make oradatavol 6800M
```

6 Create a VxFS file system in this volume, enter:

```
# mkfs -V vxfs /dev/vx/rdsk/oradatadg/oradatavol
```

7 Create a mount point for the shared file system, enter:

```
# mkdir /oradata
```

8 From the same node, mount the file system, enter:

```
# mount -V vxfs -o cluster /dev/vx/dsk/oradatadg/oradatavol \
/oradata
```

9 Set `oracle` as the owner of the file system, and set `775` as the permissions:

```
# chown oracle:oinstall /oradata
# chmod 775 /oradata
```

10 On the other node(s), complete steps 7 through 8.

11 Create the Oracle database.

See the *Oracle* documentation.

Glossary

ACL (access control list)	The information that identifies specific users or groups and their access privileges for a particular file or directory.
agent	A process that manages predefined Veritas Cluster Server (VCS) resource types. Agents bring resources online, take resources offline, and monitor resources to report any state changes to VCS. When an agent is started, it obtains configuration information from VCS and periodically monitors the resources and updates VCS with the resource status.
allocation unit	A group of consecutive blocks on a file system that contain resource summaries, free resource maps, and data blocks. Allocation units also contain copies of the super-block.
API	Application Programming Interface.
asynchronous writes	A delayed write in which the data is written to a page in the system's page cache, but is not written to disk before the write returns to the caller. This improves performance, but carries the risk of data loss if the system crashes before the data is flushed to disk.
atomic operation	An operation that either succeeds completely or fails and leaves everything as it was before the operation was started. If the operation succeeds, all aspects of the operation take effect at once and the intermediate states of change are invisible. If any aspect of the operation fails, then the operation aborts without leaving partial changes.
BLI (block-level incremental) backup	A Veritas backup capability that does not store and retrieve entire files. Instead, only the data blocks that have changed since the previous backup are backed up.
boot disk	A disk that is used for the purpose of booting a system.
boot disk group	A private disk group that contains the disks from which the system may be booted.
buffered I/O	A mode of I/O operation (where I/O is any operation, program, or device that transfers data to or from a computer) that first transfers data into the Operating System buffer cache.
bootdg	A reserved disk group name that is an alias for the name of the boot disk group.
cluster mounted file system	A shared file system that enables multiple hosts to mount and perform file operations on the same file. A cluster mount requires a shared storage device that can be accessed by other cluster mounts of the same file system. Writes to the

	shared device can be done concurrently from any host on which the cluster file system is mounted. To be a cluster mount, a file system must be mounted using the `mount -o cluster` option.
Cluster Services	The group atomic broadcast (GAB) module in the SFCFS stack provides cluster membership services to the file system. LLT provides kernel-to-kernel communications and monitors network communications.
contiguous file	A file in which data blocks are physically adjacent on the underlying media.
CVM (cluster volume manager)	The cluster functionality of Veritas Volume Manager.
CVM Master	The cluster volume manager has a master node that records changes to the volume configuration.
data block	A block that contains the actual data belonging to files and directories.
data synchronous writes	A form of synchronous I/O that writes the file data to disk before the write returns, but only marks the inode for later update. If the file size changes, the inode will be written before the write returns. In this mode, the file data is guaranteed to be on the disk before the write returns, but the inode modification times may be lost if the system crashes.
defragmentation	The process of reorganizing data on disk by making file data blocks physically adjacent to reduce access times.
direct extent	An extent that is referenced directly by an inode.
direct I/O	An unbuffered form of I/O that bypasses the kernel's buffering of data. With direct I/O, the file system transfers data directly between the disk and the user-supplied buffer.
discovered direct I/O	Discovered Direct I/O behavior is similar to direct I/O and has the same alignment constraints, except writes that allocate storage or extend the file size do not require writing the inode changes before returning to the application.
encapsulation	A process that converts existing partitions on a specified disk to volumes. If any partitions contain file systems, `/etc/filesystems` entries are modified so that the file systems are mounted on volumes instead. Encapsulation is not applicable on some systems.
extent	A group of contiguous file system data blocks treated as a single unit. An extent is defined by the address of the starting block and a length.
extent attribute	A policy that determines how a file allocates extents.
external quotas file	A quotas file (named `quotas`) must exist in the root directory of a file system for quota-related commands to work.

file system block The fundamental minimum size of allocation in a file system. This is equivalent to the fragment size on some UNIX file systems.

fileset A collection of files within a file system.

fixed extent size An extent attribute used to override the default allocation policy of the file system and set all allocations for a file to a specific fixed size.

fragmentation The on-going process on an active file system in which the file system is spread further and further along the disk, leaving unused gaps or fragments between areas that are in use. This leads to degraded performance because the file system has fewer options when assigning a file to an extent.

GB (gigabyte) 2^{30} bytes or 1024 megabytes.

hard limit The hard limit is an absolute limit on system resources for individual users for file and data block usage on a file system.

heartbeat Heartbeat messages are sent over the private link to obtain information on cluster membership changes. If a node does not send a heartbeat for 16 seconds, it is removed from the membership. The command lltconfig is used for information on the various heartbeat parameters. The low latency transport (LLT) module provides communication services across the cluster.

indirect address extent An extent that contains references to other extents, as opposed to file data itself. A single indirect address extent references indirect data extents. A double indirect address extent references single indirect address extents.

indirect data extent An extent that contains file data and is referenced via an indirect address extent.

inode A unique identifier for each file within a file system that contains the data and metadata associated with that file.

inode allocation unit A group of consecutive blocks containing inode allocation information for a given fileset. This information is in the form of a resource summary and a free inode map.

intent logging A method of recording pending changes to the file system structure. These changes are recorded in a circular intent log file.

internal quotas file VxFS maintains an internal quotas file for its internal usage. The internal quotas file maintains counts of blocks and indices used by each user.

KB (kilobyte) 2^{10} bytes or 1024 bytes.

large file A file larger than one terabyte. VxFS supports files up to 8 exabytes in size.

large file system A file system larger than one terabytes. VxFS supports file systems up to 8 exabytes in size.

latency For file systems, this typically refers to the amount of time it takes a given file system operation to return to the user.

local mounted file system A file system mounted on a single host. The single host mediates all file system writes to storage from other clients. To be a local mount, a file system cannot be mounted using the `mount -o cluster` option.

metadata Structural data describing the attributes of files on a disk.

MB (megabyte) 2^{20} bytes or 1024 kilobytes.

mirror A duplicate copy of a volume and the data therein (in the form of an ordered collection of subdisks). Each mirror is one copy of the volume with which the mirror is associated.

multi-volume file system A single file system that has been created over multiple volumes, with each volume having its own properties.

MVS (multivolume support)

node One of the hosts in a cluster.

node abort A situation where a node leaves a cluster (on an emergency basis) without attempting to stop ongoing operations.

node join The process through which a node joins a cluster and gains access to shared disks.

OLT (object location table) The information needed to locate important file system structural elements. The OLT is written to a fixed location on the underlying media (or disk).

page file A fixed-size block of virtual address space that can be mapped onto any of the physical addresses available on a system.

preallocation A method of allowing an application to guarantee that a specified amount of space is available for a file, even if the file system is otherwise out of space.

primary fileset The files that are visible and accessible to the user.

quotas Quota limits on system resources for individual users for file and data block usage on a file system.

quotas file The quotas commands read and write the external quotas file to get or change usage limits. When quotas are turned on, the quota limits are copied from the external quotas file to the internal quotas file.

reservation An extent attribute used to preallocate space for a file.

root disk group A special private disk group that always exists on the system. The root disk group is named `rootdg`.

SFCFS (Storage Foundation Cluster File System)

SFCFS Primary	There is a primary node for each file system in the cluster responsible for updating metadata in the file system.
shared disk group	A disk group in which the disks are shared by multiple hosts (also referred to as a cluster-shareable disk group).
shared volume	A volume that belongs to a shared disk group and is open on more than one node at the same time.
snapshot file system	An exact copy of a mounted file system at a specific point in time. Used to do online backups.
snapped file system	A file system whose exact image has been used to create a snapshot file system.
soft limit	The soft limit is lower than a hard limit. The soft limit can be exceeded for a limited time. There are separate time limits for files and blocks.
Storage Checkpoint	A facility that provides a consistent and stable view of a file system or database image and keeps track of modified data blocks since the last Storage Checkpoint.
structural fileset	The files that define the structure of the file system. These files are not visible or accessible to the user.
super-block	A block containing critical information about the file system such as the file system type, layout, and size. The VxFS super-block is always located 8192 bytes from the beginning of the file system and is 8192 bytes long.
synchronous writes	A form of synchronous I/O that writes the file data to disk, updates the inode times, and writes the updated inode to disk. When the write returns to the caller, both the data and the inode have been written to disk.
TB (terabyte)	2^{40} bytes or 1024 gigabytes.
transaction	Updates to the file system structure that are grouped together to ensure they are all completed.
throughput	For file systems, this typically refers to the number of I/O operations in a given unit of time.
unbuffered I/O	I/O that bypasses the kernel cache to increase I/O performance. This is similar to direct I/O, except when a file is extended; for direct I/O, the inode is written to disk synchronously, for unbuffered I/O, the inode update is delayed.
VCS (Veritas Cluster Server)	
volume	A virtual disk which represents an addressable range of disk blocks used by applications such as file systems or databases.
volume set	A container for multiple different volumes. Each volume can have its own geometry.
vxfs	The Veritas file system type. Used as a parameter in some commands.

VxFS The Veritas File System.

VxVM The Veritas Volume Manager.

Index

D

E

F

Q

R

S

T

U

V