

Veritas Storage Foundation™ for Cluster File System Administrator's Guide

Linux

5.0 Maintenance Pack 3



Veritas Storage Foundation™ for Cluster File System Administrator's Guide

The software described in this book is furnished under a license agreement and may be used only in accordance with the terms of the agreement.

Product Version: 5.0 MP3

Documentation version: 5.0MP3.0

Legal Notice

Copyright © 2008 Symantec Corporation. All rights reserved.

Symantec, the Symantec Logo, Veritas, and Storage Foundation, are trademarks or registered trademarks of Symantec Corporation or its affiliates in the U.S. and other countries. Other names may be trademarks of their respective owners.

This Symantec product may contain third party software for which Symantec is required to provide attribution to the third party ("Third Party Programs"). Some of the Third Party Programs are available under open source or free software licenses. The License Agreement accompanying the Software does not alter any rights or obligations you may have under those open source or free software licenses. Please see the Third Party Legal Notice Appendix to this Documentation or TPIP ReadMe File accompanying this Symantec product for more information on the Third Party Programs.

The product described in this document is distributed under licenses restricting its use, copying, distribution, and decompilation/reverse engineering. No part of this document may be reproduced in any form by any means without prior written authorization of Symantec Corporation and its licensors, if any.

THE DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. SYMANTEC CORPORATION SHALL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS DOCUMENTATION. THE INFORMATION CONTAINED IN THIS DOCUMENTATION IS SUBJECT TO CHANGE WITHOUT NOTICE.

The Licensed Software and Documentation are deemed to be commercial computer software as defined in FAR 12.212 and subject to restricted rights as defined in FAR Section 52.227-19 "Commercial Computer Software - Restricted Rights" and DFARS 227.7202, "Rights in Commercial Computer Software or Commercial Computer Software Documentation", as applicable, and any successor regulations. Any use, modification, reproduction release, performance, display or disclosure of the Licensed Software and Documentation by the U.S. Government shall be solely in accordance with the terms of this Agreement.

Symantec Corporation
20330 Stevens Creek Blvd.
Cupertino, CA 95014

<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/techsupp/

Contacting Technical Support

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

www.symantec.com/business/support/index.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 technical support Web page at the following URL:

www.symantec.com/techsupp/

Customer service

Customer service information is available at the following URL:

www.symantec.com/techsupp/

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

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	contractsadmin@symantec.com
Europe, Middle-East, and Africa	semea@symantec.com
North America and Latin America	supportolutions@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

Technical Support	4	
Chapter 1	Technical overview	13
	Storage Foundation Cluster File System architecture	13
	About the symmetric architecture	13
	About Storage Foundation Cluster File System primary/secondary failover	14
	About single-host file system semantics using Group Lock Manager	14
	About Veritas File System features supported in cluster file systems	14
	Veritas File System features in cluster file systems	15
	Veritas File System features not in cluster file systems	16
	Storage Foundation Cluster File System benefits and applications	17
	How Storage Foundation Cluster File System works	17
	When to use Storage Foundation Cluster File System	18
Chapter 2	Storage Foundation Cluster File System architecture	21
	Storage Foundation Cluster File System architecture overview	21
	About Veritas Cluster Server architecture	21
	Veritas Volume Manager cluster functionality	22
	When the Storage Foundation Cluster File System primary fails	23
	About Storage Foundation Cluster File System and the Group Lock Manager	23
	About asymmetric mounts	23
	Parallel I/O	24
	Storage Foundation Cluster File System namespace	25
	Storage Foundation Cluster File System backup strategies	25
	Synchronize time on Cluster File Systems	26
	Distribute a load on a cluster	27
	File system tuneables	27
	Split-brain and jeopardy handling	27
	Fencing	28

Single network link and reliability	28
I/O error handling policy	29
About Veritas Volume Manager cluster functionality	30
Shared disk groups overview	31

Chapter 3

Storage Foundation Cluster File System administration	37
About Storage Foundation Cluster File System administration	37
Veritas Cluster Server overview	38
About Group Membership and Atomic Broadcast	39
About Low Latency Transport	39
Veritas Volume Manger cluster functionality overview	39
Storage Foundation Cluster File System overview	40
Number of parallel fsck threads to run during recovery is tunable	40
Cluster and shared mounts	40
Determining or giving primaryship	40
Asymmetric mounts	41
Storage Foundation Cluster File System and Veritas Volume Manager cluster functionality agents	42
Storage Foundation Cluster File System administration commands	42
Storage Foundation Cluster File System commands	42
mount and fsclusteradm commands	43
Time synchronization for Cluster File Systems	44
Growing a Storage Foundation Cluster File System	44
The fstab file	44
Distribute the load on a Cluster	45
GUIs	45
Snapshots on Storage Foundation Cluster File System	45
Cluster snapshot characteristics	45
Performance considerations	46
Creating a snapshot on a Storage Foundation Cluster File System	46
Adding a node to a cluster	47
Configuring Storage Foundation Cluster File System and Cluster Volume Manager agents on the new node	50
Removing a node from a cluster	53

Chapter 4	Storage Foundation Cluster File System for Oracle RAC	57
	Storage Foundation Cluster File System for Oracle RAC	57
	About Storage Foundation Cluster File System for Oracle RAC	57
	How SFCFS for Oracle RAC works (high-level perspective)	58
	Component products and processes of SFCFS for Oracle RAC	60
	Communication infrastructure	60
	Data flow	60
	Communication requirements	61
	Cluster Volume Manager	62
	Cluster Volume Manager architecture	62
	Storage Foundation Cluster File System	63
	Storage Foundation Cluster File System architecture	63
	Oracle Disk Manager	64
	Oracle Disk Manager architecture	64
	Oracle Disk Manager performance enhancements	64
	Veritas Cluster Server functionality for SFCFS for Oracle RAC	65
	Veritas Cluster Server architecture	65
	Cluster configuration files	65
	Installing and configuring SFCFS for Oracle RAC	66
	Verifying the Oracle Cluster Registry and Vote-disk shared volumes	72
	Installing Oracle Clusterware and Database software	72
	Relinking Oracle Database with Veritas Extension for Oracle Disk Manager	72
	Upgrading Oracle Database	73
Chapter 5	Fencing administration	81
	I/O fencing	81
	Data disks	81
	Coordinator Disks	82
	Before you configure coordinator disks	82
	Adding or removing coordinator disks	85
	Verifying fenced configurations	86
	Disabling I/O fencing	87
	How I/O fencing works during different events	88
	Troubleshooting fenced configurations	90
	Example of a preexisting network partition (split-brain)	90

	Recovering from a preexisting network partition (split-brain)	91
Chapter 6	Using Veritas Extension for Oracle Disk Manager	93
	About Oracle Disk Manager	93
	How Oracle Disk Manager improves database performance	95
	About Oracle Disk Manager and Storage Foundation Cluster Files System	96
	About Oracle Disk Manager and Oracle Managed Files	97
	How Oracle Disk Manager works with Oracle Managed Files	97
	Setting up Veritas Extension for Oracle Disk Manager	99
	Configuring Veritas Extension for Oracle Disk Manager	100
	How to prepare existing database storage for Oracle Disk Manager	102
	Verifying that Oracle Disk Manager is configured	102
	Disabling the Oracle Disk Manager feature	103
Chapter 7	Veritas Volume Manager cluster functionality administration	105
	About Veritas Volume Manager cluster functionality administration	105
	Overview of Cluster Volume Management	106
	Private and shared disk Groups	108
	Activating modes of shared disk groups	109
	Connectivity policy of shared disk groups	111
	Limitations of shared disk groups	113
Chapter 8	Agents for Storage Foundation Cluster File System	115
	About agents for Storage Foundation Cluster File System	115
	Storage Foundation Cluster File System agents	116
	Veritas Cluster Server cluster components	116
	Resources	116
	Attributes	117
	Service groups	117
	Modifying the agents and their resources	117
	Resources and service groups for File System cluster functionality	117
	Resource and service group dependencies	118

Storage Foundation Cluster File System administrative interface	119
Storage Foundation Cluster File System resource management commands	119
Example main.cf file	122
Example CVMTypes.cf file	124
Example CFSTypes.cf file	125
CFSMount agent	126
CFSMount type definition	127
Sample of CFSMount configuration	127
CFSfsckd agent	128
CFSfsckd type definition	128
Sample of CFSfsckd configuration	129
CVMCluster agent	129
CVMCluster type definition	130
Sample of CVMCluster configuration	130
CVMVolDg agent	131
CVMVolDg type definition	131
Sample of CVMVolDg configuration	132
Appendix A Creating a starter database	133
Creating a database for Oracle 10g or 11g	133
Creating database tablespace on shared raw VxVM volumes (option 1)	133
Creating database tablespace on CFS (option 2)	135
Glossary	139
Index	145

Technical overview

This chapter includes the following topics:

- [Storage Foundation Cluster File System architecture](#)
- [About Veritas File System features supported in cluster file systems](#)
- [Storage Foundation Cluster File System benefits and applications](#)

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
- About Storage Foundation Cluster File System primary/secondary failover
- About single-host file system semantics using Group Lock Manager

About the symmetric architecture

Beginning with SFCFS 5.0, 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.

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 Cluster File System 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. Quick I/O is licensable only through Veritas Database Editions products.
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 can mount cluster file systems to NFS.
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 <code>mmap()</code> function is supported on SFCFS. See the <code>mmap(2)</code> manual page.
Disk layout versions	SFCFS supports only disk layout Version 6 and 7. Cluster mounted file systems can be upgraded, a local mounted file system can be upgraded, unmounted, and mounted again as part of a cluster. Use the <code>fstyp -v special_device</code> command to ascertain the disk layout version of a VxFS file system. Use the <code>vxupgrade</code> 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 <code>F_GETLK</code> command, if there is a process holding a conflicting lock, the <code>l_pid</code> field returns the process ID of the process holding the conflicting lock. The nodeid-to-node name translation can be done by examining the <code>/etc/llthosts</code> file or with the <code>fsclustadm</code> command. Mandatory locking, and deadlock detection supported by traditional <code>fcntl</code> locks, are not supported on SFCFS. See the <code>fcntl(2)</code> 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 <code>mknod</code> command cannot be used to create devices on a cluster mounted file system.
Cache advisories	Cache advisories are set with the <code>mount</code> 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 <code>atime</code> 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 and resizing and reorganization (defragmentation) 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.

Storage Foundation Cluster File System architecture

This chapter includes the following topics:

- [Storage Foundation Cluster File System architecture overview](#)
- [When the Storage Foundation Cluster File System primary fails](#)
- [About Veritas Volume Manager cluster functionality](#)

Storage Foundation Cluster File System architecture overview

SFCFS includes Veritas Cluster Server (VCS) and Veritas Volume Manager (VxVM). The Veritas Cluster Server (VCS) provides the communication, configuration, and membership services required to create a cluster. VCS is the first component installed and configured to set up a cluster file system.

About Veritas Cluster Server architecture

The Group Membership and Atomic Broadcast (GAB) and Low Latency Transport (LLT) are VCS-specific protocols implemented directly on an 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 file `/etc/gabtab` is used to configure GAB. Configuration is done with the `gabconfig` command. For example, the `-n` option of the command specifies the number of nodes in the cluster. GAB is configured automatically when you run

the VCS 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 files `/etc/llthosts` and `/etc/llttab` 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 events such as state 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 file `/etc/llttab` contains information you provide during installation. You may also have to reconfigure LLT when adding nodes to a cluster.

See the `llttab(4)` manual page.

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

Each component in SFCFS registers with a 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 f	Cluster File system membership
port h	Veritas Cluster Server communication between GAB and High Availability Daemon (HAD)
port u	Temporarily used by CVM
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)

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.

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 mode 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.

See [“Distribute a load on a cluster”](#) on page 27.

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, system caches must be kept coherent, and each must instantly reflect updates to cached data, regardless of the node from which they originate.

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. File systems in a cluster can be mounted with different read/write options. These are called asymmetric mounts.

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

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.

[About asymmetric mounts](#) describes the first column in the following table shows the mode in which the primary is mounted:

Figure 2-1 Primary and secondary mounts

		Secondary		
		<code>ro</code>	<code>rw</code>	<code>ro, crw</code>
Primary	<code>ro</code>	X		
	<code>rw</code>		X	X
	<code>ro, crw</code>		X	X

The check marks indicate the mode secondary mounts can use.

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.

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 feature.

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 an I/O request. 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 a different node 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.

If the file size can be predetermined, the file can be preallocated 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.

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.

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.
- Multiple snapshots of a cluster file system can be mounted on the same or different cluster node.
- A snapshot is accessible only on the node mounting a 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 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. However, a snapshot is not affected if a 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.

In addition to file-level frozen images, 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*.

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 creation (`ctime`) and modification (`mtime`) may not be consistent with the sequence in which operations actually happened.

Distribute a load on a cluster

You can use the `fsclustadm` to designate a SFCFS primary. The `fsclustadm setprimary` mount point can be used to change the primary. This change to the primary is not persistent across unmounts or reboots. The change is in effect as long as one or more nodes in the cluster have the file system mounted. The primary selection policy can also be defined by a VCS attribute associated with the SFCFS mount resource.

For example, if you have eight file systems and four nodes, designating two file systems per node as the primary is beneficial. The first node that mounts a file system becomes the primary for that file system.

File system tuneables

Tuneable parameters are updated at the time of mount using the `tunefstab` file or `vxtunefs` command. The file system `tunefs` 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 `tunefs` parameters of the primary node are used. The `tunefstab` file on the node is used if this is the first node to mount the file system. Symantec recommends that this file be identical on each node.

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. Membership change also occurs when all private-link cluster interconnects fail simultaneously, or when a node is unable to respond to heartbeat messages. 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 employs jeopardy to prevent data corruption following a split-brain.

In certain following scenarios, the possibility of data corruption remains:

- 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, potential split-brain conditions are safeguarded by jeopardy handling. If any cluster node fails following a jeopardy state notification, the cluster file system mounted on the failed nodes is disabled. If a node fails after the jeopardy state notification, all cluster nodes also leave the shared disk group membership.

Recover from jeopardy

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

Fencing

With the use of I/O enabled fencing, all remaining cases with the potential to corrupt data (for which jeopardy handling cannot protect) are addressed.

See [“I/O fencing”](#) on page 81.

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 [“I/O fencing”](#) on page 81.

Configuring low priority 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 tag
```

To remove a link

- To remove a link, type the following command:

```
# lltconfig -u tag
```

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

Note: LLT clients do not recognize the difference unless only one link is available and GAB declares jeopardy.

I/O error handling policy

I/O errors can occur for several reasons, including failures of Fibre Channel link, 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.

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.

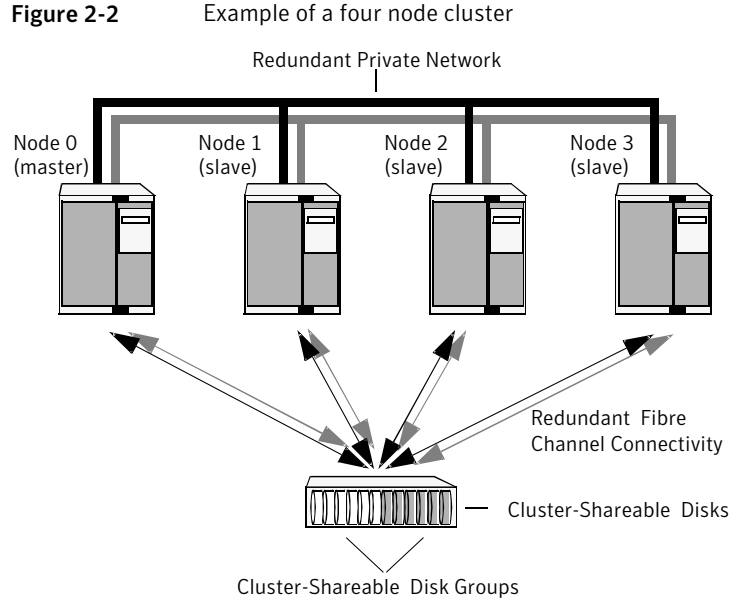
Note: RAID-5 volumes are not supported on a shared disk group.

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.

[Figure 2-2](#) 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.



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 2-2 describes the disk group types.

Table 2-2 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 `vxchg` 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.

See “[Activation modes of shared disk groups](#)” on page 33.

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 `shared-write`.

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 2-3](#) describes activation modes for shared disk groups.

Table 2-3 Activation modes for shared disk groups

Activation mode	Description
<code>exclusivewrite</code> (ew)	The node has exclusive write access to the disk group. No other node can activate the disk group for write access.

Table 2-3 Activation modes for shared disk groups (*continued*)

Activation mode	Description
<code>readonly (ro)</code>	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.
<code>sharedread (sr)</code>	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.
<code>sharedwrite (sw)</code>	The node has write access to the disk group.
<code>off</code>	The node has neither read nor write access to the disk group. Query operations on the disk group are permitted.

[Table 2-4](#) summarizes the allowed and conflicting activation modes for shared disk groups.

Table 2-4 Allowed and conflicting activation modes

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

To place activation modes under user control

- Create a defaults file `/etc/default/vxdg` 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.

If the defaults 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 diskgroup` 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 2-5](#) describes the policies for resolving this type of discrepancy.

Table 2-5 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 administration

This chapter includes the following topics:

- [About Storage Foundation Cluster File System administration](#)
- [Veritas Cluster Server overview](#)
- [Veritas Volume Manger cluster functionality overview](#)
- [Storage Foundation Cluster File System overview](#)
- [Storage Foundation Cluster File System administration commands](#)
- [Snapshots on Storage Foundation Cluster File System](#)
- [Adding a node to a cluster](#)
- [Configuring Storage Foundation Cluster File System and Cluster Volume Manager agents on the new node](#)
- [Removing a node from a cluster](#)

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.

Note: To install and administer cluster file systems, you should have a working knowledge of VxVM. To install and administer application failover functionality, you should have a working knowledge of VCS. For more information on these products, refer to the Veritas Volume Manager and Veritas Cluster Server documentation. The user guides for Volume Manager are available in the `/opt/VRTSvmddoc` directory after you install the Storage Foundation packages. The user guides for VCS are available in the `/opt/VRTSvcddoc` directory after you install the Storage Foundation Cluster File System HA packages.

Topics in this chapter include:

- VCS Overview
- CVM Overview
- SFCFS Overview
- SFCFS Administration
- Snapshots on SFCFS

Veritas Cluster Server overview

The Veritas Cluster Server (VCS) provides the communication, configuration, and membership services required to create a cluster. VCS is the first component installed and configured to set up a cluster file system.

Group membership and atomic broadcast (GAB) and Low Latency Transport (LLT) are VCS-specific protocols implemented directly on an Ethernet data link or on a Fibre Channel fabric. Both GAB and LLT run over redundant data links that connect all the servers in a cluster. VCS requires redundant cluster communication links to minimize the possibility of cluster failure due to the failure of a single communication link.

About Group Membership and Atomic Broadcast

Group Membership and Atomic Broadcast (GAB) provides membership and messaging service, both for the cluster as a whole and for groups of applications running it. The GAB membership service provides orderly startup and shutdown of a cluster.

The file `/etc/gabtab` is used to configure GAB. Configuration is done with the `gabconfig` command. For example, the `-n` option of the command specifies the number of nodes in the cluster. GAB is configured automatically when you run the VCS installation script, but you may have to reconfigure GAB when you add a node to a cluster.

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

About Low Latency Transport

Low Latency Transport (LLT) provides kernel-to-kernel communications and monitors network communications. The LLT files `/etc/llthosts` and `/etc/llttab` can be 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 events such as state 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 file `/etc/llttab` contains information derived from what you input during installation. You may also have to reconfigure LLT when you add a node to a cluster.

See the `llttab(4)` manual page.

Veritas Volume Manger cluster functionality overview

The cluster functionality (CVM) of the Veritas Volume Manager allows multiple hosts to concurrently access and manage a given set of logical devices under VxVM control. A VxVM cluster is a set of hosts sharing a set of devices; each host is a node in the cluster. The nodes are connected across a network. If one node fails, other nodes can still 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.

See [“About Veritas Volume Manager cluster functionality administration”](#) on page 105.

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

Storage Foundation Cluster File System overview

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). So it is possible to 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 45.

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

Number of parallel fsck threads to run during recovery is tunable

In previous releases the number of parallel fsck thread that could be active during recovery was set to 4. If a node failed over 12 file systems, log replay for the 12 file systems will not complete at the same time. The maximum parallelism was restricted to 4. The number was set to 4 since parallel replay of a large number of file systems would put memory pressure on smaller systems with less memory. However, on larger systems the restriction of 4 parallel processes replaying is not necessary. In this release this value is either dependent on the number of CPUs in the system, but you can get the value within given limits.

To set the number of parallel fsck thread

- On all nodes in the cluster, edit the `/opt/VRTSvcs/bin/CFSfsckd/CFSfsckd.env` file and set `FCKD_OPTS="-n N"`, where *N* is the number of parallel fsck threads that the system runs.

Cluster and shared mounts

A VxFS file system that is mounted with the `mount -o cluster` option is called 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. File systems in a cluster can be mounted with different read-write options. These are called asymmetric mounts.

Determining or giving primaryship

Both primary and secondary nodes handle their metadata intent logging for the cluster file system. 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:

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

Asymmetric mounts

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

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`). The first column in the following table shows the mode in which the primary is mounted. The check marks indicate the mode secondary mounts can use.

[Asymmetric mounts](#) describes the first column in the following table shows the mode in which the primary is mounted:

Figure 3-1 Primary and secondary mounts

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

Only mounting the primary with `-o cluster,ro` prevents the secondaries from mounting in a different mode, that is, read-write mode. Note that `rw` implies read-write capability throughout the cluster.

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

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 115.

Storage Foundation Cluster File System administration commands

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

This section includes the following topics:

- Storage Foundation Cluster File System commands
- Veritas File System Commands

Storage Foundation Cluster File System commands

[Table 3-1](#) describes the SFCFS commands.

Table 3-1 SFCFS commands

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

Table 3-1 SFCFS commands (*continued*)

Commands	Description
<code>cfsumount</code>	unmounts a cluster file system on a shared volume

mount and fsclusteradm commands

The `mount` and `fsclusteradm` 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.

fsclusteradm

The `fsclusteradm` command reports various attributes of a cluster file system. Using `fsclusteradm` 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 `fsclusteradm` 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 `fsclusteradm(1M)` manual page.

fsadm

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

See the `fsadm(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 command:

```
# 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 -t vxfs -b newsize -r device_name mount_point
```

The `fstab` file

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

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

Use the Veritas Enterprise Administrator (VEA) for various VxFS functions such as making and mounting file systems, on both local and cluster file systems.

With SFCFS HA, you can use the VCS Cluster Manager GUI to configure and monitor SFCFS. The VCS GUI provides log files for debugging LLT and GAB events.

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 File System 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 -t vxfs /dev/vx/rdisk/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 Mount the snapshot, type the following command:

```
# cfsmount /mnt1snap
```

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

```
# cfsunmount /mnt1snap
```

Adding a node to a cluster

If you want to add a new node to a multi-node cluster, first prepare the new system hardware. Physically connect the new node to the cluster using private networks and attach to any shared storage. Then install the required OS software.

Note: For a cluster to work correctly, all nodes must have the same time. If you are not running the Network Time Protocol (NTP) daemon, make sure the time on all the nodes comprising your cluster is synchronized.

To add a node to a cluster

1 Log into the new node as superuser.

2 After inserting the software disc, type the following command:

```
# mount -o ro /dev/cdrom /mnt/cdrom
```

3 Add `/opt/VRTS/bin` to your `PATH` and `/opt/VRTS/man` to your `MANPATH` environment variables.

4 Change to the SFCFS directory.

```
# cd sfcfs
```

5 Run the `installsfcfs` script with `-installonly` option to install all the required SFCFS packages on the new node.

```
# ./installsfcfs -installonly
```

6 Enter `y` to install SFCFS on these systems.

```
Do you want to install SFCFS on these systems? [y,n,q] (y)
```

7 Enter the system name of the new node to install SFCFS.

```
Enter the system names separated by spaces on which to install  
SFCFS: system03
```

8 Enter a license key for the new node.

```
Enter a SFCFS license key for system03:[?]  
XXXX-XXXX-XXXX-XXXX-XXXX-X
```

9 Enter `y` or `n` for another license key. You are prompted to press Return to continue.

```
Do you want to enter another license key for system03?  
[y,n,q,?] (n)
```

10 Enter 1 or 2 to be installed on all systems.

```
Select the rpms to be installed on all systems?  
[1-2,q,?] (2)
```

11 Press Return to continue.

```
Press [Return] to continue:
```

Do not reboot the machine now.

12 Create the `/etc/llttab` file the same as it looks on another node in the cluster.

- Change the `set-node` line to the name of the new node, set the `set-cluster` line to the cluster ID, and specify that the LAN ports for the public and private networks are configured the same as on the other cluster nodes:

```
set-node system03  
set-cluster 100  
  
link eth1 eth-[MACID for eth1] - ether --  
link eth2 eth-[MACID for eth2] - ether --
```

- Copy `/etc/llthosts` from one other node of the cluster and add a new line to it with the name of this node.
- Edit the `/etc/llthosts` file and add the new node to the file. For example:

```
0 system01  
1 system02  
2 system03
```

- Copy the `/etc/llthosts` file from one of the existing systems over to the new system. The `/etc/llthosts` file must be identical on all nodes in the cluster.

13 Create an `/etc/gabtab` file and add the following command to the file:

```
/sbin/gabconfig -c -nN
```

Where N is the number of nodes in the cluster. For a three node cluster, N would equal 3.

14 There is no need to reboot the other nodes, just update the `/etc/gabtab` file on the other nodes in the cluster. Edit the `/etc/gabtab` file on each of the existing systems, changing the content to match the file on the new system.

- 15 Start VxVM on the system that has been added.

```
# vxinstall
```

- 16 After starting VxVM, proceed to [Configuring Storage Foundation Cluster File System and Cluster Volume Manager agents on the new node](#).

Configuring Storage Foundation Cluster File System and Cluster Volume Manager agents on the new node

This section describes how to configure SFCFS and CVM agents on the new node.

To configure SFCFS and CVM agents on the new node

- 1 Verify if `/etc/VRTSvcs/conf/config/.stale` file is present, before starting VCS.

If `/etc/VRTSvcs/conf/config/.stale` file is not present, enter:

```
# touch /etc/VRTSvcs/conf/config/.stale
```

- 2 Start the VCS server and `vxfen` on the new node:

- Start LLT and GAB on the new node:

```
# /etc/init.d/llt start
```

```
# /etc/init.d/gab start
```

- For starting `vxfen` in the disable mode, run the following commands on `system03`:

```
# echo vxfen_mode=disabled > /etc/vxfenmode
```

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

- For starting `vxfen` in the enabled mode:

- Copy the following files from one of the existing cluster nodes to the new node:

```
/etc/vxfenmode
```

```
/etc/vxfendg
```

- Run the following command:

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

- 3 On the new node, verify that the GAB port memberships are *a* and *b*. Run the following command:

```
# /sbin/gabconfig -a
```

- 4 Determine the CVM master node:

```
# vxctl -c mode
```

- 5 Make a backup copy of the `main.cf` file. Enter the following commands:

```
# cd /etc/VRTSvcs/conf/config
# cp main.cf main.cf.2node
```

- 6 Open the VCS configuration for writing and add the new node. For example:

```
# haconf -makerw
# hasys -add system03
```

- 7 Add the new node to the CVM system list and specify a failover priority of zero:

```
# hagr -modify cvm SystemList -add system03 X
```

where *X* is one more than the index of the last system in System list of CVM service group in `/etc/VRTSvcs/conf/config/main.cf`.

- 8 Add the new node to the CVM AutoStartList:

```
# hagr -modify cvm AutoStartList -add system03
```

- 9 Node ID can be obtained from `CVMNodeId` of `/etc/VRTSvcs/conf/config/main.cf`. Add the new node, `system03`, and its node ID, `#`, to the `cvm_clust` resource:

```
# hares -modify cvm_clus CVMNodeId -add system03 2
```

- 10 Write the new VCS configuration to disk:

```
# haconf -dump -makero
```

- 11 Verify the syntax of `main.cf` file:

```
# hacf -verify .
```

12 To enable the existing cluster to recognize the new node, execute on all the nodes in the cluster:

```
# /etc/vx/bin/vxclustadm -m vcs -t gab reinit
# /etc/vx/bin/vxclustadm nidmap
```

13 Start CVM on the newly added node.

- Determine the node ID:

```
# cat /etc/llthosts
```

- Verify that this host ID is seen by the GAB module:

```
# gabconfig -a
```

- Start the VCS engine.

- If on the newly added node ports `f`, `u`, `v`, or `w` are present before `hastart`, then the newly added node must be rebooted to properly start VCS. To properly start VCS:

```
# shutdown -r
```

- If on the newly added node ports `f`, `u`, `v` or `w` were not present before `hastart`, then use the following command to start VCS:

```
# hastart
```

14 Check the system status to see whether the new node is online:

```
# hastatus -sum
-- SYSTEM STATE
-- System      State      Frozen
A      system01  RUNNING   0
A      system02  RUNNING   0
A      system03  RUNNING   0

-- GROUP STATE
-- Group      System      Probed  AutoDisabled  State
B cvm        system01    Y       N              ONLINE
B cvm        system02    Y       N              ONLINE
B cvm        system03    Y       N              ONLINE
```

- 15 Add shared disk groups to the cluster configuration:

```
# cfsdgadm add cfsdg system03=sw
```

- 16 Create a `/mnt` on `system03` and run the following commands for the shared mount points:

```
# cfsmntadm modify /mnt add system03=rw
```

See `cfsmntadm(1M)` manual page.

- 17 Use the `cfsmount` command to cluster mount `/mnt` on the new node:

```
# cfsmount /mnt
```

Removing a node from a cluster

This section describes how to remove a node from a cluster. As in previous examples, the following removes the node `system03` from a three-node cluster. The procedure can be done from any node remaining in the cluster or from a remote host.

To remove a node from a cluster

- 1 Log in as superuser on a node other than `system03`.
- 2 Use the `cfsmount` command to unmount the file system `/mnt` on all the nodes:

```
# cfsmount /mnt system03
```

- 3 Stop all the cluster components:

```
# cfscluster stop -f system03
```

- 4 Open the VCS configuration for writing:

```
# haconf -makerw
```

- 5 Remove system03 from the system list attribute of the CVM and SFCFS service groups:

```
# hagrps -modify service_group SystemList -delete system03
```

```
# hagrps -modify cvm SystemList -delete system03
```

where *service_group* is the command that displays the service groups by `hagrps -dep cvm`.

If an error message similar to the following is displayed by either of the above commands:

```
VCS:10456:Configuration must be ReadWrite. ('hagrps
-multiply ...
-delete (0x10f)', Sysstate=RUNNING, Channel=IPM, Flags=0x0)
```

Repeat 4 and the command that failed in 5.

- 6 Write the new VCS configuration to disk:

```
# haconf -dump -makero
```

- 7 Edit the `/etc/llthosts` file on the remaining nodes of the cluster, and remove the entry corresponding to the node being removed.

- 8 Edit the `/etc/gabtab` file on the remaining nodes of the cluster and edit the `gabconfig` command to reflect the correct and new number of nodes in the cluster.

- 9 Login to system03 and remove the following files:

```
# rm /etc/vxfenmode
```

```
# rm /etc/llthosts
```

```
# rm /etc/llttab
```

```
# rm /etc/gabtab
```

- 10 If fencing was enabled on the cluster, run the following commands:

```
# rm /etc/vxfentab
```

```
# rm /etc/vxfendg
```

- 11 Reboot system03:

```
# /sbin/shutdown -r now
```

12 Change to the `install` directory:

```
# cd /opt/VRTS/install
```

13 From the `scripts` directory, run the `uninstallsfcfs` script and remove SFCFS on `system03`:

```
# ./uninstallsfcfs
```

If you do not want to remove the Veritas Cluster Server software, enter **n** when prompted to uninstall VCS.

See the *Veritas Cluster Server Installation Guide*.

Storage Foundation Cluster File System for Oracle RAC

This chapter includes the following topics:

- [Storage Foundation Cluster File System for Oracle RAC](#)

Storage Foundation Cluster File System for Oracle RAC

Maintenance Pack 3 for Veritas Storage Foundation Cluster File System (SFCFS) 5.0 includes support for Oracle RAC.

SFCFS for Oracle RAC supports dynamic multi-pathing (DMP), Oracle Disk Manager (ODM) and Group Membership Services.

SFCFS for Oracle RAC does not support the integration of Veritas Cluster Server (VCS) with the Oracle Clusterware APIs.

In addition, although I/O Fencing may be enabled, it is not supported for use with SFCFS for Oracle RAC. If I/O Fencing is configured, the functionality is not available.

Oracle Cluster Registry (OCR) and Vote-disk are not supported on Cluster File System (CFS). Symantec recommends keeping OCR and Vote-disk on volumes created by the Cluster Volume Manager (CVM).

`CRS_HOME` is not supported on Cluster File System (CFS).

About Storage Foundation Cluster File System for Oracle RAC

SFCFS for Oracle RAC is a storage management and clustering solution.

SFCFS for Oracle RAC provides the following features:

- Enhanced scalability and availability with access to multiple RAC instances per database in a cluster.
- Back up and recovery of databases using volume-level and file system-level snapshot technologies. SFCFS for Oracle RAC enables full volume-level snapshots for off-host processing that reduce load on production systems. The file system-level snapshots that involve efficient backup and rollback processes.
- Sharing of all types of files, in addition to Oracle database files, across nodes.
- Increased availability and performance with dynamic multi-pathing (DMP), which provides wide storage array support for protection from failures and performance bottlenecks in the HBAs and SAN switches.
- Optimized I/O performance and memory footprints through storage mapping technologies, such as ODM, and tunable attributes.

How SFCFS for Oracle RAC works (high-level perspective)

Real Application Clusters (RAC) is a parallel database environment that takes advantage of the processing power of multiple computers. The Oracle database is the physical data stored in tablespaces on disk. The Oracle instance is a set of processes and shared memory that provide access to the physical database. Specifically, the instance involves server processes acting on behalf of clients to read data into shared memory and make modifications to it, and background processes to write changed data to disk.

In traditional environments, only one instance accesses a database at a specific time. SFCFS for Oracle RAC enables all nodes to concurrently run Oracle instances and execute transactions against the same database. This software coordinates access to the shared data for each node to provide consistency and integrity.

Each node adds its processing power to the cluster as a whole and can increase overall throughput or performance.

At a conceptual level, SFCFS for Oracle RAC is a cluster that manages applications (instances), networking, and storage components using resources that are contained in service groups.

SFCFS for Oracle RAC clusters have many of the same properties as Veritas Cluster Server (VCS) clusters, such as the following:

- Each node runs its own operating system.
- A cluster interconnect enables cluster communications.
- A public network connects each node to a LAN for client access.
- Shared storage is accessible by each node that needs to run the application.

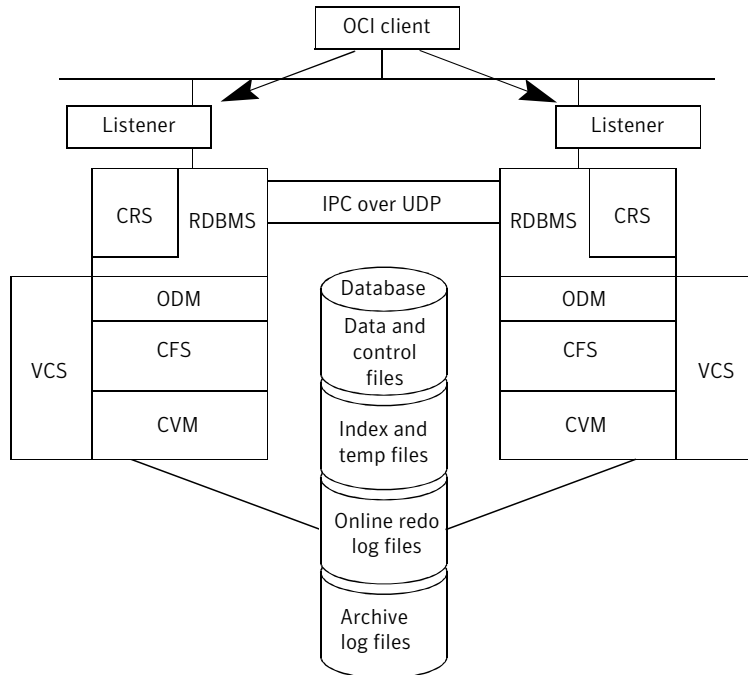
SFCFS for Oracle RAC adds the following technologies, engineered specifically to improve performance, availability, and manageability of Oracle RAC environments, to a failover cluster environment:

- Storage Foundation Cluster File System (SFCFS) and Cluster Volume Manager (CVM) technologies to manage multi-instance database access to shared storage.
- An Oracle Disk Manager (ODM) library to maximize Oracle disk I/O performance.

SFCFS for Oracle RAC provides an environment that can tolerate failures with minimal downtime and interruption to users. If a node fails as clients access the same database on multiple nodes, clients attached to the failed node can reconnect to a surviving node and resume access. Recovery after failure in the SFCFS for Oracle RAC environment is far quicker than recovery for a failover database because another Oracle instance is already up and running. The recovery process involves applying outstanding redo log entries from the failed node.

Figure 4-1 describes the SFCFS for Oracle RAC architecture.

Figure 4-1 SFCFS for Oracle RAC architecture



Component products and processes of SFCFS for Oracle RAC

To understand how Storage Foundation Cluster File System for Oracle RAC (SFCFS for Oracle RAC) manages database instances running in parallel on multiple nodes, review the architecture and communication mechanisms that provide the infrastructure for Oracle RAC.

General highlights of the component products include the following.

Cluster Volume Manager (CVM)	Enables simultaneous access to shared volumes based on technology from Veritas Volume Manager (VxVM).
Storage Foundation Cluster File System (SFCFS)	Enables simultaneous access to shared file systems based on technology from Veritas File System (VxFS).
Database Accelerator	Provides the interface with the Oracle Disk Manager (ODM) API.
Cluster Server (VCS)	Provides cluster membership, a messaging layer, management of CVM and CFS service groups and resources.

Communication infrastructure

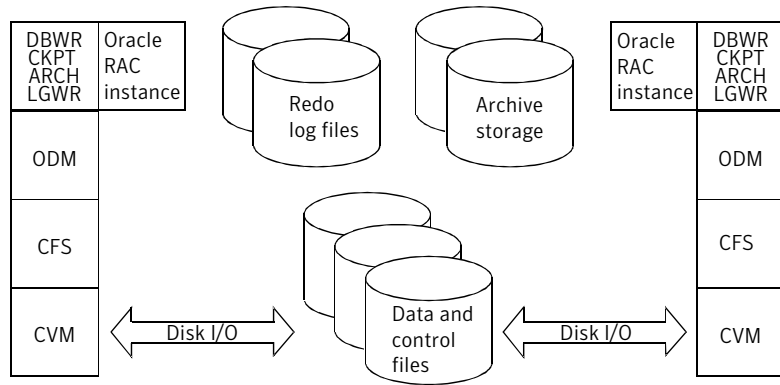
To understand the communication infrastructure, it is necessary to review the data flow and communications requirements.

Data flow

The CVM, CFS, ODM, and Oracle RAC elements reflect the overall data flow, or data stack, from an instance running on a server to the shared storage. The various Oracle processes that compose an instance (such as DB Writers, Checkpoint, Archiver, Log Writer, and Server) read and write data to the storage through the I/O stack. Oracle communicates through the ODM interface to CFS, which in turn accesses the storage through CVM.

[Figure 4-2](#) describes the data flow in SFCFS for Oracle RAC.

Figure 4-2 Data flow in SFCFS for Oracle RAC

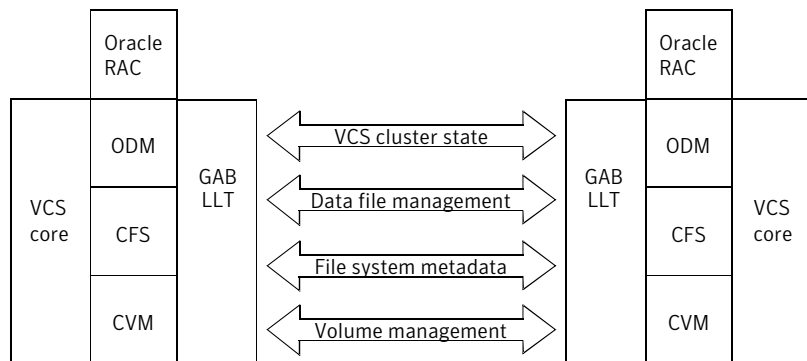


Communication requirements

End users on a client system are unaware that they are accessing a database hosted by multiple instances. The key to performing I/O to a database accessed by multiple instances is communication between the processes. Each layer or component in the data stack must reliably communicate with its peer on other nodes to function properly. RAC instances communicate to coordinate the protection of data blocks in the database. ODM processes communicate to coordinate the protection of data files and access across the cluster. SFCFS coordinates metadata updates for file systems, and CVM coordinates the status of logical volumes and maps.

[Figure 4-3](#) describes the communication requirements in SFCFS for Oracle RAC.

Figure 4-3 Communication requirements in SFCFS for Oracle RAC



Note: Oracle cache fusion and lock management traffic goes over UDP since SFCFS for Oracle RAC does not contain the Veritas provided Inter-Process Communication library (VCSIPC) and Low Latency Transport Multiplexer (LMX) module.

Cluster Volume Manager

CVM is an extension of Veritas Volume Manager, the industry standard storage virtualization platform. CVM extends the concepts of VxVM across multiple nodes. Each node recognizes the same logical volume layout, and more importantly, the same state of all volume resources.

CVM supports performance-enhancing capabilities, such as striping, mirroring, and mirror break-off (snapshot) for off-host backup. You can use standard VxVM commands from one node in the cluster to manage all storage. All other nodes immediately recognize any changes in disk group and volume configuration with no interaction.

Cluster Volume Manager architecture

CVM is designed with a "master and slave" architecture. One node in the cluster acts as the configuration master for logical volume management, and all other nodes are slaves. Any node can take over as master if the existing master fails.

The CVM master exists on a per-cluster basis and uses GAB and LLT to transport its configuration data.

Just as with VxVM, the Volume Manager configuration daemon, `vxconfigd`, maintains the configuration of logical volumes. This daemon handles changes to the volumes by updating the operating system at the kernel level. For example, if a mirror of a volume fails, the mirror detaches from the volume and `vxconfigd` determines the proper course of action, updates the new volume layout, and informs the kernel of a new volume layout. CVM extends this behavior across multiple nodes and propagates volume changes to the master `vxconfigd`. (You must perform operator-initiated changes on the master node.)

The `vxconfigd` process on the master pushes these changes out to slave `vxconfigd` processes, each of which updates the local kernel.

CVM does not impose any write locking between nodes. Each node is free to update any area of the storage. All data integrity is the responsibility of the upper application. From an application perspective, stand-alone systems access logical volumes in the same way as CVM systems. CVM imposes a "Uniform Shared Storage" model. All nodes must connect to the same disk sets for a given disk group. Any node unable to detect the entire set of physical disks for a given disk

group cannot import the group. If a node loses contact with a specific disk, CVM excludes the node from participating in the use of that disk.

Storage Foundation Cluster File System

CFS enables you to simultaneously mount the same file system on multiple nodes and is an extension of the industry-standard Veritas File System. Unlike some other file systems that send data through another node to the storage, CFS is a true SAN file system. All data traffic takes place over the storage area network (SAN), and only the metadata traverses the cluster interconnect.

In addition to using the SAN fabric for reading and writing data, CFS offers storage checkpoints and rollback for backup and recovery.

Storage Foundation Cluster File System architecture

SFCFS for Oracle RAC uses CFS to manage a file system in a large database environment. Since CFS is an extension of VxFS, it operates in a similar fashion and caches metadata and data in memory (typically called buffer cache or vnode cache). CFS uses a distributed locking mechanism called Global Lock Manager (GLM) to ensure all nodes have a consistent view of the file system. GLM provides metadata and cache coherency across multiple nodes by coordinating access to file system metadata, such as inodes and free lists. The role of GLM is set on a per-file system basis to enable load balancing.

CFS involves a primary/secondary architecture. Though any node can initiate an operation to create, delete, or resize data, the GLM master node carries out the actual operation. After creating a file, the GLM master node grants locks for data coherency across nodes. For example, if a node tries to modify a row of a block in a file, it must obtain an exclusive lock to ensure other nodes that may have the same file cached have this cached copy invalidated.

SFCFS for Oracle RAC configurations minimize the use of GLM locking. Oracle RAC accesses the file system through the ODM interface and handles its own locking; only Oracle (and not GLM) buffers data and coordinates write operations to files.

A single point of locking and buffering ensures maximum performance. GLM locking is only involved when metadata for a file changes, such as during create and resize operations.

Oracle Disk Manager

SFCFS for Oracle RAC requires Oracle Disk Manager (ODM), a standard API published by Oracle for support of database I/O. SFCFS for Oracle RAC provides a library for Oracle to use as its I/O library.

Oracle Disk Manager architecture

When the ODM library is linked, Oracle is able to bypass all caching and locks at the file system layer and to communicate directly with raw volumes.

The SFCFS for Oracle RAC implementation of ODM generates performance equivalent to performance with raw devices while the storage uses easy-to-manage file systems.

All ODM features can operate in a cluster environment. Nodes communicate with each other before performing any operation that could potentially affect another node. For example, before creating a new data file with a specific name, ODM checks with other nodes to see if the file name is already in use.

Oracle Disk Manager performance enhancements

The following describes ODM performance enhancements.

ODM enables the following performance benefits that are provided by the Oracle Disk Manager:

- Locking for data integrity.
- Few system calls and context switches.
- Increased I/O parallelism.
- Efficient file creation and disk allocation.

Databases using file systems typically incur the following additional overhead:

- Extra CPU and memory usage to read data from underlying disks to the file system cache. This scenario requires copying data from the file system cache to the Oracle cache.
- File locking that allows for only a single writer at a time. Allowing Oracle to perform locking allows for finer granularity of locking at the row level.
- File systems generally go through a standard Sync I/O library when performing I/O. Oracle can make use of Kernel Async I/O libraries (KAIO) with raw devices to improve performance.

Veritas Cluster Server functionality for SFCFS for Oracle RAC

The following VCS functionality is provided for SFCFS for Oracle RAC.

Veritas Cluster Server provides the following functionality for SFCFS for Oracle RAC:

- Management of cluster node membership, including the monitoring of node health by using heart-beat technology.
- A messaging layer between cluster nodes.
- Management of CVM and SFCFS service groups and resources and their interdependencies.

Veritas Cluster Server architecture

The High Availability Daemon (HAD) is the main VCS daemon running on each node. HAD tracks changes in the cluster configuration and monitors resource status by communicating with GAB and LLT. HAD manages all application services using agents, which are installed programs to manage resources (specific hardware or software entities).

The VCS architecture is modular for extensibility and efficiency. HAD does not need to know how to start up Oracle or any other application under VCS control.

Instead, you can add agents to manage different resources with no effect on the engine (HAD). Agents only communicate with HAD on the local node, and HAD communicates status with HAD processes on other nodes. Because agents do not need to communicate across systems, VCS is able to minimize traffic on the cluster interconnect.

SFCFS for Oracle RAC provides specific agents for VCS to manage CVM and SFCFS.

Cluster configuration files

VCS uses the following configuration files in a default configuration.

<code>main.cf</code>	Defines the entire cluster, including the cluster name, systems in the cluster, definitions of service groups, resources, and in addition to service group and resource dependencies.
<code>types.cf</code>	Defines the resource types

Additional files similar to `types.cf` file may be present if you add agents.

Installing and configuring SFCFS for Oracle RAC

You can install SFCFS for Oracle RAC on clusters of up to 16 nodes. You can also upgrade an existing SFCFS cluster to SFCFS for Oracle RAC.

The Veritas product `installer` or the `installsfcfsrac` scripts may be used to perform a fresh installation of SFCFS for Oracle RAC.

The following packages are installed on each cluster node:

- Veritas Cluster Server (VCS)
- Veritas Volume Manager (VxVM)
- Veritas File System (VxFS)
- Oracle Disk Manager (ODM)

To perform system checks and configure SFCFS for Oracle RAC, you can use the Veritas product `installer` or the `-configure` option of the `installsfcfsrac` script. You can configure the following components of SFCFS for Oracle RAC:

- VCS
- CVM (Veritas Volume Manager enabled for clusters)
- CFS (Veritas File System enabled for clusters)

The `installer` and `installsfcfsrac` scripts also start the SFCFS for Oracle RAC processes.

See the *Veritas Storage Foundation Installation Guide*.

Recommendations before installing Oracle software

After installing and configuring SFCFS for Oracle RAC and before installing Oracle software, read the following recommendations.

About the location of ORACLE_HOME

`ORACLE_HOME` is the location where the Oracle Database binaries are installed. Before installing Oracle binaries on `ORACLE_HOME` locally on each node or on a cluster file system on shared storage, read and comply with the following information:

- Local installations provide a comfort level using traditional installation methods and the possibility of improved protection against a single point of failure.
- SFCFS installations provide a single Oracle installation to manage, regardless of the number of nodes. This scenario offers a necessary reduction in storage requirements and easy addition of nodes.

Select the location based on your high availability (HA) requirements. Symantec and Oracle recommend using local installations.

About the location of CRS_HOME

CRS_HOME is the location where Oracle Clusterware binaries are installed. CRS_HOME is not supported on Cluster File System (CFS).

Creating the Oracle user and group

On each system, create a local group and local user for Oracle. Be sure to assign the same group ID, user ID, and home directory for the user on each system. The following example creates the group **oinstall** and the user **oracle**.

To create the operating system Oracle user and group on each system

- 1 Create the **oinstall** group on each system:

```
# groupadd -g 1000 oinstall
# groupadd -g 1001 dba
```

- 2 Create the **oracle** user and the oracle ID on each system:

```
# useradd -g oinstall -u user_id -G dba -d /home/oracle oracle
```

- 3 Enable `rsh` and key-based authentication `ssh` for the **oracle** user on all nodes.

See the Oracle product documentation for more information on creating the Oracle user and group.

Creating CRS_HOME

The following procedure creates CRS_HOME.

To create CRS_HOME on each system

- 1 Log in as root user on one system.

```
# su - root
```

- 2 On one node, create a disk group:

```
# vxdg init crsdg sdc
```

- 3 Create the volume in the group for `CRS_HOME`:

```
# vxassist -g crsdg make crsvol 500M
```

Refer to the Oracle product documentation for exact size requirements for `CRS_HOME`.

- 4 Create a VxFS file system on which to install CRS:

```
# mkfs -t vxfs /dev/vx/rdisk/crsdg/crsvol
```

- 5 Create the mount point for `CRS_HOME`:

```
# mkdir /oracle/crs
```

- 6 Mount the file system, using the device file for the block device:

```
# mount -t vxfs /dev/vx/dsk/crsdg/crsvol /oracle/crs
```

- 7 Edit the `/etc/fstab` file and list the new file system:

```
/dev/vx/dsk/crsdg/crsvol /oracle/crs vxfs defaults 1 1
```

- 8 Export the `CRS_HOME` directory as `/oracle/crs` for the **oracle** user.

- 9 Assign ownership of the directory to **oracle** and the group **oinstall**:

```
# chown -R oracle:oinstall /oracle/crs
```

```
# chmod -R 755 /oracle/crs
```

- 10 On each cluster node, repeat step 1 through step 9.

Creating ORACLE_HOME

The following procedure creates `ORACLE_HOME`.

To create ORACLE_HOME on each system

- 1 Log in as root user on one system.

```
# su - root
```

- 2 On one node, create a disk group:

```
# vxdg init orabindg sde
```

For a shared ORACLE_HOME, run the following command on the CVM master:

```
# vxdg -s init orabindg sde
```

- 3 Create the volume in the group for ORACLE_HOME:

```
# vxassist -g orabindg make orabinvol 500M
```

Refer to the Oracle product documentation for exact size requirements for ORACLE_HOME.

- 4 Create a VxFS file system on which to install Oracle Database:

```
# mkfs -t vxfs /dev/vx/rdisk/orabindg/orabinvol
```

- 5 Create the mount point for ORACLE_HOME:

```
# mkdir /oracle/orabin
```

- 6 Mount the file system, using the device file for the block device:

```
# mount -t vxfs /dev/vx/dsk/orabindg/orabinvol /oracle/orabin
```

For each shared ORACLE_HOME, run the following command on the CVM master:

```
# mount -t vxfs -o cluster /dev/vx/dsk/orabindg/orabinvol /oracle/orabin
```

- 7 For each local ORACLE_HOME, only, edit the /etc/fstab file and list the new file system:

```
/dev/vx/dsk/orabindg/orabinvol /oracle/orabin vxfs defaults 1 1
```

- 8 Export the ORACLE_HOME directory as /oracle/orabin for the oracle user.

- 9 Assign ownership of the directory to **oracle** and the group **oinstall**:

```
# chown -R oracle:oinstall /oracle/orabin
# chmod -R 755 /oracle/orabin
```

- 10 For each local `ORACLE_HOME`, repeat step 1 through step 9 on each cluster node.
For each shared `ORACLE_HOME`, repeat step 5 through step 6.

Configuring private IP addresses for Oracle Clusterware Installation

The CRS daemon requires a private IP address on each system to enable communications and heartbeating.

See the Oracle product documentation for information on configuring private IP addresses on all of the cluster nodes.

Obtaining public virtual IP addresses for use by Oracle

Before starting the Oracle installation, configure a virtual IP address for each node. Register an IP address and an associated host name in the domain name service (DNS) for each public network interface.

See the Oracle product documentation for information on configuring virtual IP addresses on all of the cluster nodes.

Creating Oracle Cluster Registry and Voting-disk volumes for Cluster Ready Services installation

Create the Oracle Cluster Registry (OCR) and Vote-disk shared volumes. The CFS directory is not supported by Oracle for OCR and Vote-Disk. After creating shared volumes, you can add them in the VCS configuration to make them highly available. The installation of Oracle Clusterware (CRS) requires a predefined location for the Oracle Cluster Registry (OCR) and Vote-disk components.

To create the Oracle Cluster Registry and Vote-disk shared volumes for Cluster Ready Services installation

- 1 Log in as superuser.
- 2 On the master node, create a shared disk group. For example:

```
# vxvg -s init ocrvotedg sdd
```

where *ocrvotedg* is the Oracle Cluster Registry Vote-disk group.

- 3 Create volumes in the shared group for OCR and Vote-disk. For example:

```
# vxassist -g ocrvotedg make ocrvol 500M
# vxassist -g ocrvotedg make vdvol 500M
```

where *ocrvotedg* is the Oracle Cluster Registry Vote-disk group.

where *ocrvol* is the Oracle Cluster Registry volume.

where *vdvol* is the Vote-disk volume.

where *500M* is the MB size of the volumes.

See the *Oracle* documentation for minimum size requirements for OCR and Vote-disk.

The OCR volume can be mirrored to provide high availability in case of failures for one of the mirrors. During the CRS operation, if one of the mirrors fails, then the other mirror can be used immediately without interrupting CRS functionality.

See the *Veritas Volume Manager Administrator's Guide* for details about creating mirrored volumes.

Mirroring of the OCR volume is not a requirement from Oracle. More than one volumes can be specified for Voting-disk, if `Normal Redundancy` option is selected in the `Specify Voting Disk Location` screen of the Oracle CRS installer. Create those many Vote-disk volumes.

- 4 Assign ownership of the volumes using the `vxedit(1M)` command. For example:

```
# vxedit -g ocrvotedg set user=oracle group=oinstall mode=640 ocrvol
# vxedit -g ocrvotedg set user=oracle group=oinstall mode=644 vdvol
```

where *ocrvotedg* is the Oracle Cluster Registry Vote-disk group.

where *root* and *oracle* are the user names.

where *oinstall* is the group name.

where *640* and *644* are the mode values.

where *ocrvol* is the Oracle Cluster Registry volume.

where *vdvol* is the Vote-disk volume.

- 5 Proceed to [Verifying the Oracle Cluster Registry and Vote-disk shared volumes](#).

Verifying the Oracle Cluster Registry and Vote-disk shared volumes

Verifying the OCR and Vote-disk shared volumes that have been created and configured.

To verify the Oracle Cluster Registry and Vote-disk shared volumes

- ◆ Verify that the Oracle Cluster Registry and Vote-disk shared volumes are present, enter:

```
# ls -l /dev/vx/rdisk/ocrvotedg/*
```

Sample output:

```
crw-r----- 1 root oinstall 199, 3 Jun 26 15:58
/dev/vx/rdisk/ocrvotedg/ocrvol

crw-r--r-- 1 oracle oinstall 199, 4 Jun 26 15:58
/dev/vx/rdisk/ocrvotedg/vdvol
```

These shared volumes are used during the Oracle Clusterware (CRS) Installation. During the CRS installation, use the following locations whenever requested by the CRS installer.

For OCR, use: /dev/vx/rdisk/ocrvotedg/ocrvol

For Vote-disk, use: /dev/vx/rdisk/ocrvotedg/vdvol

Installing Oracle Clusterware and Database software

See the Oracle product documentation for information on installing Oracle Clusterware and Database software.

Note: Veritas Storage Foundation Cluster File System for Oracle RAC does not support Symantec's implementation of SCSI-3 PGR-based I/O fencing. Oracle Clusterware (CRS) is expected to handle any split-brain situations.

See the following TechNote for more information:

<http://entsupport.symantec.com/docs/306411>

Relinking Oracle Database with Veritas Extension for Oracle Disk Manager

After installing Oracle Database, you must relink Oracle Database with Veritas Extension for Oracle Disk Manager (ODM).

See [Configuring Veritas Extension for Oracle Disk Manager](#).

After the installation of both CRS, database software and re-linking of Oracle Database software, the Oracle database can be created.

See [Creating a database for Oracle 10g or 11g](#) for more information on creating an Oracle database on shared raw VxVM volumes or shared VxFS file systems.

Upgrading Oracle Database

This section provides information for upgrading Oracle Database. You can upgrade the database only if you are already running a cluster on SFCFS for Oracle RAC.

The following list provides an overview of the tasks you must perform to upgrade Oracle Database:

- Upgrade the operating system on each node, if required
- Upgrade to SFCFS for Oracle RAC 5.0MP3 using the installer (`installmp` or `installer`)
- Upgrade Oracle Database

Upgrade paths for the operating system

[Table 4-1](#) shows the upgrade paths for the operating systems on the nodes.

Table 4-1 Upgrade paths for the operating system

Base operating system	Architecture	Upgrade operating system
RedHat Enterprise Linux 4 Update 4	x86_64 Intel Xeon, AMD, Opteron	<ul style="list-style-type: none"> ■ RedHat Enterprise Linux 4 Update 5 ■ RedHat Enterprise Linux 4 Update 6
RedHat Enterprise Linux 4 Update 4	x86_64 Intel Xeon, AMD, Opteron	<ul style="list-style-type: none"> ■ RedHat Enterprise Linux 5 Update 1 ■ RedHat Enterprise Linux 5 Update 2
Oracle Enterprise Linux 4 Update 4	x86_64 Intel Xeon, AMD, Opteron	<ul style="list-style-type: none"> ■ Oracle Enterprise Linux 4 Update 5 ■ Oracle Enterprise Linux 5 Update 1
SUSE Linux Enterprise Server 9 Service Pack 3	x86_64 Intel Xeon, AMD, Opteron	<ul style="list-style-type: none"> ■ SUSE Linux Enterprise Server 9 Service Pack 4

Table 4-1 Upgrade paths for the operating system (*continued*)

Base operating system	Architecture	Upgrade operating system
SUSE Linux Enterprise Server 9 Service Pack 3	x86_64 Intel Xeon, AMD, Opteron	<ul style="list-style-type: none"> ■ SUSE Linux Enterprise Server 10 Service Pack 1 ■ SUSE Linux Enterprise Server 10 Service Pack 2

Upgrading the operating system

The following procedure upgrades the operating system on the cluster nodes.

To upgrade the operating system

- ◆ Upgrade the operating system on each cluster node, if required.
 - See the operating system documentation for information on upgrading the operating system.
 - If you perform the upgrade on the same operating system distribution, such as from RHEL4 U4 to RHEL4 U6, then after the upgrade the Oracle resources might start automatically.

Upgrade paths for Veritas Storage Foundation Cluster File System for Oracle RAC to 5.0MP3

The only supported upgrade paths to SFCFS for Oracle RAC 5.0MP3 are from Storage Foundation for Oracle RAC 4.1 or 5.0, including all rolling patches and maintenance packs. Before upgrading to SFCFS for Oracle RAC 5.0MP3, upgrade the Linux system to a version that is supported for this release.

See “[Upgrading the operating system](#)” on page 74.

[Table 4-2](#) shows the upgrade paths for SFCFS for Oracle RAC on RHEL4.

Table 4-2 Upgrade paths for Storage Foundation Cluster File System for Oracle RAC on RHEL4

Storage Foundation for Oracle RAC version	RHEL4 Update 3 upgrade path	RHEL4 Update 4 upgrade path	RHEL4 Update 5 upgrade path	RHEL4 Update 6 upgrade path
4.1, 4.1 MP1, 4.1 MP2, 4.1 MP3, 4.1 MP4, 5.0, 5.0 MP1	Uninstall SF for Oracle RAC, then install SFCFS for Oracle RAC 5.0 MP3	Uninstall SF for Oracle RAC, then install SFCFS for Oracle RAC 5.0 MP3	Uninstall SF for Oracle RAC, then install SFCFS for Oracle RAC 5.0 MP3	Uninstall SF for Oracle RAC, then install SFCFS for Oracle RAC 5.0 MP3

Table 4-2 Upgrade paths for Storage Foundation Cluster File System for Oracle RAC on RHEL4 (*continued*)

Storage Foundation for Oracle RAC version	RHEL4 Update 3 upgrade path	RHEL4 Update 4 upgrade path	RHEL4 Update 5 upgrade path	RHEL4 Update 6 upgrade path
5.0 MP2	Install SFCFS for Oracle RAC 5.0 MP3	Install SFCFS for Oracle RAC 5.0 MP3	Install SFCFS for Oracle RAC 5.0 MP3	Install SFCFS for Oracle RAC 5.0 MP3
None	Install SFCFS for Oracle RAC 5.0 MP3	Install SFCFS for Oracle RAC 5.0 MP3	Install SFCFS for Oracle RAC 5.0 MP3	Install SFCFS for Oracle RAC 5.0 MP3

Table 4-3 shows the upgrade paths for SFCFS for Oracle RAC on RHEL4, including upgrading RHEL4 to RHEL5.

Table 4-3 Upgrade paths for Storage Foundation Cluster File System for Oracle RAC on RHEL4, including upgrading RHEL4 to RHEL5

Storage Foundation for Oracle RAC version	RHEL4 Update 3 upgrade path	RHEL4 Update 4 upgrade path	RHEL4 Update 5 upgrade path	RHEL4 Update 6 upgrade path
4.1, 4.1 MP1, 4.1 MP2, 4.1 MP3, 4.1 MP4, 5.0, 5.0 MP1	Uninstall SF for Oracle RAC, upgrade to RHEL5 Update 1 or Update 2, then install SFCFS for Oracle RAC 5.0 MP3	Uninstall SF for Oracle RAC, upgrade to RHEL5 Update 1 or Update 2, then install SFCFS for Oracle RAC 5.0 MP3	Uninstall SF for Oracle RAC, upgrade to RHEL5 Update 1 or Update 2, then install SFCFS for Oracle RAC 5.0 MP3	Uninstall SF for Oracle RAC, upgrade to RHEL5 Update 1 or Update 2, then install SFCFS for Oracle RAC 5.0 MP3
5.0 MP2	Upgrade to RHEL5 Update 1 or Update 2, then install SFCFS for Oracle RAC 5.0 MP3	Upgrade to RHEL5 Update 1 or Update 2, then install SFCFS for Oracle RAC 5.0 MP3	Upgrade to RHEL5 Update 1 or Update 2, then install SFCFS for Oracle RAC 5.0 MP3	Upgrade to RHEL5 Update 1 or Update 2, then install SFCFS for Oracle RAC 5.0 MP3

Table 4-3 Upgrade paths for Storage Foundation Cluster File System for Oracle RAC on RHEL4, including upgrading RHEL4 to RHEL5 (*continued*)

Storage Foundation for Oracle RAC version	RHEL4 Update 3 upgrade path	RHEL4 Update 4 upgrade path	RHEL4 Update 5 upgrade path	RHEL4 Update 6 upgrade path
None	Upgrade to RHEL5 Update 1 or Update 2, then install SFCFS for Oracle RAC 5.0 MP3	Upgrade to RHEL5 Update 1 or Update 2, then install SFCFS for Oracle RAC 5.0 MP3	Upgrade to RHEL5 Update 1 or Update 2, then install SFCFS for Oracle RAC 5.0 MP3	Upgrade to RHEL5 Update 1 or Update 2, then install SFCFS for Oracle RAC 5.0 MP3

Table 4-4 shows the upgrade paths for SFCFS for Oracle RAC on RHEL5.

Table 4-4 Upgrade paths for Storage Foundation Cluster File System for Oracle RAC on RHEL5

Storage Foundation for Oracle RAC version	RHEL5	RHEL5 Update 1	RHEL5 Update 2
4.1, 4.1 MP1, 4.1 MP2, 4.1 MP3, 4.1 MP4, 5.0, 5.0 MP1, 5.0 MP2	Not supported	Not supported	Not supported
None	Upgrade to RHEL5 Update 1 or Update 2, then install SFCFS for Oracle RAC 5.0 MP3	Install SFCFS for Oracle RAC 5.0 MP3	Install SFCFS for Oracle RAC 5.0 MP3

Table 4-3 shows the upgrade paths for SFCFS for Oracle RAC on SLES9.

Table 4-5 Upgrade paths for Storage Foundation Cluster File System for Oracle RAC on SLES9

Storage Foundation for Oracle RAC version	SLES 9 SP1 upgrade path	SLES 9 SP2 upgrade path	SLES 9 SP3 upgrade path	SLES 9 SP4 upgrade path
4.1, 4.1 MP1, 4.1 MP2, 4.1 MP3, 4.1 MP4, 5.0, 5.0 MP1	Uninstall SF for Oracle RAC, upgrade to SLES 9 SP3 or SP4, then install SFCFS for Oracle RAC 5.0 MP3	Uninstall SF for Oracle RAC, upgrade to SLES 9 SP3 or SP4, then install SFCFS for Oracle RAC 5.0 MP3	Uninstall SF for Oracle RAC, then install SFCFS for Oracle RAC 5.0 MP3	SLES9 SP4 is supported in SF for Oracle RAC 4.1 MP4 RP2; uninstall SF for Oracle RAC, then install SFCFS for Oracle RAC 5.0 MP3
5.0 MP2	Not supported	Not supported	Install SFCFS for Oracle RAC 5.0 MP3	SLES9 SP4 is supported in SF for Oracle RAC 5.0 MP2 RP1; install SFCFS for Oracle RAC 5.0 MP3
None	Upgrade to SLES9 SP3 or SP4, then install SFCFS for Oracle RAC 5.0 MP3	Upgrade to SLES9 SP3 or SP4, then install SFCFS for Oracle RAC 5.0 MP3	Install SFCFS for Oracle RAC 5.0 MP3	Install SFCFS for Oracle RAC 5.0 MP3

[Table 4-6](#) shows the upgrade paths for SFCFS for Oracle RAC on SLES9, including upgrading SLES9 to SLES10 SP1 or SLES10 SP2.

Table 4-6 Upgrade paths for Storage Foundation Cluster File System for Oracle RAC on SLES9, including upgrading SLES9 to SLES10 SP1 or SLES10 SP2

Storage Foundation for Oracle RAC version	SLES 9 SP1 upgrade path	SLES 9 SP2 upgrade path	SLES 9 SP3 upgrade path	SLES 9 SP4 upgrade path
4.1, 4.1 MP1, 4.1 MP2, 4.1 MP3, 4.1 MP4, 5.0, 5.0 MP1	Uninstall SF for Oracle RAC, upgrade to SLES 10 SP1 or SP2, then install SFCFS for Oracle RAC 5.0 MP3	Uninstall SF for Oracle RAC, upgrade to SLES 10 SP1 or SP2, then install SFCFS for Oracle RAC 5.0 MP3	Uninstall SF for Oracle RAC, upgrade to SLES 10 SP1 or SP2, then install SFCFS for Oracle RAC 5.0 MP3	SLES9 SP4 is supported in SF for Oracle RAC 4.1 MP4 RP2; uninstall SF for Oracle RAC, upgrade to SLES10 SP1 or SP2, then install SFCFS for Oracle RAC 5.0 MP3
5.0 MP2	Not supported	Not supported	Upgrade to SLES10 SP1 or SP2, then install SFCFS for Oracle RAC 5.0 MP3	SLES9 SP4 is supported in SF for Oracle RAC 5.0 MP2 RP1; upgrade to SLES10 SP1 or SP2, then install SFCFS for Oracle RAC 5.0 MP3
None	Upgrade to SLES10 SP1 or SP2, then install SFCFS for Oracle RAC 5.0 MP3	Upgrade to SLES10 SP1 or SP2, then install SFCFS for Oracle RAC 5.0 MP3	Upgrade to SLES10 SP1 or SP2, then install SFCFS for Oracle RAC 5.0 MP3	Upgrade to SLES10 SP1 or SP2, then install SFCFS for Oracle RAC 5.0 MP3

Table 4-7 shows the upgrade paths for SFCFS for Oracle RAC on SLES10.

Table 4-7 Upgrade paths for Storage Foundation Cluster File System for Oracle RAC on SLES10

Storage Foundation for Oracle RAC version	SLES10	SLES10 SP1	SLES10 SP2
4.1, 4.1 MP1, 4.1 MP2, 4.1 MP3, 4.1 MP4, 5.0, 5.0 MP1, 5.0 MP2	Not supported	Not supported	Not supported
None	Upgrade to SLES10 SP1 or SP2, then install SFCFS for Oracle RAC 5.0 MP3	Install SFCFS for Oracle RAC 5.0 MP3	Install SFCFS for Oracle RAC 5.0 MP3

Upgrading Veritas Storage Foundation Cluster File System for Oracle RAC to 5.0 MP3

The following procedure upgrades Veritas Storage Foundation Cluster File System for Oracle RAC to the 5.0 MP3 release.

To upgrade Veritas Storage Foundation Cluster File System for Oracle RAC to 5.0 MP3

- 1 Load and mount the disc.
- 2 Move to the top-level directory on the DVD.

```
# cd /mnt/cdrom
```
- 3 For new installations or upgrades from releases prior to 5.0, run the `installer` script:

```
# ./installer
```

For upgrades from 5.0 releases, including maintenance packs and rolling patches, run the `installmp` script:

```
# ./installmp
```

See the *Veritas Storage Foundation Installation Guide* for information on upgrading SFCFS for Oracle RAC stack.

See the *Veritas Storage Foundation Installation Guide* for the complete installation and upgrade procedures.

Fencing administration

This chapter includes the following topics:

- [I/O fencing](#)
- [Troubleshooting fenced configurations](#)

I/O fencing

Symantec recommends configuring the cluster with I/O fencing enabled. I/O fencing requires shared devices to support SCSI-3 Persistent Reservations (PR). Enabling I/O fencing prevents data corruption caused by a split brain scenario.

Symantec Storage Foundation Cluster File System is supported without I/O fencing enabled. However, without I/O fencing enabled, split brain scenarios can result in data corruption.

Symantec Storage Foundation Cluster File System for Oracle RAC is supported without I/O fencing enabled. However, without I/O fencing enabled, split brain scenarios can result in data corruption.

I/O fencing allows write access to members of the active cluster and blocks access to non-members. The physical components of I/O fencing are data disks and coordinator disks. Each has a unique purpose and uses different physical disk devices.

See the *Veritas Cluster Server Installation Guide*.

See the Hardware Compatibility List (HCL) at the following URL:

<http://support.veritas.com/docs/283161>

Data disks

Data disks are standard disk devices used for data storage. These can be physical disks or RAID Logical Units (LUNs). These disks must support SCSI-3 PGR. Data

disks are incorporated in standard VxVM/CVM disk groups. CVM is responsible for fencing data disks on a disk-group basis. Because VxVM enables I/O fencing, several other features are also provided. Disks added to a group are automatically fenced, as are new paths to a device.

Coordinator Disks

Coordinator disks are special-purpose disks. They comprise three (or an odd number greater than three) standard disks, or LUNs, set aside for use by I/O fencing during cluster reconfiguration.

The coordinator disks act as a global lock device during a cluster reconfiguration. This lock mechanism determines which node is allowed to fence off data drives from other nodes. From a high level, a system must eject a peer from the coordinator disks before it can fence the peer from the data drives. This concept of “racing” for control of coordinator disks is the key to understanding how fencing helps prevent split-brain.

Coordinator disks cannot be used for any other purpose. You cannot store data on them, or include them in a disk group for user data. They can be any three disks that support SCSI-3 PGR. Symantec recommends the coordinator disks use the smallest LUNs. Because coordinator disks do not store data, cluster nodes need only register with them, not reserve them.

Before you configure coordinator disks

I/O fencing requires coordinator disks to be configured in a disk group that each cluster system can access. The use of coordinator disks enables the `vxfen` driver to resolve potential split-brain conditions and prevent data corruption. A coordinator disk is not used for data storage, so it can be configured as the smallest LUN on a disk array to avoid wasting space.

Coordinator disks must meet the following requirements:

- There must be at least three coordinator disks and the total number of coordinator disks must be an odd number. This ensures a majority of disks can be achieved.
- Each of the coordinator disks must use a physically separate disk or LUN.
- Each of the coordinator disks should be on a different disk array, if possible.
- Coordinator disks in a disk array should use hardware-based mirroring.
- The coordinator disks must support SCSI-3 PR. Note that the use of the `vxfcntlsthdw` utility to test for SCSI-3 PR support requires that disks be 1MB or greater. Smaller disks can be tested manually.

Contact Veritas support (<http://support.veritas.com>) for the procedure.

Setting up the disk group for coordinator disks

This section describes setting up the disk group for coordinator disks.

To set up the disk group for coordinator disks

- 1 If you have already added and initialized disks you intend to use as coordinator disks, you can begin the following procedure at 5.
- 2 Physically add the three disks you intend to use for coordinator disks. All cluster nodes should physically share them. Symantec Corporation recommends that you use the smallest size disks or LUNs, so that space for data is not wasted.
- 3 If necessary, use the `vxdisk scandisks` command to scan the disk drives and their attributes. This command updates the VxVM device list and reconfigures DMP with the new devices. For example:

```
# vxdisk scandisks
```

- 4 Use the command `vxdisksetup` command to initialize a disk as a VxVM disk. The example command that follows specifies the CDS format:

```
# vxdisksetup -i vxvm_device_name format=cdsdisk
```

For example:

- 5 From one node, create a disk group named `vxfencoorddg`. This group must contain an odd number of disks or LUNs and a minimum of three disks. Symantec recommends that you use only three coordinator disks, and that you use the smallest size disks or LUNs to conserve disk space.

For example, assume the disks have the device names `EMC0_12`, `EMC0_16`, and `EMC0_17`.

- 6 On any node, create the disk group by specifying the device name of one of the disks.

```
# vxdbg -o coordinator=on init vxfencoorddg EMC0_12
```

Requirements for testing the coordinator disk group

Review the requirements for testing the coordinator disk group.

Run the `vxfstshdw` utility

Review the following guidelines on testing support for SCSI-3:

- The utility requires that the coordinator disk group be accessible from two systems. For example, if you have a four-system cluster, select any two systems for the test.
- If you configured ssh (SSH client) for the cluster nodes, VXFENTSTHDW can be used as long as ssh commands between nodes can execute without password prompting and confirmation.
If you did not configure ssh, enable each node to have remote rsh access to the other nodes during installation and disk verification. On each node, placing a “+” character in the first line of the `.rhosts` file gives remote access to the system running the install program. You can limit the remote access to specific nodes. Refer to the manual page for the `.rhosts` file for more information. Remove the remote rsh access permissions after the installation and disk verification process.
- ssh is used by default and rsh is only used if you do use the `vxfcntlsthdw -n` command.
- To ensure both nodes are connected to the same disk during the test, use the `vxfcntlsthdw -i diskpath` command to verify the disk serial number.
- The `vxfcntlsthdw` utility has additional options suitable for testing many disks. You can test disks without destroying data using the `-r` option. The options for testing disk groups (`-g`) and disks listed in a file (`-f`) are described in detail:

Testing the coordinator disk group

This section describes testing the coordinator disk group.

To test the coordinator disk group

- 1 To start the utility from one node, type the following command:

```
/opt/VRTSvcs/vxfen/bin/vxfcntlsthdw
```

Make sure system-to-system communication is functioning properly before performing this step.

See the `vxfcntlsthdw(1M)` man page.
- 2 After reviewing the overview and warning about overwriting data on the disks, confirm to continue the process and enter the node names.
- 3 Enter the name of the disk you are checking.

Creating the vxfcntlsthdw file

This section describes creating the `vxfcntlsthdw` file.

To create the `vxfgndg` file

- 1 To deport the disk group, type the following command:

```
# vxfgd deport vxfgcoorrdg
```

- 2 Import the disk group with the `-t` option to avoid automatically importing it when the nodes restart:

```
# vxfgd -t import vxfgcoorrdg
```

- 3 Deport the disk group. This operation prevents the coordinator disks from serving other purposes:

```
# vxfgd deport vxfgcoorrdg
```

- 4 On all nodes, type:

```
# echo "vxfgcoorrdg" > /etc/vxfgndg
```

Do not use spaces between the quotes in the “`vxfgcoorrdg`” text.

This command creates the `/etc/vxfgndg` file, which includes the name of the coordinator disk group. Based on the contents of the `/etc/vxfgndg` file, the `rc` script creates the `/etc/vxfgntab` file for use by the `vxfgn` driver when the system starts. The `rc` script also invokes the `vxfgnconfig` command, which configures the `vxfgn` driver to start and use the coordinator disks listed in `/etc/vxfgntab`. `/etc/vxfgntab` is a generated file; do not modify this file.

Adding or removing coordinator disks

Before adding coordinator disks, verify the disks support SCSI-3 persistent reservations.

To add and remove coordinator disks

- 1 Log in as `root` on any cluster node.
- 2 Import the coordinator disk group. The file `/etc/vxfendg` includes the name of the disk group containing the coordinator disks. Type:

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

where:

- | | |
|----|--|
| -t | Specifies that the disk group is imported only until the system reboots. |
| -f | Specifies that the import is to be done forcibly, which is necessary if one or more disks is inaccessible. |
| -C | Specifies any import blocks are removed. |

- 3 To add disks to, or remove them from, the disk group, use the VxVM disk administrator utility, `vxdiskadm`.

- 4 After disks are added or removed, deport the disk group:

```
# vxdg deport `cat /etc/vxfendg`
```

- 5 Reboot each system in the cluster to make the coordinator disks accessible.

Verifying fenced configurations

Administrators can use the `vxfenadm` command to test and troubleshoot fenced configurations. The `vxfenadm` command includes the following options:

- | | |
|----|--|
| -d | Display current I/O fencing mode |
| -g | Read and display keys |
| -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 |

Formatting the registration key

The key defined by VxVM associated with a disk group consists of seven bytes maximum. This key becomes unique among the systems when the VxVM prefixes it with the ID of the system. The key used for I/O fencing, therefore, consists of eight bytes.

[Table 5-2](#) describes the keys currently assigned to the disks.

Table 5-1 The keys currently assigned to the disks

0	1	2	3	4	5	6	7
Node	VxVM	VxVM	VxVM	VxVM	VxVM	VxVM	VxVM
ID	Defined	Defined	Defined	Defined	Defined	Defined	Defined

To display keys currently assigned to disks

- To display keys currently assigned to disks from the system with node ID 1, type the following command:

```
# vxfenadm -g /dev/device_name
Reading SCSI Registration Keys...
Device Name: device_name
Total Number of Keys: 1
key[0]:
Key Value [Numeric Format]: 65,80,71,82,48,48,48,48
```

The `-g` option of `vxfenadm` displays the eight bytes of a key value in two formats. In the numeric format, the first byte, representing the node ID, contains the system ID plus 65. The remaining bytes contain the ASCII values of the key's letters. In this case, "PGR0000." In the next line, the node ID 0 is expressed as "A" and node ID 1 would be "B."

Disabling I/O fencing

You may have to disable fencing in the following cases:

- The cluster has been upgraded to the latest SFCFS stack and the storage does not support the SCSI-3 PGR feature.
- During installation fencing was turned on but later disabled.

By default, the VxFEN driver operates with I/O fencing enabled. To disable this feature without removing the coordinator disks, you must create the file `/etc/vxfenmode` and include a string within the file to notify the VxFEN driver.

To stop and restart the driver

- 1 To stop and restart the driver, type the following commands:

```
# echo "vxfen_mode=disabled" > /etc/vxfenmode
```
- 2 Additionally, Veritas recommends removing the `/etc/vxfendg` file if fencing is to be later reenabled.

How I/O fencing works during different events

The following table describes how I/O fencing works to prevent data corruption during different failure scenarios. For each event, corrective actions are indicated.

[Table 5-2](#) describes corrective actions to prevent data corruption.

Table 5-2 Corrective actions to prevent data corruption

Event	Node A: What Happens?	Node B: What Happens?	Action
All 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 removes itself from the cluster.	When Node B is ejected from cluster, repair the private networks before attempting to bring Node B back.
All 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.	Reboot 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 on the console about jeopardy and 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 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 reboot Node A.

Table 5-2 Corrective actions to prevent data corruption (*continued*)

Event	Node A: What Happens?	Node B: What Happens?	Action
<p>Nodes A and B and private networks lose power. Coordinator and data disks retain power.</p> <p>Power returns to nodes and they reboot, but private networks still have no power.</p>	<p>Node A reboots 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:</p> <p>Potentially a preexisting split brain. Dropping out of the cluster. Refer to the user documentation for steps required to clear preexisting split brain.</p>	<p>Node B reboots 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:</p> <p>Potentially a preexisting split brain. Dropping out of the cluster. Refer to the user documentation for steps required to clear preexisting split brain.</p>	<p>Refer to section in Troubleshooting chapter for instructions on resolving preexisting split brain condition.</p>
<p>Node A crashes while Node B is down. Node B comes up and Node A is still down.</p>	<p>Node A is crashed.</p>	<p>Node B reboots 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:</p> <p>Potentially a preexisting split brain. Dropping out of the cluster. Refer to the user documentation for steps required to clear preexisting split brain.</p>	<p>Refer to section in Troubleshooting chapter for instructions on resolving preexisting split brain condition.</p>
<p>The disk array containing two of the three coordinator disks is powered off.</p>	<p>Node A continues to operate as long as no nodes leave the cluster.</p> <p>Node A races for a majority of coordinator disks. Node A fails because only one of three coordinator disks is available. Node A removes itself from the cluster.</p>	<p>Node B continues to operate as long as no nodes leave the cluster.</p> <p>Node B leaves the cluster.</p>	<p>Power on failed disk array and restart I/O fencing driver to enable Node A to register with all coordinator disks.</p>

Table 5-2 Corrective actions to prevent data corruption (*continued*)

Event	Node A: What Happens?	Node B: What Happens?	Action
Node B leaves the cluster and the disk array is still powered off.			

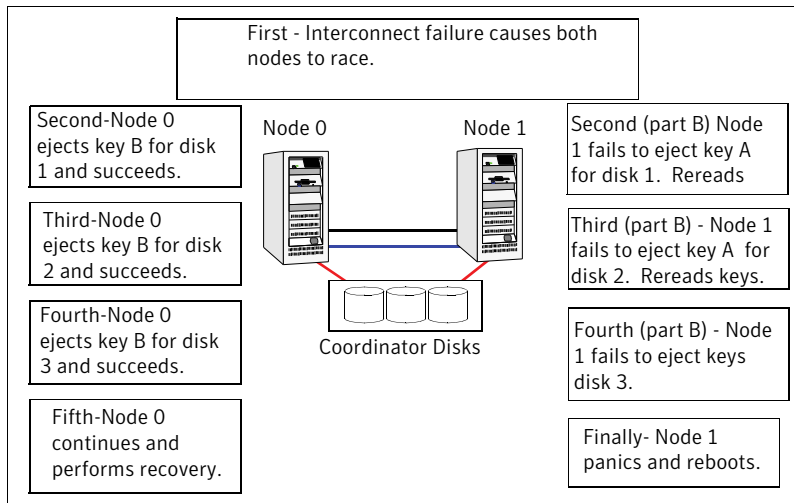
Troubleshooting fenced configurations

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 5-1 shows a two-node cluster in which the severed cluster interconnect poses a potential split-brain condition.

Figure 5-1 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.

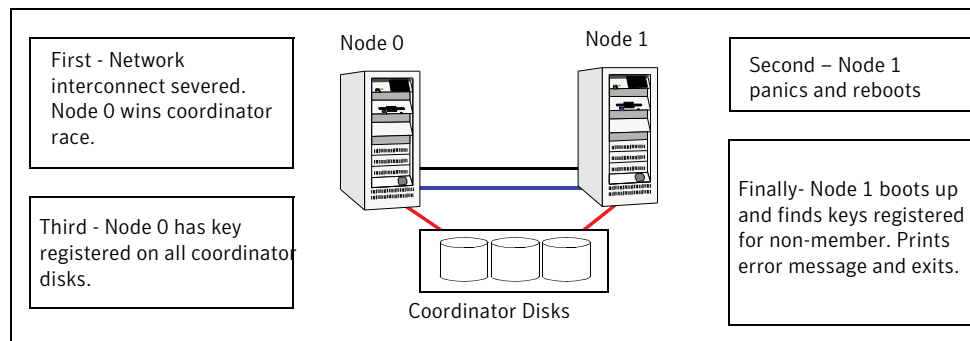
Recovering from a preexisting network partition (split-brain)

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 5-2 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 5-2 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.

Using Veritas Extension for Oracle Disk Manager

This chapter includes the following topics:

- [About Oracle Disk Manager](#)
- [About Oracle Disk Manager and Storage Foundation Cluster Files System](#)
- [About Oracle Disk Manager and Oracle Managed Files](#)
- [Setting up Veritas Extension for Oracle Disk Manager](#)
- [Configuring Veritas Extension for Oracle Disk Manager](#)
- [How to prepare existing database storage for Oracle Disk Manager](#)
- [Verifying that Oracle Disk Manager is configured](#)
- [Disabling the Oracle Disk Manager feature](#)

About Oracle Disk Manager

Veritas Extension for Oracle Disk Manager is specifically designed for Oracle9i 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 Oracle9i 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.

When using Oracle Resilvering, you can turn off VxVM Dirty Region Logging (DRL), which increases performance.

Oracle Disk Manager reduces administrative overhead by providing enhanced support for Oracle Managed Files. Veritas Extension for Oracle Disk Manager is transparent to the user. 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.

Note: Veritas Storage Foundation 4.1 for Oracle was the last major release that supported Oracle Disk Manager for raw devices.

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 Oracle9i 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.

The benefits of using Oracle Disk Manager are:

- 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.

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 Files 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.vxodmfs(8)` man page for more information on its cluster support modes.

About Oracle Disk Manager and Oracle Managed Files

Oracle9i 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 Oracle9i 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 OMF.

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.

This example shows the `init.ora` contents and the command for starting the database instance. To simplify Oracle UNDO management, the new Oracle9i or later `init.ora` parameter `UNDO_MANAGEMENT` is set to `AUTO`. This is known as System-Managed Undo.

```
$ 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 oracle9i dba 96 May  3 15:43 EMP_INDEX/
drwxr-xr-x 2 oracle9i dba 96 May  3 15:43 EMP_TABLE/
-rw-r--r-- 1 oracle9i dba 104858112 May  3 17:28 ora_1_BEhYgc0m.log
-rw-r--r-- 1 oracle9i dba 104858112 May  3 17:27 ora_2_BEhYu4NA.log
-rw-r--r-- 1 oracle9i dba 806912 May  3 15:43 ora_BEahlfUX.ctl
-rw-r--r-- 1 oracle9i dba 10489856 May  3 15:43 ora_sys_undo_BEajPSVq.dbf
-rw-r--r-- 1 oracle9i dba 104861696 May  3 15:4 ora_system_BEaiFE8v.dbf
-rw-r--r-- 1 oracle9i dba 186 May  3 15:03 PROD.ora

./EMP_INDEX:
total 204808
-rw-r--r-- 1 oracle9i dba 104861696 May  3 15:43
ora_emp_inde_BEakGfun.dbf

./EMP_TABLE:
total 204808
-rw-r--r-- 1 oracle9i dba 104861696 May  3 15:43
ora_emp_tabl_BEaklLqK.dbf
```

Setting up Veritas Extension for Oracle Disk Manager

Veritas Extension for Oracle Disk Manager is part of Veritas Storage Foundation for Oracle. Veritas Extension for Oracle Disk Manager is enabled once Veritas Storage Foundation for Oracle and Oracle9i or later are installed, and 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 | <ul style="list-style-type: none">■ Veritas Storage Foundation for Oracle must be installed on your system.■ Oracle9i, 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. (Cached Quick I/O is not available on Linux.) |
| Usage Notes | <ul style="list-style-type: none">■ 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. (Quick I/O is not available on Linux.)■ Oracle uses default file access methods if Oracle9i or later or Veritas Storage Foundation for Oracle is not installed, or VxFS 5.0 is not available in the kernel. |

Configuring Veritas Extension for Oracle Disk Manager

If `ORACLE_HOME` is on a shared file system, run the following commands from any node, otherwise run them on each node.

where `ORACLE_HOME` is the location where Oracle database binaries have been installed.

To configure Veritas Extension for Oracle Disk Manager

- 1 Log in as `oracle`.
- 2 If the Oracle database is running, then shutdown the Oracle database.
- 3 Verify that `/opt/VRTSodm/lib64/libodm.so` exists.
- 4 Link Oracle's ODM library present in `ORACLE_HOME` with Veritas Extension for Oracle Disk Manager library:

For Oracle9i:

- Change to the `$ORACLE_HOME/lib` directory, enter:

```
# cd $ORACLE_HOME/lib
```

- Take backup of `libodm9.so`, enter:

```
# mv libodm9.so libodm9.so.oracle-`date +%m_%d_%y-%H_%M_%S`
```

- Link `libodm9.so` with Veritas ODM library, enter

```
# ln -s ln -s /opt/VRTSodm/lib64/libodm.so libodm9.so
```

For Oracle10g:

- Change to the `$ORACLE_HOME/lib` directory, enter:

```
# cd $ORACLE_HOME/lib
```

- Take backup of `libodm10.so`, enter.

```
# mv libodm10.so libodm10.so.oracle-`date +%m_%d_%y-%H_%M_%S`
```

- Link `libodm10.so` with Veritas ODM library, enter:

```
# ln -s /opt/VRTSodm/lib64/libodm.so libodm10.so
```

For Oracle11g:

- Change to the `$ORACLE_HOME/lib` directory, enter:

```
# cd $ORACLE_HOME/lib
```

- Take backup of `libodm11.so`, enter.

```
# mv libodm11.so libodm11.so.oracle-`date +%m_%d_%y-%H_%M_%S`
```

- Link `libodm11.so` with Veritas ODM library, enter:

```
# ln -s /opt/VRTSodm/lib64/libodm.so libodm11.so
```

- 5 Start the Oracle database.
- 6 To confirm that the Oracle database starts with Veritas Extension for ODM, the alert log will contain the following text:

```
Veritas <version> ODM Library
```

where *<version>* is the ODM library version shipped with the product.

How to prepare existing database storage for Oracle Disk Manager

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.

You must be running Oracle9i or later to use Oracle Disk Manager.

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/lib64/libodm.so` must exist.
 - If you are using Oracle9i, `$ORACLE_HOME/lib/libodm9.so` is linked to `/opt/VRTSodm/lib64/libodm.so`.
 - If you are using Oracle 10g, `$ORACLE_HOME/lib/libodm10.so` is linked to `/opt/VRTSodm/lib64/libodm.so`.
 - If you are using Oracle 11g, `$ORACLE_HOME/lib/libodm11.so` is linked to `/opt/VRTSodm/lib64/libodm.so`.
 - The VRTSdbed license must be valid.
 - The VRTSodm package must be installed.

To verify that Oracle Disk Manager is configured

1 Check the VRTSdbed license:

```
# /opt/VRTS/bin/vxlictest -n "VERITAS Storage Foundation for Oracle" \  
-f "ODM"
```

```
ODM feature is licensed
```

2 Check that the VRTSodm package is installed:

```
# rpm -qa | grep VRTSodm  
VRTSodm-common-5.0.13.00-MP1_RHEL4  
VRTSodm-platform-5.0.13.00-MP1_RHEL4
```

3 Check that `libodm.so` is present.

```
# ls -lL /opt/VRTSodm/lib64/libodm.so  
-rw-r--r-- 1 root sys 14336 Apr 25 18:42  
/opt/VRTSodm/lib64/libodm.so
```

Note: You can also use the `dbed_checkconfig` command, which is installed with Veritas Storage Foundation for Oracle to check these conditions. See the `dbed_checkconfig(1M)` manual page for more information.

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:

```
# lsmod | grep odm
vxodm      203880  1
```

- 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 ## ODM Library,
Version ##
```

Disabling the Oracle Disk Manager feature

Because 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: 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 the `ln` commands to remove the link to the Oracle Disk Manager Library as follows:

For Oracle 11g, enter:

```
# rm ${ORACLE_HOME}/lib/libodm11.so
# ln -s ${ORACLE_HOME}/lib/libodmd11.so \
${ORACLE_HOME}/lib/libodm11.so
```

For Oracle 10g, enter:

```
# rm ${ORACLE_HOME}/lib/libodm10.so
# ln -s ${ORACLE_HOME}/lib/libodmd10.so \
${ORACLE_HOME}/lib/libodm10.so
```

For Oracle9i, enter:

```
# rm ${ORACLE_HOME}/lib/libodm9.so
# ln -s ${ORACLE_HOME}/lib/libodmd9.so \
${ORACLE_HOME}/lib/libodm9.so
```

- 3 Restart the database instance.

Veritas Volume Manager cluster functionality administration

This chapter includes the following topics:

- [About Veritas Volume Manager cluster functionality administration](#)
- [Overview of Cluster Volume Management](#)

About Veritas Volume Manager cluster functionality administration

A cluster consists of a number of hosts or nodes that share a set of disks. The main benefits of cluster configurations are:

Availability

If one node fails, the other nodes can still access the shared disks. When configured with suitable software, mission-critical applications can continue running by transferring their execution to a standby node in the cluster. This ability to provide continuous uninterrupted service by switching to redundant hardware is commonly termed failover.

Failover is transparent to users and high-level applications for database and file-sharing. You must configure cluster management software, such as VCS, to monitor systems and services, and to restart applications on another node in the event of either hardware or software failure. VCS also allows you to perform general administration tasks such as making nodes join or leave a cluster.

Off-host processing Clusters can reduce contention for system resources by performing activities such as backup, decision support and report generation on the more lightly loaded nodes of the cluster. This allows businesses to derive enhanced value from their investment in cluster systems.

The 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 disk configuration and any changes to this is available on all the nodes. When the cluster functionality is enabled, all the nodes in the cluster can share VxVM objects. This chapter discusses the cluster functionality that is provided with VxVM.

For complete information on VxVM and CVM. Online versions of the VxVM documentation set are installed under the `/opt/VRTS/docs` directory.

See the *Veritas Volume Manager Administrator's Guide*

For complete information on VCS. Online versions of the VCS documentation set are installed under the `/opt/VRTS/docs` directory.

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

Overview of Cluster Volume Management

Tightly coupled cluster systems have become increasingly popular in enterprise-scale mission-critical data processing. The primary advantage of clusters is protection against hardware failure. If the primary node fails or otherwise becomes unavailable, applications can continue to run by transferring their execution to standby nodes in the cluster. This ability to provide continuous availability of service by switching to redundant hardware is commonly termed failover.

Another major advantage of clustered systems is their ability to reduce contention for system resources caused by activities such as backup, decision support and report generation. Enhanced value can be derived from cluster systems by performing such operations on lightly loaded nodes in the cluster instead of on the heavily loaded nodes that answer requests for service. This ability to perform some operations on the lightly loaded nodes is commonly termed load balancing.

To implement cluster functionality, VxVM works together with the cluster monitor daemon that is 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 VxVM with support for cluster functionality. 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.

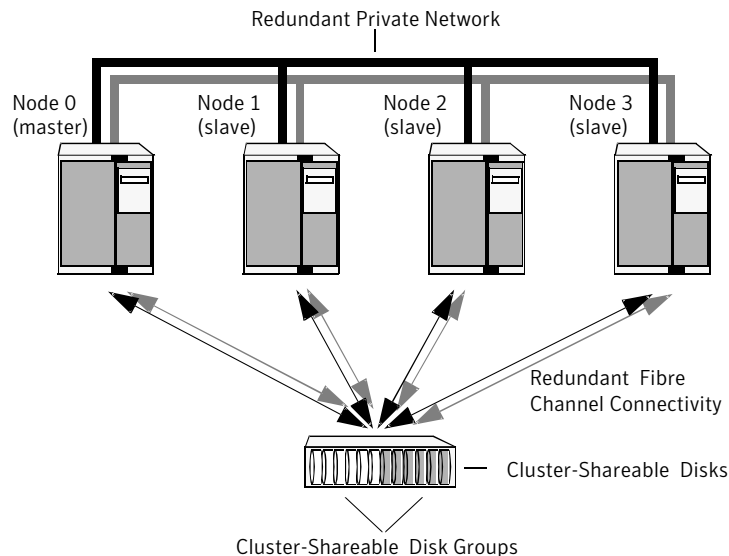
Note: The cluster functionality of VxVM is supported only when used in conjunction with a cluster monitor that has been configured correctly to work with VxVM.

Figure 7-1 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 other 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 7-1 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 that one node act as the master node; all other nodes in the cluster are slave nodes. Any node is capable

of being the master node, and it 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, creating and reconfiguring volumes, and performing snapshot operations.

VxVM designates the first node to join a cluster performs the function of the master node. If the master node leaves the cluster, one of the slave nodes is chosen to be the new master.

Private and shared disk Groups

The following types of disk groups are defined:

Private disk groups Belong 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 access is restricted to one system only. The boot disk group (usually aliased by the reserved disk group name `bootdg`) is always a private disk group.

Shared disk groups 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 `vxchg` command to designate a disk group as cluster-shareable.

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

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 departs 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.

See [Activating modes of shared disk groups](#).

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 a distributed lock manager, for example).

Activating modes of shared disk groups

A shared disk group must be activated on a node for the volumes in the disk group to become accessible for I/O from that node. The ability of applications to read from or to write to volumes is determined by the activation mode of a shared disk group. Valid activation modes for a shared disk group are `exclusive-write`, `read-only`, `shared-read`, `shared-write`, and `off` (inactive).

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

Applications such as high availability and off-host backup can use disk group activation to explicitly control volume access from different nodes in the cluster.

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.

[Table 7-1](#) describes Activation modes for disk groups.

Table 7-1 Activation modes for shared disk groups

Activation mode	Description
exclusive-write (ew)	The node has exclusive write access to the disk group. No other node can activate the disk group for write access.
read-only (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.
shared-read (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.
shared-write (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 7-2 summarizes allowed and conflicting activation modes or shared disk groups:

Table 7-2 Allowed and conflicting activation modes

Disk group activated in cluster as...	exclusive-write	read-only	shared-read	shared-write
exclusive-write	Fails	Fails	Succeeds	Fails
read-only	Fails	Succeeds	Succeeds	Fails
shared-read	Succeeds	Succeeds	Succeeds	Succeeds
shared-write	Fails	Fails	Succeeds	Succeeds

Shared disk groups can be automatically activated in any mode during disk group creation or during manual or auto-import.

To control auto-activation of shared disk groups

- 1 To create the defaults file `/etc/default/vxdg`.
- 2 The defaults file `/etc/default/vxdg` must contain the following lines:

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

The activation-mode is one of `exclusive-write`, `read-only`, `shared-read`, `shared-write`, or `off`.

When enabling activation using the defaults file, it is advisable that the defaults file be identical on all nodes in the cluster. Otherwise, the results of activation are unpredictable.

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.

If the defaults 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.

Note: 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 `vx dg list diskgroup` command.

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

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 7-3](#) describes one of the following policies for resolving this type of discrepancy.

Table 7-3 Policies for resolving this type of discrepancy

Policy	Description
global	The detach occurs cluster-wide (globally) if any node in the cluster reports a disk failure. This is the default policy.

Table 7-3 Policies for resolving this type of discrepancy (*continued*)

Policy	Description
local	In the event of disks failing, the failures are confined to the particular nodes that saw the failure. 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.

The `vxvg` command can be used to set the disk detach and dg fail policy. The `dgfailpolicy` sets the disk group failure policy in the case that the master node loses connectivity to the configuration and log copies within a shared disk group. This attribute requires that the disk group version is 120 or greater.

[Table 7-4](#) describes the supported policies.

Table 7-4 Supported policies

Policies	Description
dgdisable	The master node disables the diskgroup for all user or kernel initiated transactions. First write and final close fail. This is the default policy.
leave	The master node panics instead of disabling the disk group if a log update fails for a user or kernel initiated transaction (including first write or final close). If the failure to access the log copies is global, all nodes panic in turn as they become the master node.

Disk group failure policy

The local detach policy by itself is insufficient to determine the desired behavior if the master node loses access to all disks that contain copies of the configuration database and logs. In this case, the disk group is disabled. As a result, the other nodes in the cluster also lose access to the volume. In release 4.1, the disk group failure policy was introduced to determine the behavior of the master node in such cases.

[Table 7-5](#) illustrates the policy's possible settings.

Table 7-5 Behavior of master node for different failure policies

Type of I/O failure	Leave (dgfailpolicy=leave)	Disable (dgfailpolicy=dgdisable)
Master node loses access to all copies of the logs.	The master node panics with the message “klog update failed” for a failed kernel-initiated transaction, or “cvm config update failed” for a failed user-initiated transaction.	The master node disables the disk group.

The behavior of the master node under the disk group failure policy is independent of the setting of the disk detach policy. If the disk group failure policy is set to `leave`, all nodes panic in the unlikely case that none of them can access the log copies.

Limitations of shared disk groups

The boot disk group (usually aliased as `bootdg`) cannot be made `clustershareable`. 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.

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.

If you have RAID-5 volumes in a private disk group that you wish to make shareable, you must first relayout the volumes as a supported volume type such as `stripe-mirror` or `mirror-stripe`. Online relayout is supported provided that it does not involve RAID-5 volumes.

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`.

Agents for Storage Foundation Cluster File System

This chapter includes the following topics:

- [About agents for Storage Foundation Cluster File System](#)
- [Storage Foundation Cluster File System agents](#)
- [Veritas Cluster Server cluster components](#)
- [Modifying the agents and their resources](#)
- [Storage Foundation Cluster File System administrative interface](#)
- [CFSMount agent](#)
- [CFSfsckd agent](#)
- [CVMCluster agent](#)
- [CVMVolDg agent](#)

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

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 `cfscscluster config` command.

See the *Symantec Cluster Server Bundled Agents Reference Guide*.

PDF versions of this guide are located under the `/opt/VRTS/docs` directory.

This appendix includes the following topics:

- [Storage Foundation Cluster File System agents](#)
- [Veritas Cluster Server cluster components](#)
- [Modifying the agents and their resources](#)
- [Storage Foundation Cluster File System administrative interface](#)

Storage Foundation Cluster File System agents

SFCFS includes the following agents:

- [CFSSMount agent](#)
- [CFSfsckd agent](#)
- [CVMCluster agent](#)
- [CVMVolDg agent](#)

Veritas Cluster Server cluster components

Resources, attributes, and service groups are components integral to cluster functionality.

See the *Symantec 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 119.

The following VCS resource types required for SFCFS functionality are:

- CFSfsckd
- CFSSMount

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 119.

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 CFSSMount resource controls the cluster mount of a shared volume on a specified mount point. CFSSMount 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 CFSSMount 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 120.

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 CFSSMount 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 8-1](#) describes the SFCFS commands.

Table 8-1 SFCFS commands

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

The `cfsccluster` command

The `cfsccluster` 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 `cfsccluster config` command. The `cfsccluster config` command adds all the resource type definitions and adds resource instances, one each of type `CVMCluster` and `CFSfscd`. The `cfsccluster config` command also brings the resources online, and `cfsccluster status` can be used to query the status of VCS.

See [“Resources and service groups for File System cluster functionality”](#) on page 117.

The `cfsscuster 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 `cfsscuster(1M)` manual page.

You must manually take CFSMount resources offline (using the `cfsumount` command) before executing the `cfsscuster 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 name is specified, `cfsmntadm add` creates a default service group, `cfs`. 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 117.

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 `cfsgadm` command

The `cfsgadm` command is the administrative interface for shared disk groups. Using `cfsgadm`, 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 `cfsgadm` command.

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

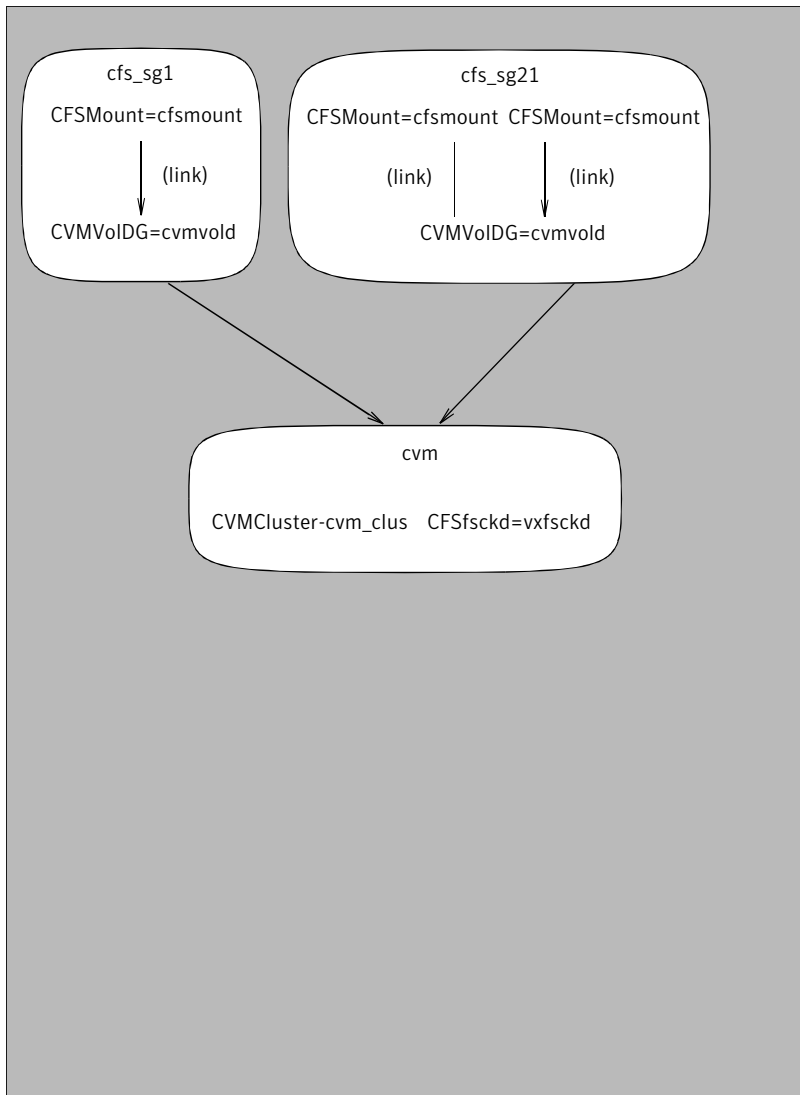
The `cfsmount` and `cfsumount` command

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 8-1](#) describes the SFCFS service groups and resource dependencies.

Figure 8-1 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 = off }
    ActivationMode @system02 = { cfsdg = off }
)
CVMCluster cvm_clus (
    CVMClustName = omcluster
    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/voll"
    NodeList = { system01 , system02 }
    RemountRes @system01 = DONE
    RemountRes @system02 = DONE
)
CVMVolDg cvmvoldg1 (
    Critical = 0
    CVMDiskGroup = cfsdg
    CVMActivation @system01 = off
    CVMActivation @system02 = off
)
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 }
    static str ArgList[] = { CVMDiskGroup, CVMVolume, CVMActivation
}
    str CVMDiskGroup
    keylist CVMVolume
    str CVMActivation
    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 CFSSMount (
    static keylist RegList = { MountOpt, Policy, NodeList, ForceOff,
SetPrimary }
    static int FaultOnMonitorTimeouts = 1

    static int OnlineWaitLimit = 1
    static str ArgList[] = { MountPoint, BlockDevice, MountOpt }
    str MountPoint
    str MountType
    str BlockDevice
    str MountOpt
    keylist NodeList
    keylist Policy

```

```

temp str Primary
str SetPrimary
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 8-2](#) describes the CFSMount agent entry points.

Table 8-2 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 <code>fsclustadm</code> 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 <code>fsclustadm</code> policy on file system.

[Table 8-3](#) describes the CFSMount agent attributes.

Table 8-3 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.
NodeList (required)	string-keylist	List of nodes on which to mount.

Table 8-3 CFSMount agent attributes (*continued*)

Attributes	Type and Dimension	Definition
Policy (optional)	string-scalar	Node list for the primary file system selection policy.
MountOpt (optional)	string-scalar	Options for the <code>mount</code> command. To create a valid MountOpt attribute string: <ul style="list-style-type: none"> ■ Use VxFS type-specific options only. ■ Do not use the <code>-o</code> flag to specify the VxFS-specific options. ■ Do not use the <code>-t vxfs</code> file system type option. ■ The <code>cluster</code> option is not required. ■ Specify options in a comma-separated list as in these examples: <pre>ro ro,cluster blkclear,mincache=closesync</pre>
MountType, Primary, SetPrimary, RemountRes, ForceOff (internal)		Not user configured used only by system.

CFSMount type definition

The CFSMount type definition:

```
type CFSMount (
    static int RestartLimit = 2
    static str LogLevel
    static str ArgList[] = {MountPoint, BlockDevice, MountOpt}
    NameRule = resource.MountPoint
    str MountPoint
    str BlockDevice
    str MountOpt
)
```

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 `cfsccluster config` command.

[Table 8-4](#) describes the CFSfsckd agent entry points.

Table 8-4 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 = 2
    static str LogLevel
    static str ArgList[] = { }
)

```

```
NameRule = ""
)
```

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 CVMMultiNIC resource and must be configured to depend on CVMMultiNIC. 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 8-5](#) describes the CVMCluster agent entry points.

Table 8-5 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 8-6](#) describes the CVMCluster agent attributes.

Table 8-6 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.

Table 8-6 CVMCluster agent attributes (*continued*)

Attributes	Type and Dimension	Definition
CVMTransport (required)	string-association	The CVM transport mode, either <i>gab</i> or <i>udp</i> . For SFCFS, <i>gab</i> is the only valid transport mode.
PortConfigd (required)	integer-scalar	Port number used by CVM for <i>vxconfigd</i> -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 }
    NameRule = ""
    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 (
    Critical = 0
    CVMClustName = vcs
    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 8-8](#) describes the CVMVolDg agent entry points.

Table 8-7 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 8-8](#) describes the CVMVolDg agent attributes.

Table 8-8 CVMVolDg agent attributes

Attributes	Type and Dimension	Definition
CVMDiskGroup (required)	string-scalar	Shared disk group name.
CVMVolume (required)	string-keylist	Shared volume names.
CVMActivation (required)	string-scalar	Activation mode for the disk group. Must be set to shared-write (<code>sw</code>). This is a localized attribute.

CVMVolDg type definition

The CVMVolDg type definition:

```
type CVMVolDg (  
    static keylist RegList = { CVMActivation }  
    static str ArgList[] = { CVMDiskGroup, CVMActivation }  
    NameRule = ""  
    str CVMDiskGroup  
    str CVMActivation  
)
```

Sample of CVMVolDg configuration

This is a sample of CVMVolDg configuration:

```
CVMVolDg testdg (  
    CVMDiskGroup = testdg  
    CVMActivation @system01 = sw  
    CVMActivation @system02 = sw  
)
```

Creating a starter database

This appendix includes the following topics:

- [Creating a database for Oracle 10g or 11g](#)

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
sda auto:none - - online invalid
sdb auto:none - - online invalid
sdc auto:cdsdisk - tempdg online shared
sdd auto:none - ocrvotedg online shared
sde auto:cdsdisk - - online shared
sdf auto:cdsdisk - - online shared
```

The above sample output indicates that shared disks `sde` and `sdf` 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 sde sdf
```

- 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
sda auto:none - - online invalid
sdb auto:none - - online invalid
sdc auto:cdsdisk - tempdg online shared
sdd auto:none - ocrvotedg online shared
sde auto:cdsdisk - - online shared
sdf auto:cdsdisk - - online shared
```

The above sample output indicates that shared disks `sde` and `sdf` are free and can be used for Oracle database tablespaces.

- 4 Create a shared disk group. For example, enter:

```
# vxdbg -s init oradatadg sdd
```

10g or 11g can be used in the example above.

- 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 -t vxfs /dev/vx/rdisk/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 -t vxfs -o cluster /dev/vx/dsk/oradatadg/oradatavol \
/oradata
```

- 9 Set `oracle` as the owner of the file system, and set `755` as the permissions:

```
# chown oracle:oinstall /oradata
# chmod 755 /oradata
```

- 10** On the other node(s), complete steps 7 through 9.
- 11** Create the Oracle database.
See the *Oracle* documentation.

Glossary

access control list (ACL)	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.
Block-Level Incremental Backup (BLI 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	The cluster functionality of Veritas Volume Manager.
CVM Master	The cluster volume manager (CVM) 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, <code>/etc/fstab</code> 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 <code>quotas</code>) must exist in the root directory of a file system for quota-related commands to work. Support for group quotas requires a group quotas file is named <code>quotas.grp</code> .

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.
Heartbeats	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 <code>lltconfig</code> 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 and group.
K	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 <code>mount -o cluster</code> 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	Multi-volume 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.
object location table (OLT)	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 <code>quotas</code> commands read and write the external <code>quotas</code> file to get or change usage limits. When quotas are turned on, the quota limits are copied from the external <code>quotas</code> file to the internal <code>quotas</code> 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 <code>rootdg</code> .
SFCFS	The Veritas 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	The 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

Symbols

/etc/default/vxdg defaults file 110

/etc/default/vxdg file 34

A

activating resources for CFS 119

activation modes for shared disk groups 33–34, 109

adding a new node to a cluster 47

agents

 CFS 42, 116

 CFSfsckd 128

 CFSMount 126, 128

 CVM 116

 CVMCluster 129

 CVMQlogckd 129

 CVMVolDg 131

 modifying 117

 VCS 116

 VCS bundled 116

asymmetric mounts 24, 41

attributes defined 117

C

CFS

 activating resources 119

 agents 42, 116

 applications 18

 cluster mounts 40

 load distribution 27, 44

 memory mapping 15

 modifying resources 119

 overview 40

 primary 41

 secondary 41

 snapshots 26, 45

 supported features 15

 synchronization 27, 44

cfscluster command 42, 119

cfsdgadm command 42, 119

CFSfsckd agent 128

cfsmntadm command 42, 119

CFSMount agent 126, 128

cfsmount command 42, 119

CFSTypes.cf file 125

CFSTypes.cf file example 125

cfsumont command 43, 119

cluster mounted file systems 40

cluster-shareable disk groups in clusters 32

clusters

 activating disk groups 34

 activation modes for shared disk groups 33, 109

 adding a new node 47

 benefits 105

 cluster-shareable disk groups 32

 designating shareable disk groups 32, 108

 limitations of shared disk groups 36, 113

 maximum number of nodes in 106

 nodes 30, 106

 private disk groups 32

 private networks 30, 107

 protection against simultaneous writes 33, 109

 removing a node from 53

 resolving disk status in 35, 111

 shared disk groups 32, 108

 shared objects 109

commands

 cfscluster 42, 119

 cfsdgadm 42, 119

 cfsmntadm 42, 119

 cfsmount 42, 119

 cfsumont 43, 119

configuration files

 CFSTypes.cf 125

 CVMTypes.cf 124

 main.cf 122

 modifying 117

 types.cf 117

coordinator disks

 definition of 82

creating resource and service groups 120

CVM

 agents 116

CVM (*continued*)

- cluster functionality of VxVM 105
- CVM overview 39
- CVMCluster agent 129
- CVMQlogckd agent 129
- CVMTypes.cf file 124
- CVMTypes.cf file example 124
- CVMVolDg agent 131

D

- dependencies of resource and service groups 118
- disable failure policy 112
- disk groups
 - activation in clusters 34
 - cluster-shareable 32
 - defaults file for shared 34, 110
 - designating as shareable 32, 108
 - failure policy 112
 - private in clusters 32
 - shared in clusters 32, 108
- disks
 - resolving status in clusters 35, 111

E

- exclusive-write mode 34, 109
- exclusivewrite mode 33

F

- failover 106
- failure policies 112
- file systems
 - CFSMount agent monitoring 126
- fsclustadm
 - how to determine a cluster file system
 - primary 41, 44
 - how to set a cluster file system primary 41
- fstab file
 - using with SFCFS 44
- fstyp
 - using to determine the disk layout version 15

G

- GAB description 22, 39
- gabtab file description 22, 39
- GLM description 23
- GUI
 - Cluster Manager 45
 - VEA 45

H

- how to determine a cluster file system primary 44
- how to determine a CVM master node 44
- how to determine the disk layout version 15

I

- I/O fencing
 - definition of 81
 - disabling 87
 - scenarios for 88

K

- keys
 - registration keys
 - formatting of 87

L

- leave failure policy 112
- LLT description 22, 39
- llttab file description 22, 39
- load balancing 106
- LUNs
 - using for coordinator disks 83

M

- main.cf file
 - example 122
- managing resource and service groups 117
- master node 31, 108
- modifying agents 117

N

- network partition 30, 107
- nodes
 - in clusters 30, 106
 - maximum number in a cluster 106
- NTP
 - network time protocol daemon 27, 44

O

- off-host processing 106
- OMF 97
 - working with Oracle Disk Manager 97
- Oracle Disk Manager 93
 - benefits 94
 - disabling 104

Oracle Disk Manager (*continued*)
 preparing existing databases for use with 102
 setting up 100
 Oracle Managed Files 97
 working with Oracle Disk Manager 97

P

primary cluster file system 41
 primaryship
 setting with fsclustadm 45
 private disk groups in clusters 32
 private networks in clusters 30, 107

R

read-only mode 34, 109
 readonly mode 33
 registration key
 displaying with vxfenadm 87
 formatting of 87
 registrations
 key formatting 87
 resource and service groups
 creating 120
 defined 117
 dependencies 118
 managing 117

S

service groups 117
 SFCFS
 using commands 43
 shared disk groups
 activation modes 33–34, 109
 in clusters 32, 108
 limitations of 36, 113
 shared-read mode 34, 109
 shared-write mode 34, 109
 sharedread mode 33
 sharedwrite mode 33
 slave nodes 31, 108

T

types.cf file 117

U

upgrade
 from raw devices 102

V

VCS
 attributes 117
 bundled agents 116
 configuration files
 CFSTypes.cf 125
 main.cf 122
 VCS overview 21, 38
 vxdctl
 how to determine a CVM master node 44
 vxfenadm command
 options for administrators 86
 vxfentab file
 created by rc script 84
 vxfentsthdw
 testing disks 84
 VxVM
 cluster functionality (CVM) 105
 limitations of shared disk groups 36, 113
 shared objects in cluster 109