VERITAS™

VERITAS Storage Foundation™ Cluster File System 4.1

Installation and Administration Guide

HP-UX

N16826G

June 2005

Disclaimer

The information contained in this publication is subject to change without notice. VERITAS Software Corporation makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. VERITAS Software Corporation shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this manual.

VERITAS Copyright

VERITAS Software Corporation
350 Ellis Street
Mountain View, CA 94043
USA
Phone 650–527–8000 Fax 650–527–2908
www.veritas.com

Third-Party Copyrights

Data Encryption Standard (DES)

Support for data encryption in VCS is based on the MIT Data Encryption Standard (DES) under the following copyright:

Contents

Preface

This guide describes how to install and upgrade the components of the VERITAS Storage Foundation Cluster File System™ (SFCFS). SFCFS is the clustering functionality of the VERITAS File System™ (VxFS), and is distributed by VERITAS as part of the Storage Foundation and Storage Foundation™ HA (high availability).

This guide is for system administrators who configure and maintain UNIX systems with the VERITAS File System, and assumes that you have a:

- Basic understanding of system administration
- Working knowledge of the UNIX operating system
- General understanding of file systems

How This Guide is Organized

Chapter 1. “Technical Overview” on page 1 provides a technical overview of the VERITAS Storage Foundation Cluster File System (SFCFS).

Chapter 2. “Installing and Configuring” on page 9 provides lists of key terms, software packages, and prerequisites. Also includes instructions on installing and configuring the product and describes licensing requirements.

Chapter 3. “Upgrading from SFCFS 3.5” on page 42 provides instructions for, and examples of, partial and full upgrades. Includes sample output from a two-node cluster.

Chapter 4. “Adding and Removing a Node” on page 45 describes how to add and remove a node to a multinode cluster.

Chapter 5. “Uninstalling” on page 53 describes how to uninstall the SFCFS.

Chapter 6. “SFCFS Architecture” on page 55 defines the roles of component products and presents the concepts of I/O fencing, “split-brain,” jeopardy, and others.

Chapter 7. “SFCFS Administration” on page 71 describes the graphical-user interface VERITAS Enterprise Administrator, or “VEA.” Provides instructions on how to install VEA, and describes several basic procedures you can perform using VEA. This chapter also includes descriptions of the various SFCFS commands.

Chapter 8. “Fencing Administration” on page 83 provides greater detail on the concept of I/O fencing, including how to set up data and coordinator disks, and how to test and troubleshoot your fenced configuration.

Chapter 9. “CVM Administration” on page 97 describes the various SFCFS and CVM agents and their roles.

Chapter 10. “Agents for SFCFS/SFCFS HA” on page 105 describes the agents for the SFCFS.

Appendix A. “Troubleshooting and Recovery” on page 123 provides troubleshooting tips for installation, high-availability, and SCSI.

Conventions

Typeface	Usage	Examples
`monospace`	Computer output, files, directories, software elements such as command options, function names, and parameters	Read tunables from the `/etc/vx/tunefstab` file. See the `vxtunefs`(1M) manual page for more information.
`monospace` (**`bold`**)	User input	# **`mount -F vxfs /h/filesys`**
italic	New terms, book titles, emphasis, variables replaced with a name or value	See the *User's Guide* for details. The variable *vxfs_ninode* determines the value of...

Symbol	Usage	Examples
%	C shell prompt	
$	Bourne/Korn/Bash shell prompt	
#	Superuser prompt (all shells)	
\	Continued input on the following line; you do not type this character	# **`mount -F vxfs`** \ **`/h/filesys`**
[ ]	In a command synopsis, brackets indicates an optional argument	`ls [ -a ]`
|	In a command synopsis, a vertical bar separates mutually exclusive arguments	`mount [ suid | nosuid ]`
blue text	Indicates an active hypertext link	In PDF and HTML files, click on links to move to the specified location

Getting Help

For technical assistance, visit http://support.veritas.com and select phone or email support. This site also provides access to resources such as TechNotes, product alerts, software downloads, hardware compatibility lists, and the VERITAS customer email notification service. Use the Knowledge Base Search feature to access additional product information, including current and past releases of product documentation.

Diagnostic tools are also available to assist in troubleshooting problems associated with the product. These tools are available on disc or can be downloaded from the VERITAS FTP site. See the `README.VRTSspt` file in the `/support` directory for details.

For license information, software updates and sales contacts, visit https://my.veritas.com/productcenter/ContactVeritas.jsp. For information on purchasing product documentation, visit http://webstore.veritas.com.

Technical Overview 1

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.

VERITAS Cluster File System Architecture

Master/Slave File System Design

The VERITAS Cluster File System uses a master/slave, or primary/secondary, architecture to manage file system metadata on shared disk storage. 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. Any CFS file system's metadata, however, is only updated by its CFS primary node (the first node to mount the file system). The CFS primary node is responsible for making all metadata updates and for maintaining the file system's metadata update intent log. Other servers update file system metadata, to allocate new files or delete old ones for example, by sending requests to the primary, which performs the actual updates and responds to the requesting server. This guarantees consistency of file system metadata and the intent log used to recover from system failures.

CFS Failover

If the server on which the CFS 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.

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 any metadata repair. CFS recovery from secondary node failure is therefore faster than recovery from primary node failure.

CFS and the Group Lock Manager

CFS 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 will retrieve 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, no matter from which cluster node 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.

VxFS Functionality on Cluster File Systems

The VERITAS 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:

- 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 following is a list of features and commands that operate on CFS. Every VxFS online manual page has a section on 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. You can also review the *VERITAS Storage Foundation Cluster File System Release Notes* for information on the latest CFS features.

Supported Features

Features and Commands Supported on CFS	
Quick I/O	The Quick I/O for Databases feature, using clusterized Oracle Disk Manager (ODM), is supported on CFS. 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.
Freeze and thaw	Synchronizing operations, which require freezing and thawing file systems, are done on a cluster-wide basis.
Memory mapping	Shared memory mapping established by the `mmap`() function is supported on CFS. See the `mmap`(2) manual page for more information on mapping memory pages.
Disk layout versions	CFS supports only disk layout Version 4, 5 or 6. Cluster mounted file systems cannot be upgraded, but a local mounted file system can be upgraded, unmounted, and mounted again as part of a cluster. Use the `fstyp –v` *special_device* command to ascertain the disk layout version of a VxFS file system. Use the `vxupgrade` command to update the disk layout version.
Locking	Advisory file and record locking are supported on CFS. For the F_GETLK command, if there is a process holding a conflicting lock, the `l_pid` field returns the process ID of the process holding the conflicting lock. The nodeid-to-node name translation can be done by examining the `/etc/llthosts` file or with the `fsclustadm` command. Mandatory locking, and deadlock detection supported by traditional `fcntl` locks, are not supported on CFS. See the `fcntl`(2) manual page for more information on record and file locking.

Unsupported Features

Functionality described as not supported 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 CFS, or to alternate mounting file systems with these options as local and cluster mounts.

Features and Commands Not Supported on CFS

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

Cluster File System Benefits and Applications

Advantages to Using CFS

CFS simplifies or eliminates system administration tasks that result from hardware limitations:

- The CFS single file system image administrative model simplifies administration by making all file system management operations, except resizing and reorganization (defragmentation), independent of the location from which they are invoked.
- You can create and manage terabyte-sized volumes, so partitioning file systems to fit within disk limitations is usually not necessary.
- CFS can support file systems with up to 256 terabyte in size, so only extremely large data farms must be partitioned because of file system addressing limitations.
- Because all servers in a cluster have access to CFS 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 CFS file system can be the primary on any cluster node, the file system recovery portion of failover time in an *n*-node cluster can be reduced by a factor of *n* by distributing the primaryship of 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 CFS 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 CFS

You should use CFS 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. CFS 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 CFS. Applications that are not “cluster-aware” can operate on and access data from anywhere in a cluster. If multiple cluster unaware 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. CFS 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, CFS provides shared data access, enabling application-level load balancing across cluster nodes.

- For single-host applications that must be continuously available, CFS 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, CFS provides shared data to all application instances concurrently. CFS 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 CFS eliminates time consuming and error prone data copying by making files available at all stations.
- For backup, the CFS 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 the CFS:

Using CFS 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 CFS 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 CFS-based cluster greatly simplifies scaling and administration for this type of application.

Installing and Configuring 2

This chapter describes how to install the VERITAS Storage Foundation Cluster File System (SFCFS). SFCFS requires several VERITAS software packages to configure a cluster and to provide messaging services. These packages include the VERITAS Cluster Server (VCS) to monitor systems and application services, VERITAS Low Latency Transport (LLT) and VERITAS Group Membership and Atomic Broadcast (GAB) for communication and messaging, the VERITAS Volume Manager (VxVM) to create the shared volumes necessary for cluster file systems, and the VERITAS File System package.

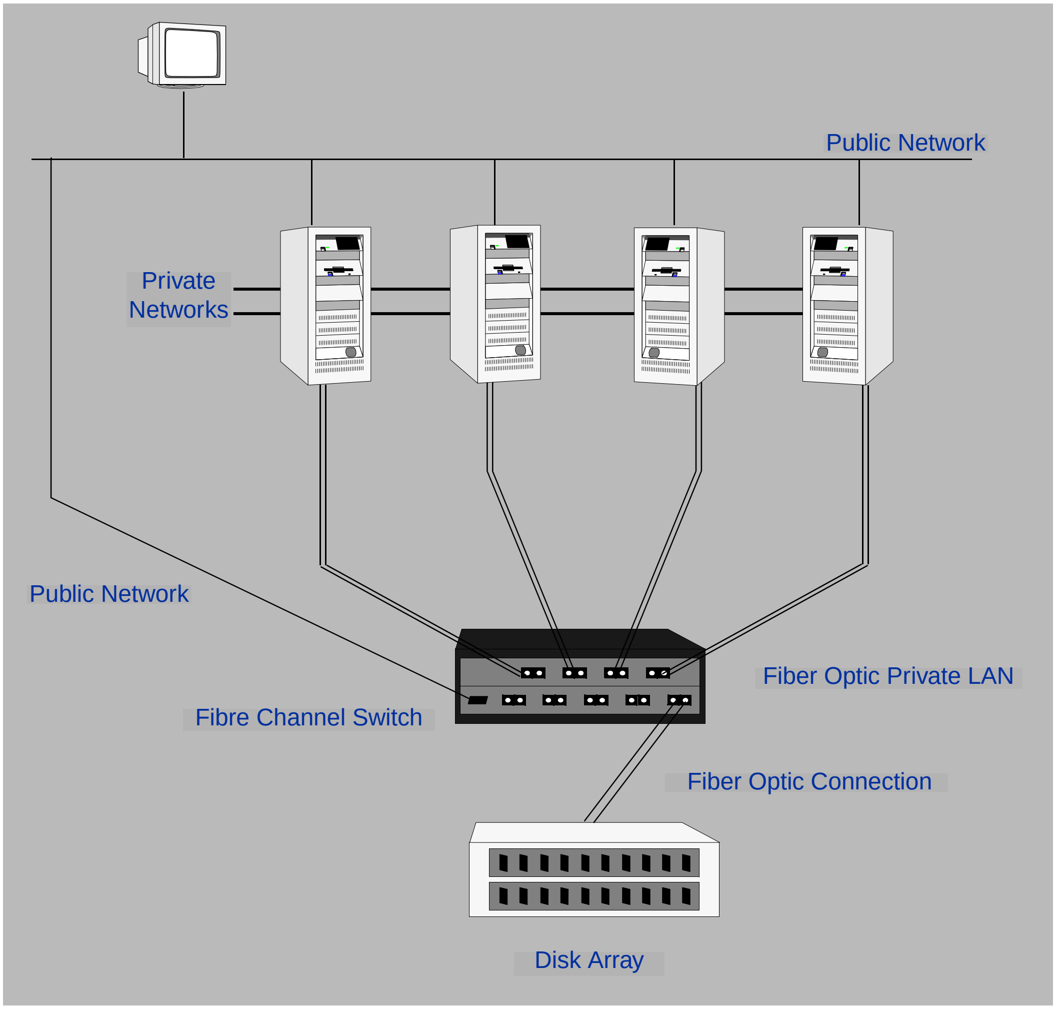
Topics covered in this chapter include:

Hardware Overview

VxFS cluster functionality runs optimally on a *Fibre Channel fabric*. Fibre Channel technology provides the fastest, most reliable, and highest bandwidth connectivity currently available. By employing Fibre Channel technology, CFS can be used in conjunction with the latest VERITAS *Storage Area Network* (SAN) applications to provide a complete data storage and retrieval solution.

The figure below shows the configuration of a cluster file system on a Fibre Channel fabric with a disk array.

Four Node CFS Cluster Built on Fibre Channel Fabric



Shared Storage

Shared storage can be one or more shared disks or a disk array connected either directly to the nodes of the cluster or through a Fibre Channel Switch. Nodes can also have non-shared or local devices on a local I/O channel. The `root` file system is on a local device.

Fibre Channel Switch

Each node in the cluster must have a Fibre Channel I/O channel to access shared storage devices. The primary component of the Fibre Channel fabric is the *Fibre Channel switch*.

Cluster Platforms

There are several hardware platforms that can function as nodes in a cluster file system cluster (see the *Storage Foundation Cluster File System HA Release Notes*). Install the HP-UX 11i 64-bit operating system with the September 2004 HP-UX 11i Version 2.0 on each node and install a Fibre Channel host bus adapter to allow connection to the Fibre Channel switch.

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 systems comprising your cluster are synchronized.

Software Components

SFCFS and SFCFS HA

Storage Foundation (SFCFS) is the name of the VERITAS Cluster File System product and its supporting software packages. Storage Foundation Cluster File System HA (SFCFS HA) contains the SFCFS packages, plus the packages to support the application failover functionality of the VERITAS Cluster Server.

Note SFCFS and SFCFS HA are separately licensed. A SFCFS key will not allow installation of the SFCFS HA packages that provide VCS high availability (see "Additional Packages Installed With SFCFS HA Only" on page 13).

Packages Installed with SFCFS Only

The software packages listed below are required for implementing cluster file system functionality. They are available on the *VERITAS Storage Foundation* software disc in the `depot` directory and are installed on each node in the cluster using the `installer` installation script.

VRTScpi—VERITAS Cross Product Installation Framework

VRTSvlic—VERITAS License Utilities

VRTSperl—VERITAS Perl 5.8.0 Redistribution

VRTSob—VERITAS Enterprise Administrator Service

VRTSat—VERITAS Authentication Service

VRTSllt —VERITAS Low Latency Transport

VRTSgab—VERITAS Group Membership and Atomic Broadcast

VRTSvxfen—VERITAS I/O Fencing

VRTSvcs—VERITAS Cluster Server

VRTSvcsmg—VERITAS Cluster Server Message Catalogs

VRTSvxvm—VERITAS Volume Manager Binaries

VRTSalloc—VERITAS Volume Manager Intelligent Storage Provisioning

VRTSvmpro—VERITAS Volume Manager Management Services Provider

VRTSddlpr—VERITAS Device Discovery Layer Services Provider

VRTSfspro—VERITAS File System Management Services Provider

VRTSvcsvr—VERITAS Cluster Server Agents for VVR

VRTSvrmcsg—MC/ServiceGuard Agent for VVR

VRTSvrpro—VERITAS Volume Replicator Client Extension and Provider for VERITAS Enterprise Administrator

VRTSvxfs—VERITAS File System

VRTSfsman—VERITAS File System Manual Pages

VRTSglm—VERITAS Group Lock Manager

VRTScavf —VERITAS Cluster Server Agents for Cluster File System

Additional Packages Installed With SFCFS HA Only

In addition to the packages listed above, the following are installed with SFCFS HA:

VRTSvcsw—VERITAS Cluster Manager (Web Console)

VRTScscw—VERITAS Cluster Server Configuration Wizards

VRTSvcsag—VERITAS Cluster Server Bundled Agents

VRTSvcsdc—VERITAS Cluster Server Documentation

VRTScutil—VERITAS Cluster Utilities

VRTSjre—VERITAS Java Runtime Environment Redistribution

VRTSweb—VERITAS Java Web Server

Optional SFCFS/SFCFS HA Packages

The following packages are optional for SFCFS / SFCFS HA.

VRTSobgui—VERITAS Enterprise Administrator

VRTSvcsmn—VERITAS Cluster Server Man Pages

VRTScscm—VERITAS Cluster Server Cluster Manager

VRTScssim—VERITAS Cluster Server Simulator

VRTSvmdoc—VERITAS Volume Manager Documentation

VRTSvrdoc—VERITAS Volume Replicator Documentation

VRTSvrw—VERITAS Volume Replicator Web Console

VRTSap—VERITAS Action Provider

VRTStep—VERITAS Task Provider

VRTSfsdoc—VERITAS File System Documentation

Note CFS 4.1 operates only on HP-UX 11i 64-bit operating system with the September 2004 HP-UX 11i Version 2.0. All cluster nodes must be running this OS version.

For cluster file system functionality to work reliably on HP-UX, you must have the required HP-UX patches installed (see “Required Patches” on page 14).

Required Patches

Required patches include the following:

HP-UX Patch ID	Description
PHCO_32385	Enables `fscat`(1M).
PHCO_32387	Enables `getext`(1M).
PHCO_32388	Enables `setext`(1M).
PHCO_32389	Enables `vxdump`(1M).
PHCO_32390	Enables `vxrestore`(1M).
PHCO_32391	Enables `vxfsstat`(1M).
PHCO_32392	Enables `vxtunefs`(1M).
PHCO_32393	Enables `vxupgrade`(1M).
PHCO_32488	Enables LIBC for VxFS 4.1 file system.
PHCO_32523	Enhancement to `quota`(1) for supporting large uids.
PHCO_32524	Enhancement to `edquota` for supporting large uids.
PHCO_32551	Enhancement to `quotaon`/`quotaoff` for supporting large uids.
PHCO_32552	Enhancement to `repquota` for supporting large uids.
PHCO_32596	Enables `df`(1M).
PHCO_32608	Enables `bdf`(1M).
PHCO_32609	Enables `fstyp`(1M).
PHCO_32610	Enables `mount`(1M).
PHCO_32611	Fix `fs_wrapper` to accept “vxfs” from subtype.
PHKL_31500	Sept04 Base Patch
PHKL_32272	Changes to fix intermittent failures in `getacl`/`setacl`.
PHKL_32425	Changes to fix the leaking of VM pages in case of I/O errors.
PHKL_32430	Changes to separate vxfs symbols from libdebug.a, so that VxFS 4.1 symbols are easily available in q4/p4.
PHKL_32431	Changes to disallow mounting of a file system on a vnode having VNOMOUNT set. Enhancements for supporting quotas on large uids.

In addition to the above patches the EnableVXFS41 bundle needs to be installed before installing the SFCFS 4.1. This bundle is a HP bundle and contains enhancements to various commands to understand the new Version 6 layout. The EnableVXFS41 bundle contains the following patches:

HP-UX Patch ID	Description
FSLibEnh	Enhancement to LIBC libraries to understand VxFS disk layout Version 6.
DiskQuota-Enh	Enhancements to various quota related commands to support large uids.
FSCmdsEnh	Enhancements to the mount command to support VxFS 4.1.

Note Install all the latest required HP-UX patches *before* you install SFCFS. You can use the swlist command to determine whether the correct update and patches are installed. The installation procedure terminates if the correct patches are not found.

HP may release patches that supersede the ones in this list. To verify that you have the latest HP-UX patches, go to the VERITAS support website to view the following TechNote:

http://support.veritas.com/docs/275787

Also, you can get the patches from Hewlett-Packard’s Patch Database offered under the Maintenance and Support section of the HP Services & Support - IT Resource Center. HP’s Patch Database provides fast, accurate searches for the latest recommended and superseded patches available for VERITAS File System or VERITAS Volume Manager.

Setting PATH and MANPATH Environment Variables

The software and online manual pages for the packages comprising SFCFS/SFCFS HA are installed in several different directories. However, there are symbolic links to all commands in the /opt/VRTS/bin directory, and symbolic links to all manual pages in /opt/VRTS/man. To make all SFCFS/SFCFS HA commands and manual pages accessible when you do the installation, add /opt/VRTS/bin to your PATH and /opt/VRTS/man to your MANPATH environment variables. Command line examples in this guide assume these environment variables are set.

To prevent conflicts with VxFS manual pages previously installed with JFS / OnLineJFS 3.5, the VxFS 4.1 manual pages are installed in the `/opt/VRTS/vxfs4.1/man` directory. The `/opt/VRTS/vxfs4.1/man` directory is automatically added to `/etc/MANPATH` when the VxFS 4.1 package is installed. Make sure that the `/opt/VRTS/man` directory or the `/opt/VRTS/vxfs4.1/man` directory goes before `/usr/share/man` in you `MANPATH` environment variable so that the latest version of the VxFS manual pages display.

Installing the Product

The product installer is the recommended method to license and install the product. The installer also enables you to configure the product, verify preinstallation requirements, and view the product’s description.

At most points during an installation, you can type **b** (“`back`”) to return to a previous section of the installation procedure. The back feature of the installation scripts is context-sensitive, so it returns to the beginning of a grouped section of questions. If an installation procedure hangs, use **`Control–c`** to stop and exit the program. There is a short delay before the script exits.

The following sample procedure is based on the installation of a VERITAS Storage Foundation Cluster File System HA cluster with two nodes: “system01” and “system02.” If you are installing on standalone systems only, some steps are unnecessary, and these are indicated. Default responses are enclosed by parentheses. Press **`Return`** to accept defaults.

Note If you have obtained a VERITAS product from an electronic download site, the single product download files do not contain the `installer` installation script, so you must use the product installation script to install the product. For example, if you download VERITAS Cluster File System, use the `installcfs` script instead of the `installer` script.

▼ To install the product

1. Log in as superuser.

2. Insert the appropriate media disc into your system’s DVD-ROM drive connected to your system.

3. Create a directory under which to mount the VERITAS CD and mount the CD using the appropriate DVD-ROM drive name. For example:

```
# mkdir /dvdrom
# /usr/sbin/pfs_mount -t rrip /dev/dsk/c3t2d0 /dvdrom
```

4. Run the `installer` command to install the VERITAS SFCFS / SFCFS HA. For example:

   ```
   # cd /dvdrom
   # ./installer
   ```

5. From the Installation menu, choose the **`I`** option for Install and select the **`VERITAS Storage Foundation Cluster File System`**.

6. Enter one or more system names to install SFCFS. For example:

   ```
   Enter the system names separted by spaces on which to install
   SFCFS: system01 system02
   .
   .
   .
   Initial system check completed successfully.

   Press [Return] to continue:
   ```

7. Press **Return** to continue.

   ```
   VERITAS Infrastructure packages installed successfully.

   Press [Return] to continue:
   ```

8. After the VERITAS Infrastructure packages installs successfully, press **Return** to continue. You will be prompted for the SFCFS license key.

   ```
   Enter a SFCFS license key for system01: [?]
   ```

9. Enter the license key. You will be prompted to enter another license key.

   ```
   Do you want to enter another license key for system01?
   [y,n,q,?]
   ```

10. Enter **n** to decline another license key or repeat step 8 and step 9, as prompted.

    ```
    SFCFS licensing completed successfully.

    Press [Return] to continue:
    ```

11. After SFCFS licensing completes successfully, press **Return** to continue.

    ```
    VRTSobgui    VERITAS Enterprise Administrator
    VRTSvcsmn    VERITAS Cluster Server Man Pages
    VRTSvmdoc    VERITAS Volume Manager Documentation
    VRTSvrdoc    VERITAS Volume Replicator Documentation
    VRTSvrw      VERITAS Volume Replicator Web Console
    VRTSap       VERITAS Action Provider
    VRTStep      VERITAS Task Provider
    VRTSfsdoc    VERITAS File System Documentation
    VRTScfsdc    VERITAS Cluster File System Documentation

         1)  Install all of the optional packages
         2)  Install none of the optional packages
         3)  View package descriptions and select optional packages

    Select the optional packages to be installed on all systems?
    [1-3,q,?] (1)
    ```

12. Select the optional packages to be installed on the system(s).

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

13. After the installer displays all the optional packages, press **Return** to continue.

14. After the installation requirements check completes successfully, press **Return** to continue.

15. Enter **y** or **n**, for configuring I/O fencing. For more information on fencing, see *VCS Administration Guide.*

    ```
    Would you like to install Storage Foundation Cluster File System
    on all systems simultaneously? [y,n,q,?] (y)
    ```

16. Enter **y** to install SFCFS on all the systems simultaneously.

17. After SFCFS installation completes successfully, press **Return** to continue.

18. After the entire installation completes, reboot all the system(s) in the cluster.

19. Proceed to “Configuring the Components.” .

Configuring the Components

This sections describes the configuration of SFCFS / SFCFS HA components.

▼ To configure the components

1. Log in as superuser.

2. Run the `installer` command to install the SFCFS HA. For example:

   ```
   # cd /dvdrom
   # ./installer
   ```

3. From the Installation menu, choose the **C** option for Configuration and select the **VERITAS Storage Foundation Cluster File System**.

4. Enter one or more system names to configure SFCFS. For example:

   ```
   Enter the system names separted by spaces on which to configure
   SFCFS: system01 system02
   .
   .
   .
   Initial system check completed successfully.

   Press [Return] to continue:
   ```

5. After the system check completes successfully, press **Return** to continue.

6. After the SFCFS license verification completes successfully, press **Return** to continue.

   ```
   Do you want to stop SFCFS processes? [y,n,q] (y)
   ```

7. Enter **y** to stop the SFCFS processes. You are prompted configuration information.

```
installer will now ask sets of SFCFS configuration-related
questions.

When a [b] is presented after a question, 'b' may be entered to go
back to the first question of the configuration set.

When a [?] is presented after a question, '?' may be entered for
help or additional information about the question.

Following each set of questions, the information you have entered
will be presented for confirmation.  To repeat the set of
questions and correct any previous errors, enter 'n' at the
confirmation prompt.

No configuration changes are made to the systems until all
configuration questions are completed and confirmed.
```

8. Press **Return** to continue. Only for SFCFS HA configuration, you are prompted Configuration for Cluster Manager, SMTP and SNMP.

```
Enter the unique cluster name: [?]
Enter the unique Cluster ID number between 0-255: [b,?]
Enter the NIC for the first private heartbeat link on system01:
[b,?]
Would you like to configure a second private heartbeat link?
[y,n,q,b,?] (y)
Enter the NIC for the low priority heartbeat link on hpslia05:
[b,?] (lan0)
Are you using the same NICs for private heartbeat links on all
systems? [y,n,q,b,?] (y)
.
.
.
Is this information correct? [y,n,q] (y)
```

9. Answer the prompts to configure VCS for SFCFS. You are prompted to enter the CVM cluster reconfiguration timeout.

```
Enter Cluster Volume Manager cluster reconfiguration timeout
(sec): (200)
```

10. Enter CVM cluster reconfiguration timeout in seconds. You are prompted to set up the enclosure-based naming scheme.

```
Do you want to set up the enclosure-based naming scheme?
[y,n,q,?] (n)
```

11. Enter **y** or **n** for the set up of enclosure-based naming scheme.

12. After SFCFS configures successfully, press **Return** to continue.

```
Do you want to start Storage Foundation Cluster File System
processes now? [y,n,q] (y)
```

13. Enter **y** to start SFCFS processes.

14. After done with starting vxfen on target systems, press **Return** to continue.

Note To verify the CFS configuration, enter **hastatus** on any system in cluster and verify that the service groups are `ONLINE`.

```
Do you want to set up the default disk group for each system?
[y,n,q,?] (y)
```

15. Enter **y** or **n** to set up a default disk group for each system.

16. Enter **y** or **n** to specify one disk group name for all eligible systems. You are prompted to specify a default disk group.

```
Specify a default disk group for all systems or type 'l' to
display a listing of existing disk group(s). [?]
```

17. Enter a specific default disk group for all systems or enter **l** to list the existing disk groups.

```
Is this correct? [y,n,q] (y)
```

18. Enter **y**, if this is correct.

19. After the Storage Foundation Cluster File System starts successfully, press **Return** to continue.

20. After pressing Return to continue, verify that you received the following:

```
Configuration of Storage Foundation Cluster File System 4.1 has
completed successfully.
```

Verifying the Configuration Files

You can inspect the contents of the configuration files that were installed and modified after a successful installation process. These files reflect the configuration based on the information you supplied.

▼ **To verify the configuration files**

1. Log in as `root` to any system in the cluster.

2. Set up your environment PATH variable.

   ```
   # export PATH=$PATH:/sbin:/usr/sbin:/opt/VRTS/bin
   ```

LLT Configuration Files

The following files are required by the VCS communication services for Low Latency Transport (LLT).

/etc/llthosts

The file `llthosts`(4) is a database, containing one entry per system, that links the LLT system ID (in the first column) with the LLT host name. This file is identical on each system in the cluster.

For example, the file `/etc/llthosts` contains entries that resemble:

```
0 system01
1 system02
```

/etc/llttab

The file `llttab`(1M) contains information that is derived during installation and used by the utility `lltconfig`(1M). After installation, this file lists the network links that correspond to the specific system.

For example, the file `/etc/llttab` contains entries that resemble:

```
set-node system01
set-cluster 100
link lan1 /dev/lan:1 - ether - -
link lan2 /dev/lan:2 - ether - -
```

The first line identifies the system. The second line identifies the cluster (that is, the cluster ID you entered during installation). The next two lines, beginning with the `link` command, identify the two network cards used by the LLT protocol.

See the `llttab`(4) manual page for details about how the LLT configuration may be modified. The manual page describes the ordering of the directives in the `llttab` file.

Checking LLT Operation

Use the `lltstat` command to verify that links are active for LLT. This command returns information about the links for LLT for the system on which it is typed. Refer to the `lltstat`(1M) manual page for more information. In the following example, `lltstat -n` is typed on each system in the cluster.

▼ To check LLT operation

1. Log into system01.

```
  # lltstat -n
```

Output resembles:

```
  LLT node information:
  Node       State   Links
* 0  system01 OPEN     2
  1  system02 OPEN     2
```

2. Log into system02.

```
  # lltstat -n
```

Output resembles:

```
  LLT node information:
  Node         State   Links
  0  system01 OPEN     2
* 1  system02 OPEN     2
```

Note Each system has two links and that each system is in the OPEN state. An asterisk (*) denotes the system on which the command is typed.

With LLT configured correctly, the output of `lltstat -n` shows all of the systems in the cluster and two links for each system. If the output shows otherwise, you can use the verbose option of `lltstat`. For example, type `lltstat -nvv | more` on a system to view additional information about LLT. In the following example, `lltstat -nvv | more` is typed on a system in a two-node cluster.

3. Log into system01.

```
# lltstat -nvv | more
```

Output resembles:

```
  Node       State      Link    Status     Address
* 0 system01 OPEN
                        lan1    UP         08:00:20:93:0E:34
                        lan2    UP         08:00:20:93:0E:34
  1 system02 OPEN
                        lan1    UP         08:00:20:8F:D1:F2
                        lan2    DOWN       08:00:20:8F:D1:F2
  2          CONNWAIT
                        lan1    DOWN
                        lan2    DOWN
  .
  .
  .
  31         CONNWAIT
                        lan1    DOWN
                        lan2    DOWN
```

Note The output lists 32 nodes. It reports on the two cluster nodes, `system01` and `system02`, plus non-existent nodes. For each correctly configured system, the information shows a state of OPEN, a status for each link of UP, and an address for each link. However, in the example above, the output shows that for node `system02`, the private network may have failed, or the information in `/etc/llttab` may be incorrect.

To obtain information about the ports open for LLT, type `lltstat -p` on any system. In the following example, `lltstat -p` is typed on one system in the cluster.

4. Log into system01.

```
# lltstat -p
```

Output resembles:

```
LLT port information:
Port  Usage    Cookie
0     gab      0x0
      opens:  0 1 3 4 5 6 7 8 9 10 11 12 13...
      connects:   0 1
sys1
```

Note The two systems (0 and 1) are connected.

GAB Configuration Files

The following files are required by the VCS communication services for Group Membership and Atomic Broadcast (GAB).

/etc/gabtab

After installation, the file `/etc/gabtab` contains a `gabconfig`(1M) command that configures the GAB driver for use.

The file `/etc/gabtab` contains a line that resembles:

```
/sbin/gabconfig -c -n N
```

where the `-c` option configures the driver for use and `-n` *N* specifies that the cluster will not be formed until at least *N* systems are ready to form the cluster. *N* is the number of systems in the cluster.

Checking GAB Operation

This section describes how to check GAB operation.

▼ To check GAB operation

1. Enter the following command on each node in the cluster.

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

   If GAB is operational, the following output displays with GAB port membership information:

   ```
   GAB Port Memberships
   ===============================================================
   Port a gen   1bbf01 membership 01
   Port b gen   1bbf06 membership 01
   Port f gen   1bbf0f membership 01
   Port h gen   1bbf03 membership 01
   Port v gen   1bbf0b membership 01
   Port w gen   1bbf0d membership 01
   ```

   If GAB is not operational, the following output display with no GAB port membership information:

   ```
   GAB Port Memberships
   ===============================================================
   ```

   For more information on GAB, refer to the *VERITAS Cluster Server User's Guide*.

Checking Cluster Operation

This section describes how to check cluster operation.

▼ To check cluster operation

1. Enter the following command on any system:

```
# hastatus -summary
```

The output for an SFCFS HA installation resembles:

```
-- SYSTEM STATE
-- System        State     Frozen

A  system01     RUNNING   0
A  system02     RUNNING   0

-- GROUP STATE
-- Group           System    Probed AutoDisabled State

B ClusterService system01    Y       N            ONLINE
B ClusterService system02    Y       N            OFFLINE
```

Note If the `State` value is RUNNING, VCS is successfully installed and running on that node. The group state lists the ClusterService group, which is ONLINE on `system01` and OFFLINE on `system02`. Refer to `hastatus`(1M) manual page. In the *VERITAS Cluster Server User's Guide*, Appendix A, “System States,” describes system states and the transitions between them.

2. Enter the following command on any systems:

```
# hasys -display
```

The example on the next page shows the output of `system01`; the list continues with similar information for `system02` (not shown) and any other systems in the cluster. On each system, the output should be similar.

For more information on the `hasys -display` command, see the `hasys`(1M) manual page. Also refer to the chapter in the *VERITAS Cluster Server User’s Guide*, “Administering VCS From the Command Line.”

```
#System    Attribute           Value
system01   AgentsStopped       0
system01   AvailableCapacity   1
system01   Capacity            1
system01   ConfigBlockCount    54
system01   ConfigCheckSum      29776
system01   ConfigDiskState     CURRENT
system01   ConfigFile          /etc/VRTSvcs/conf/config
system01   ConfigInfoCnt       0
system01   ConfigModDate       Tues June 25 23:00:00 2004
system01   CurrentLimits
system01   DiskHbStatus
system01   DynamicLoad         0
system01   Frozen              0
system01   GUIIPAddr
system01   LLTNodeId           0
system01   Limits
system01   LoadTimeCounter     1890
system01   LoadTimeThreshold   600
system01   LoadWarningLevel    80
system01   MajorVersion        2
system01   MinorVersion        0
system01   NodeId              0
system01   OnGrpCnt            1
system01   ShutdownTimeout     60
system01   SourceFile          ./main.cf
system01   SysName             system01
system01   SysState            RUNNING
system01   SystemLocation
system01   SystemOwner
system01   TFrozen             0
system01   TRSE                0
system01   UpDownState         Up
system01   UserInt             0
system01   UserStr
```

Verifying Agent Configuration

This section describes how to verify the agent configuration.

▼ To verify the agent configuration

Enter the cluster status command from any node in the cluster:

```
# cfscluster status
```

Output resembles:

```
Node              :  system01
Cluster Manager   :  running
CVM state         :  running
No mount point registered with cluster configuration


Node              :  system02
Cluster Manager   :  running
CVM state         :  running
No mount point registered with cluster configuration
```

main.cf File

The VCS configuration file `/etc/VRTSvcs/conf/config/main.cf` is created during the installation procedure. After installation, the `main.cf` file contains the base definitions of the cluster and its nodes. Additionally, the file `types.cf` listed in the include statement defines the bundled agents for VCS resources. The `types.cf` file is also in the directory `/etc/VRTSvcs/conf/config` after installation. Refer to the *VERITAS Cluster Server User's Guide* for more extensive examples of `main.cf` and `types.cf`.

A typical VCS configuration file for CFS file resembles:

```
include "types.cf"
include "CFSTypes.cf"
include "CVMTypes.cf"

cluster cfs_cluster (
        HacliUserLevel = COMMANDROOT
        CounterInterval = 5
        )

system system01 (
        )
system system02
        )
group cvm (
        SystemList = { system01 = 0, system02 = 1 )
        AutoFailOver = 0
        Parallel = 1
        AutoStartList = { system01, system02 )
        )
        CFSfsckd vxfsckd (
                )

        CVMCluster cvm_clus (
                CVMClustName = cfscluster
                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
        //       {
        //       CVMfsckd vxfsckd
        CVMVxconfigd cvm_vxconfigd (
                Critical = 0
                CVMVxconfigdArgs = { syslog }
                )

        cvm_clus requires cvm_vxconfigd
        vxfsckd requires cvm_clus

// resource dependency tree
        //
        //       group cvm
        //       {
        //       CVMfsckd vxfsckd
        //          {
        //          CVMCluster cvm_clus
        //             {
        //             CVMVxconfigd cvm_vxconfigd
        //             }
        //          }
        //       }
```

(For SFCFS HA Only) If you configured VCS Cluster Manager (Web Console), a service group, “ClusterService,” was created that includes IP, Process, and Notifier resources. These resources were configured according to information you provided during the installation procedure. A resource dependency was also created.

Starting VEA

The VERITAS Enterprise Administrator (VEA) is the graphical administrative interface for configuring shared storage devices used for CFS. The VEA GUI server package, `VRTSob`, is installed by the `installer` script on all nodes. To use VEA, the client package, `VRTSobgui`, must also be installed. The VEA client can be a node in the cluster or a remote system. If installed on a remote system, VEA can be used to configure storage for multiple clusters. See the *VERITAS Volume Manager Installation Guide* for more information on installing and starting the VERITAS Enterprise Administrator.

The information on VEA presented in the following procedure is sufficient to create a shared disk group, a shared concatenated volume, and a cluster file system. For detailed information about using VEA, see the *VERITAS Volume Manager User s Guide – VERITAS Enterprise Administrator.*

1. Choose a node or remote system and install the VEA client package (see “Verifying the Configuration Files” on page 22 for information on the CD mounting procedure):

   ```
   # swinstall –s /dvdrom/depot VRTSobgui
   ```

2. To start the VEA GUI, enter:

   ```
   # /opt/VRTS/bin/vea &
   ```

   The VEA application will open and you can begin creating disk groups.

Creating a Dynamic (Shared) Disk Group

In the VEA GUI, use the **New Dynamic Disk Group** procedure on the CVM master node to create a dynamic disk group. Click the **Cluster Nodes** icon to find which cluster node is the master. Disks must be placed in dynamic disk groups before they can be used by CVM. The default disk group (`rootdg`) is usually created during VxVM installation and always exists on a system running VxVM. You can create additional dynamic disk groups to organize disks into logical sets. You cannot make `rootdg` a shared disk group.

When you place a disk under VxVM control, the disk is initialized. Initialization destroys any existing data on the disk.

▼ To create a new dynamic disk group

1. Select the Disk Groups folder, or select a disk.

2. Select **Actions > New Dynamic Disk Group**. The New Dynamic Disk Group wizard screen appears.

3. Click **Next** to continue. A new screen appears.

4. Enter a name for the dynamic disk group.

5. Check the **Create Cluster Group** checkbox.

6. Select **Activation Mode** and choose the mode from the displayed submenu. Click **OK.**

7. Select the shared disks you want to include in the group. Make sure the disks you want to include are in the right pane of the window, then click **Next**.

8. The next screen confirms the disks you have selected. Choose **Yes** to continue if the the disk selection is correct. If you want to change your selected disks, click the **No** button to go back to the previous screen.

   Typically you add all the disks you want in the group at this point. You can add more disks later with the **Add Disk to Dynamic Disk Group** command.

9. The next screen displays the specifications. Click **Finish** to create the shared disk group.

Notes
The **New Dynamic Disk Group** task requires a dynamic disk group name, the name of at least one disk, and must be done on the CVM master node. The dynamic disk group name must be unique. The new dynamic disk group must contain at least one disk. Only disks that are online and do not belong to a dynamic disk group can be used to create a dynamic disk group. The Volume Manager disk name must be unique within the dynamic disk group. If multiple disks are specified in the Disk Device(s) field and only one disk name is specified in Disk Name(s) field, VxVM appends numbers to the disk name so that each disk name is unique within its dynamic disk group.

Creating a Dynamic (Shared) Volume

▼ To create a dynamic volume, on the CVM master node

1. Right-click a dynamic disk group in the tree view of the VEA right pane and select **New Volume** from the context menu. The Create Volume wizard appears.

   You can also select the command from the **Actions** menu or click the **New Volume** tool on the toolbar (the third tool from the left side of the toolbar).

Note The Activation Mode must be set to SW (shared write).

2. Click **Next** to continue. The screen for selecting attributes comes up. Enter a Group name for the volume from the pull-down list.

3. Type in the Volume Name. This is a Volume Manager-specific name that is used in some Volume Manager commands. It is different from the volume label for the file system.

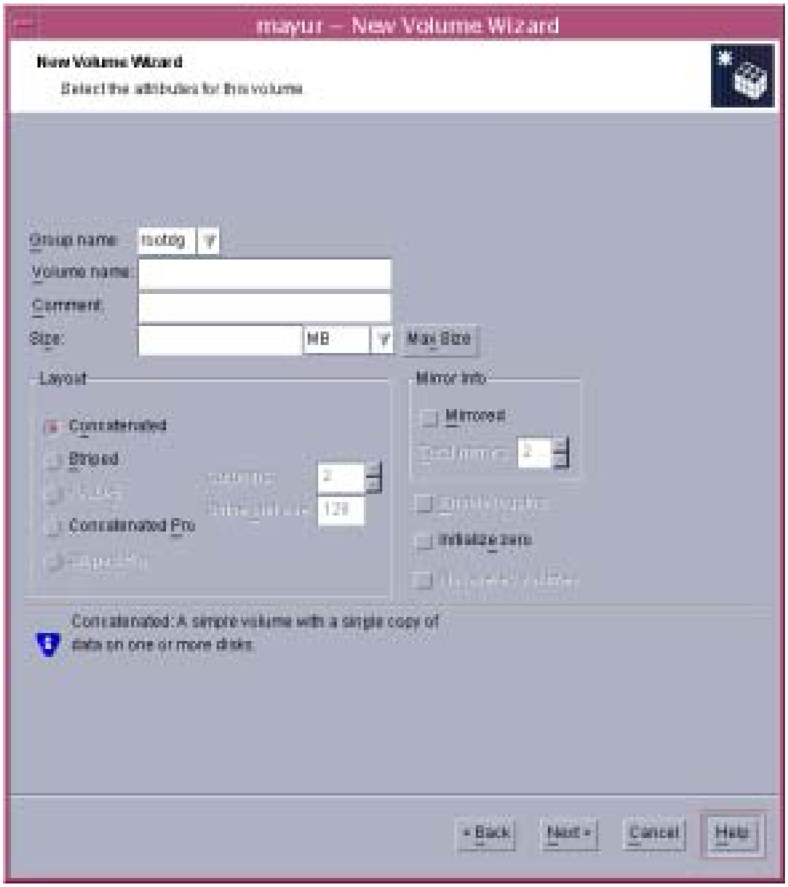
4. Specify the volume size, or select **Maxsize**. The **Maxsize** button works differently, depending on whether a disk is selected.

   If you do not have a disk selected when you start the **Create Volume** wizard, no figure displays in the volume size box. If you then click on a volume type and also click the **Maxsize** button, a size appears in the volume size box that represents the maximum volume for that layout for all disks in the dynamic group.

   If you do have a disk selected when you start the **Create Volume** wizard, a size is shown in the volume size box that represents the maximum concatenated (simple or spanned) volume size on the selected disk. If you then click the **Maxsize** button, a new number appears in the volume size box that represents the maximum size for a spanned volume that spans all the disks in the dynamic group. You can also click another volume layout and then click the **Maxsize** button to get the maximum size for that layout that involves unallocated space on all disks in the dynamic group.

5. Select the Concatenated volume type.

Note You can choose any type for shared volumes except RAID-5.



mayur – New Volume Wizard

New Volume Wizard

Select the attributes for this volume

Group name: rootdg

Volume name:

Comment:

Size: MB

Max Size

Layout

● Concatenated

○ Striped

○ Concatenated Pro

Mirror Info

□ Mirrored

□ Initialize zero

Concatenated: A simple volume with a single copy of data on one or more disks.

6. Select one or more shared disks in the **Select disks to use for volume** screen.

   The default setting is for Volume Manager to assign the disks for you. To manually select the disks, click the **Manually select disks to create volume** radio button. If you select disks manually, the disks that you select will be displayed in the right pane when you click **Next**.

7. You can create a file system at this time by clicking the **Cluster Mount** box under **Mount File System Details**.

8. Check your selections in the final screen and click **Finish** to create the volume. By clicking the **Previous** button, you can go back and make changes before you click **Finish**.

Notes
Deleting a volume using VEA does not remove the CFS entries from the `main.cf` configuration file for that volume. After deleting a volume using VEA, select the file system and Remove Cluster Mount. The `main.cf` configuration file is not updated when VEA is used to change volume or disk group names using the Rename and Disk Group Split and Join actions. If you select disks manually, you can specify that the volume is to be mirrored or striped across controllers, trays, targets, or enclosures. You can also select ordered allocation. Ordered allocation uses the specified storage to first concatenate disks, then to form columns, and finally to form mirrors.

Concatenated

A concatenated volume consists of one or more regions of the specified disks. You have the option of placing a file system on the new volume or mirroring the volume. You can create a regular concatenated volume or a concatenated pro volume. A concatenated pro volume is layered and mirrored.

Layout:	Choose **Concatenated** or **Concatenated Pro** for the volume layout.
Options:	- To mirror the volume, select **Mirrored**. In the Total Number of Mirrors field, enter the total number of mirrors for the volume. **Note** Concatenated pro volumes are mirrored by default. To enable logging for a mirrored volume, select **Enable Logging**. - To place a cluster file system on the volume, click **Mount File System Detail**, specify the VxFS file system type, and add the name of the cluster mount. - To clear the volume before enabling it for general use, select **Initialize Zero**. - To prevent the creation of a layered volume, select **No Layered Volumes**. In cases where a layered volume layout is appropriate, VxVM can create a layered volume when a non-layered layout is specified. This option ensures that the volume has a non-layered layout. If a layered (Pro) layout is selected, this option is ignored.

Notes
If no disks are assigned, VxVM uses available space on disks in the selected disk group. The data in a concatenated volume is not protected against disk failure unless the volume is mirrored. At least one additional disk is required to mirror a concatenated volume. Concatenated pro volumes are mirrored by default, so a concatenated pro volume requires more disks than an unmirrored concatenated volume. You cannot use a concatenated pro volume for a `root` volume. You cannot use a concatenated pro volume for a `swap` volume.

Command Line Examples

You can also create shared volumes, create shared disk groups, and mount cluster file systems from the command line or using a script as shown in the examples below.

Creating a Shared Disk Group from the Command Line

You can use the following script to create a new shared disk group, for example, "`cfsdg`," and add the disks to it. Fill in the name of your shared disk group, `shared_dg_name`, and the list of devices and controllers, `shared_device_list`. If you want to create more disk groups, you must run this script for each.

```
#!/usr/bin/sh -x
#
export PATH=$PATH:/usr/sbin
#
# Name of the Shared disk group
#
shared_dg_name="cfsdg"
#
# List of shared devices to be part of the shared disk group
#
shared_device_list="c2t0d0 c2t1d0 c2t2d0 c2t3d0 c2t4d0
c1t6d0 c1t8d0 c3t4d0"

first="yes"
count=0

for i in $shared_device_list; do
  /etc/vx/bin/vxdisksetup $i
  vxdisk online $i
  vxdisk -f init $i
  count=‘expr $count + 1‘

  if [ $first = "yes" ]; then
     vxdg -s init $shared_dg_name $shared_dg_name$count=$i
     first="no"
  else
     vxdg -g $shared_dg_name adddisk $shared_dg_name$count=$i
  fi
done
```

Creating a Shared Volume from the Command Line

Create a shared volume on the CVM master. In this example, the volume is 20 MB in size:

```
# vxassist -g cfsdg make vol1 20M
```

Activate the shared disk group for shared write access on all nodes:

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

Then check the configuration:

```
# vxprint
TY NAME      ASSOC   KSTATE  LENGTH  PLOFFS  STATE   TUTIL0  PUTIL0
v  vol1      fsgen   ENABLED 40960   -       ACTIVE  -       -
pl vol1-01   vol1    ENABLED 41760   -       ACTIVE  -       -
sd cfsdg     vol1-01 ENABLED 41760   0       -       -       -
```

Creating a Cluster File System from the Command Line

On any cluster node, create a file system on the shared storage volume:

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

If you do not specify *size*, the complete volume is used for the file system.

Mounting a Cluster File System in Shared Mode from the Command Line

To mount the file system on all nodes in the cluster, enter.

```
# cfsmntadm add cfsdg vol1 /mnt all=rw
# cfsmount /mnt
```

See the `cfsmount`(1M) and `cfsmntadm`(1M) for information on adding and mounting a shared file system.

VCS Application Failover Services

If you installed SFCFS HA, you can begin implementing the application monitoring failover services provided by the VERITAS Cluster Server. Information about setting up VCS services is beyond the scope of this document. For complete information on VCS, refer to the VCS documentation set in the `/opt/VRTSvcsdc` directory installed previously by the `installsfcfs` script. It is advisable to begin with the *VERITAS Cluster Server User's Guide*, `/opt/VRTSvcsdc/vcs_ug.pdf` and the *VERITAS Cluster Server Installation Guide*, `/opt/VRTSvcsdc/vcs_ig.pdf`. Also review the *VERITAS Cluster Server Release Notes* on the *VERITAS Storage Foundation 4.1 for HP-UX Disc* .

Upgrading 3

This chapter describes how to upgrade from VERITAS Storage Foundation Cluster File System (SFCFS) 3.5 HP-UX 11i Verision 1 to SFCFS 4.1 HP-UX 11i Verision 2.

Topics covered in this chapter include:

- Upgrade requirements
- Upgrading SFCFS 3.5

Upgrade Requirements

The following documentation is required:

- *VERITAS Cluster Server Installation Guide*
- Hewlett Packard HP-UX Operating System documentation

Upgrading from SFCFS 3.5

The following procedure upgrades SFCFS 3.5 HP-UX 11i Version 1 to SFCFS 4.1 HP-UX 11i Version 2.

▼ To upgrade from SFCFS 3.5

1. Log in as superuser.

2. Save the configuration files, such as `main.cf`, `types.cf`, `CFSTypes.cf`, and `CVMTypes.cf`. For more information on how to save VERITAS Cluster Server (VCS) configuration files, see the *VERITAS Cluster Server Installation Guide*.

3. Remove the `VRTScavf` and `VRTSglm` 3.5 packages from your system:

   ```
   # swremove VRTScavf VRTSglm
   ```

4. Uninstall VCS 3.5 from your system. For more information on how to uninstall VCS, see the *VERITAS Cluster Server Installation Guide*.

   ```
   # cd /opt/VRTSvcs/install
   # ./uninstallvcs
   ```

5. Upgrade the operating system from HP-UX 11i Version 1 to 11i Version 2. See the HP-UX Operating System documentation for more information.

6. Install SFCFS 4.1. See “Installing and Configuring” on page 9.

7. Copy the saved configuration files in step 2 into the `/etc/VRTSvcs/conf/config` directory. For more information on how to restore VCS configurations files, see the *VERITAS Cluster Server Installation Guide*.

8. Start LLT and GAB:

   ```
   # /opt/VRTS/bin/lltconfig -c
   # /sbin/gabconfig -c
   ```

9. Start vxfen. vxfen can be started either in disabled or enabled mode. For more information on vxfen configuration, see *VERITAS Cluster Server Installation Guide*.

10. Add the vxconfigd resource and delete qlogckd from the VCS configuration files:

```
# /opt/VRTS/bin/hastart
# /opt/VRTS/bin/haconf -makerw
# /opt/VRTS/bin/hatype -add CVMVxconfigd
# /opt/VRTS/bin/hares -add cvm_vxconfigd CVMVxconfigd cvm
# /opt/VRTS/bin/hares -modify cvm_vxconfigd Enabled 1
# /opt/VRTS/bin/hares -delete qlogckd
# /opt/VRTS/bin/haconf -dump -makero
```

11. Start VCS:

```
# /opt/VRTS/bin/hastop -all -force
# /opt/VRTS/bin/hastart
```

Adding and Removing a Node 4

This chapter provides information on how to add a node to an existing cluster and removing a node from a cluster. Topics include:

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 system to the cluster using private networks and attach to any shared storage. Then install the required OS software (review the information under "Cluster Platforms" on page 11).

▼ To add a node to a cluster

1. Log in to the new system as superuser.

2. Determine the appropriate DVD-ROM device name. Enter:

   ```
   # ioscan -fn -C disk
   ```

3. Run the following commands to start PFS (Portable File System):

   ```
   # nohup pfs_mountd &
   # nohup pfsd &
   ```

4. Create a directory under which to mount the VERITAS CD and mount the CD using the appropriate DVD-ROM drive name. For example:

   ```
   # mkdir /dvdrom
   # /usr/sbin/pfs_mount -t rrip /dev/dsk/c3t2d0 /dvdrom
   ```

5. Add `/opt/VRTS/bin` to your `PATH` and `/opt/VRTS/man` to your `MANPATH` environment variables "Setting PATH and MANPATH Environment Variables" on page 15.

6. Change to the SFCFS directory.

   ```
   # cd storage_foundation_cluster_file_system
   ```

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

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

8. Enter the system name of the new node to install SFCFS. For example,

   ```
   Enter the system names separted by spaces on which to install
   SFCFS: system03
   .
   .
   .
   Press [Return] to continue:
   ```

9. After the system check completes successfully, press **`Return`** to continue.

10. After the infrastructure packages install successfully, press **`Return`** to continue.

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

11. Enter a SFCFS license key. You are prompted to enter another license key.

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

12. Enter **`y`** or **`n`** for another license key.

13. After SFCFS licensing completes successfully, press **`Return`** to continue. You are prompted to install the optional SFCFS packages.

    ```
    installsfcfs can install the following optional SFCFS packages:

      VRTSobgui VERITAS Enterprise Administrator
      VRTSvcsmn VERITAS Cluster Server Man Pages
      VRTSvmdoc VERITAS Volume Manager Documentation
      VRTSvrdoc VERITAS Volume Replicator Documentation
      VRTSvrw VERITAS Volume Replicator Web Console
      VRTSap VERITAS Action Provider
      VRTStep VERITAS Task Provider
      VRTSfsdoc VERITAS File System Documentation
      VRTScfsdc VERITAS Cluster File System Documentation
      1) Install all of the optional packages
      2) Install none of the optional packages
      3) View package descriptions and select optional packages

      Select the optional packages to be installed on all systems?
      [1-3,q,?] (1)
    ```

14. Select the optional packages to be installed on the systems.

15. After the installer displays all the optional packages, press **`Return`** to continue.

16. After the installation requirements completes successfully, press **`Return`** to continue.

17. Enter **`y`** or **`n`**, for configuring I/O fencing. For more information on fencing, see the *VERITAS VCS Administration Guide* on the software disc.

    ```
    Would you like to install Storage Foundation Cluster File System
    on all systems simultaneously? [y,n,q,?] (y)
    ```

18. Enter **`y`** to install SFCFS on all the systems simultaneously.

19. After SFCFS installation completes successfully, press **Return** to continue.

20. Create the /etc/llthosts file on the new node and modify /etc/llthosts on the other nodes to add the new node name. See "/etc/llttab" on page 22. Following the previous convention in this guide, add the name system03:

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

    The /etc/llthosts files on all the nodes must be the same.

21. Create the /etc/llttab file the same as it looks on another node in the cluster. See "/etc/llthosts" on page 22. Change the set-node line to the name of the new node 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 234
    link lan2 /dev/lan:2 - ether - -
    link lan0 /dev/lan:0 - ether - -
    start
    ```

22. Create /etc/gabtab file the same as it is on another node in the cluster. See "/etc/gabtab" on page 25. For example,

    ```
    /sbin/gabconfig -c -n 3
    ```

    There is no need to reboot the other nodes, just update the /etc/gabtab file on the other nodes in the cluster.

23. Reboot the system that has been added.

    ```
    # /usr/sbin/shutdown -R -y 0
    ```

24. Enable vxconfigd daemon on the system that has been added.

    ```
    # vxinstall
    ```

25. After enabling vxconfigd daemon, proceed to "Configuring CFS and CVM Agents on the New Node." .

Configuring CFS and CVM Agents on the New Node

After rebooting the new system, you must configure the CFS and CVM agents on it. Use the command line procedure below.

▼ **To configure CFS and CVM agents on the new node**

1. Start HAD and `vxfen` on system03.

   a. Use `hastart` on system03 for starting HAD.

   b. For starting `vxfen`, run the following commands on system03:

   ```
   # echo vxfen_mode=disabled > /etc/vxfenmode
   # /sbin/init.d/vxfen start
   ```

2. Check that there are no service groups dependent on CVM, such as CFS, that are still online:

   ```
   # hagrp -dep cvm
   ```

3. If there are any dependencies, take them offline, then take the CVM service group offline:

   ```
   # hagrp -offline cvm -sys system01
   # hagrp -offline cvm -sys system02
   ```

4. Open the VCS configuration for writing:

   ```
   # haconf —makerw
   ```

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

   ```
   # hagrp —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`.

6. Add the new node to the CVM AutoStartList:

   ```
   # hagrp —modify cvm AutoStartList system01 system02 system03
   ```

7. 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
   ```

8. Write the new VCS configuration to disk:

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

9. Put the CVM resources back online, in the following order:

   ```
   # hagrp -online cvm -sys system01
   # hagrp -online cvm -sys system02
   # hagrp -online cvm -sys system03
   ```

10. 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
    ```

11. Add shared disk groups to the cluster configuration:

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

12. Create a `/mnt` on system03 and run the following commands:

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

    Refer to `cfsmntadm` man page for more details.

13. Use `cfsmount` command to cluster mount `/mnt` back on all the nodes:

    ```
    # 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 system `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 root on a node other than system03.

2. Stop all the cluster components on `system01`:

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

3. Open the VCS configuration for writing:

   ```
   # haconf -makerw
   ```

4. Remove `system03` from the system list attribute of the CVM and CFS service groups:

   ```
   # hagrp -modify service_group SystemList -delete system03
   # hagrp -modify cvm SystemList -delete system03
   ```

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

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

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

   repeat step 3 and the command that failed in step 4.

5. Write the new VCS configuration to disk:

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

6. Remove the following files on `system03`:

   ```
   # rm /etc/llthosts
   # rm /etc/llttab
   # rm /etc/gabtab
   ```

7. Reboot `system01`:

   ```
   # /usr/sbin/shutdown -R -y 0
   ```

8. Change to the `install` directory:

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

9. From the `scripts` directory, run the deinstallation script:

```
# ./uninstallsfcfs
```

If you do not want to remove the VERITAS Cluster Server software, enter “**N**” (no) when prompted to uninstall VCS. For complete information on uninstalling VCS, refer to the *VERITAS Cluster Server Installation Guide*, on the software disc.

Uninstalling 5

Uninstalling SFCFS HA

If you need to uninstall SFCFS HA software. Use the uninstallation script.

▼ To uninstall SFCFS HA

1. Log in as `root`.

2. Stop VCS on all nodes. For example:

   ```
   # hastop -all
   ```

3. Run the `uninstallsfcfs` command to uninstall SFCFS. For example:

   ```
   # cd /opt/VRTS/install
   # ./uninstallsfcfs
   ```

4. Enter the system names to unstall SFCFS. For example,

   ```
   Enter the system names separated by spaces on which to uninstall
   SFCFS: system01 system02
   .
   .
   .
   Are you sure you want to uninstall SFCFS packages? [y,n,q] (y)
   ```

5. Enter **`y`** to uninstall the SFCFS packages.

6. After the Storage Foundation Cluster File System package uninstall completes successfully, press **`Return`** to continue.

7. After the entire uninstallation completes, reboot all the system(s) in the cluster.

   ```
   # /usr/sbin/shutdown -R -y 0
   ```

SFCFS Architecture 6

The Role of Component Products

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.

VCS

GAB and 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 for additional information.

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 for details on how to modify the LLT configuration. For more information on GAB or LLT, 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. Examples of port memberships include:

`port a`	heartbeat membership
`port b`	I/O fencing membership
`port f`	Cluster File system membership
`port u`	Temporarily used by CVM
`port v`	Cluster Volume Manager membership

CVM

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.

About CFS

SFCFS uses a master/slave, or *primary/secondary*, architecture to manage file system metadata on shared disk storage. The first node that mounts is the primary node and the remaining nodes are secondaries. Secondaries send requests to the primary to perform metadata updates.The primary node updates the metadata and maintains the file system's metadata update intent log. Data can also be updated from any node directly to shared storage. Other file system operations, such as allocating or deleting files, can originate from any node in the cluster.

If the server on which the SFCFS primary is running fails, the remaining cluster nodes elect a new primary. See "Distributing Load on a Cluster" on page 60 for details on the election process. 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.

SFCFS 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.

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`). 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. See the `mount_vxfs`(1M) man page for details on the cluster read-write (`crw`) mount option.

		Secondary		
		ro	**rw**	**ro, crw**
Primary	**ro**	✔		
	rw		✔	✔
	ro, crw		✔	✔

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

CFS 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.

SFCFS 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 snapshot file system for backups increases the load on the system because of the resources used to perform copy-on-writes and to read data blocks from 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.

There are several characteristics of a cluster snapshot, including:

- 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* for details.

Synchronizing 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.

Distributing Load on a Cluster

Because the primary node on the cluster performs all metadata updates for a given file system, it makes sense to distribute this load by designating different nodes to serve as primary for each of the cluster file systems in use. Distributing workload in a cluster provides performance and failover advantages. Because each cluster mounted file system can have a different node as its primary, SFCFS enables easy load distribution.

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.

You can also 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.

A node can be mounted using the `mount –seconly` option, the result of which the node does not assume the primary mode for the file system. If a primary node fails and the remaining nodes are have the file systems mounted as secondary-only, the file system is disabled. The `mount –seconly` option overrides the `fsclustadm` command. See the `mount_vxfs`(1M) man page for details on the `–seconly` option.

Note that recovery for the primary takes more time than the secondary. Distributing primaries across the cluster would provide a uniform recovery time in the case of a node failure.

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. VERITAS 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. Note that in certain scenarios, the possibility of data corruption remains. For example:

- 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.

Recovering 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. For a detailed explanation of this topic, 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. For more information see "Fencing Administration" on page 83.

Single Network Link and Reliability

Certain environments may prefer using a single private link or a pubic 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. Note that these nodes start in jeopardy state, as described in "I/O Fencing" on page 83. I/O fencing is used to handle split-brain scenarios. The option for single network is given during installation.

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

```
# lltconfig -d device -t tag
```

▼ To remove a link

```
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 CVM

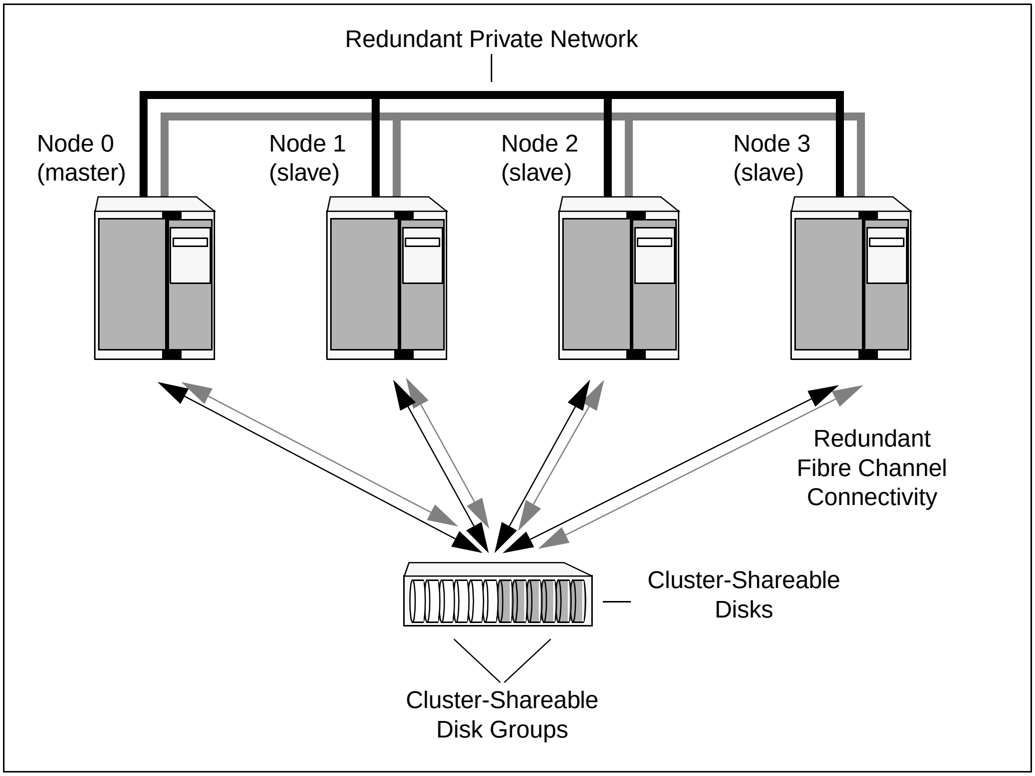
CVM allows up to 16 nodes in a cluster to simultaneously access and manage a set of disks under VxVM control (VM disks). The same logical view of the disk configuration and any changes are available on each node. When the cluster functionality is enabled, all cluster nodes can share VxVM objects. Features 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.

The figure “Example of a Four-Node Cluster” on page 65 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*.



Example of a Four-Node Cluster

To the cluster monitor, all nodes are the same. VxVM objects configured within shared disk groups can potentially be accessed by all nodes that join the cluster. However, the cluster functionality of VxVM requires one node to act as the *master node*; all other nodes in the cluster are *slave nodes*. Any node is capable of being the master node, which is responsible for coordinating certain VxVM activities.

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

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

Private and Shared Disk Groups

There are two types of disk groups:

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

In a cluster, most disk groups are shared. Disks in a shared disk group are accessible from all nodes in a cluster, allowing applications on multiple cluster nodes to simultaneously access the same disk. A volume in a shared disk group can be simultaneously accessed by more than one node in the cluster, subject to licensing and disk group activation mode restrictions.

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

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

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

Whether all members of the cluster have simultaneous read and write access to a cluster-shareable disk group depends on its *activation mode* setting as discussed in "Activation 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 VERITAS 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 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 To maintain compatibility with previous VxVM releases, the default activation mode for shared disk groups is `shared-write`.

Note The default activation mode for shared disk groups is `off` (inactive).

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 impossible to activate a disk group on a node if it is activated in a conflicting mode on another node in the cluster.

The following table summarizes the allowed and conflicting activation modes for shared disk groups:

Disk group activated in cluster as...	**Attempt to activate disk group on another node 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, the defaults file `/etc/default/vxdg` must be created.

The defaults file `/etc/default/vxdg` must contain the following lines:

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

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 variable *activation-mode* represents exclusive-write, read-only, shared-read, shared-write, or off. Note that when enabling activation using the defaults file, VERITAS recommends the file be identical on each node in the cluster. Otherwise, activation results 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 changes to take effect.

Caution If the default activation node is anything other than off, an activation following a cluster join, or 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` *disk_group* command. You can also use the `vxdg` 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. Either of the following policies for resolving this type of discrepancy may be applied:

- Under the *global* connectivity policy, the detach occurs cluster-wide (globally) if any node in the cluster reports a disk failure. This is the default policy.
- Under the *local* connectivity policy, 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 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.

SFCFS Administration 7

The VERITAS Cluster File System (CFS) 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, CFS requires the integrated set of VERITAS products included in the VERITAS Storage Foundation Cluster File System (SFCFS).

To configure a cluster, CFS requires the VERITAS Cluster Server (VCS). VCS supplies two major components integral to CFS. The Low Latency Transport (LLT) package provides node-to-node communications and monitors network communications. The Group Membership and Atomic Broadcast (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 (see “Packages Installed with SFCFS Only” on page 12) when installing SFCFS HA.

CFS 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/VRTSvmdoc` directory after you install the Storage Foundation packages. The user guides for VCS are available in the `/opt/VRTSvcsdc` directory after you install the Storage Foundation Cluster File System HA packages.

Topics in this chapter include:

VCS Overview

The VERITAS Cluster Server 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.

GAB and 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.

VCS Messaging—GAB

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 for additional information.

VCS Communication—LLT

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 for details on how you can modify the LLT configuration.

CVM 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. There must be at least one non-shared disk on each node for the creation of `rootdg` to initialize VxVM. See “CVM Administration” on page 97 for extensive background information on CVM.

CFS Overview

A file system cluster consists of one primary, and up to fifteen 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 (“Distributing the Load on a Cluster” on page 78).

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

Cluster/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.

CFS Primary and CFS Secondary

The primary file system handles the 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.

Use the following command to determine primaryship:

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

Use the following command to give primaryship to a node:

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

A cluster file system node can be mounted using the `mount seconly` option to prevent it from ever assuming the primaryship of the cluster. If a primary node fails and all the remaining nodes in the cluster file system are mounted as secondary-only, the cluster is disabled. The `mount seconly` option overrides any setting by the `fsclustadm` command. See the `mount_vxfs`(1M) man page for more information on the `seconly` option.

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.

		Secondary		
		ro	**rw**	**ro, crw**
Primary	**ro**	✓		
	rw		✓	✓
	ro, crw		✓	✓

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) man page for information on the cluster read-write (`crw`) mount option.

CFS and CVM Agents

Agents are VCS processes that manage predefined resource types. CFS 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 CFS and CVM agents are add-on resources to VCS specifically for the VERITAS File System and VERITAS Volume Manager (see "Agents for SFCFS / SFCFS HA" on page 105 for detailed information on CFS and CVM agents).

CFS Administration

This section describes some of the major aspects of cluster file system administration and the ways in which it differs from single-host VxFS administration.

CFS Resource Management Commands

To make resources easier to manage, five CFS administrative commands were introduced in this release. The commands are:

- `cfscluster`—cluster configuration command
- `cfsmntadm`—adds, deletes, modifies, and sets policy on cluster mounted file systems
- `cfsdgadm`—adds or deletes shared disk groups to / from a cluster configuration
- `cfsmount` / `cfsumount`—mounts / unmounts a cluster file system on a shared volume

CFS Commands

The `mount` and `fsclustadm` commands are also 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 for information on allowable mount options.

fsclustadm

The fsclustadm command reports various attributes of a cluster file system. Using fsclustadm you can show and set the primary node in a cluster, translate node IDs to host names and vice versa, list all nodes that currently have a cluster mount of the specified file system mount point, and determine whether a mount is a local or cluster mount. The fsclustadm command operates from any node in a cluster on which the file system is mounted, and can control the location of the primary for a specified mount point. See the fsclustadm(1M) manual page for information on usage.

fsadm

The fsadm command must be run from the primary node; it fails if invoked on secondaries. See the fsadm(1M) manual page for information on allowable mount options.

Running 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, CFS 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

CFS 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 Cluster File System

There is a master node for CVM as well as a primary for CFS. When growing a file system, you grow the volume from the CVM master, and then grow the file system from the CFS primary. The CVM master and the CFS primary can be two different nodes. To determine the primary file system in a cluster, enter:

```
# fsclustadm –v showprimary mount_point
```

To determine if the current node is the master CVM node, enter:

```
# vxdctl -c mode
```

To actually increase the size of the file system, run the following two commands. On the master CVM node, enter:

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

On the CFS primary, enter:

```
# fsadm –F 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.

Distributing the Load on a Cluster

Distributing the workload in a cluster provides performance and failover advantages. Because each cluster mounted file system can have a different node as its primary, CFS lets you easily distribute the load in a cluster.

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 CFS 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.

Using 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 CFS. The VCS GUI provides log files for debugging LLT and GAB events.

Snapshots on CFS

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

1. 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.

2. Multiple snapshots of a cluster file system can be mounted on the same or a different node in a cluster.

3. A snapshot is accessible only on the node mounting a snapshot. The snapshot device cannot be mounted on two different nodes simultaneously.

4. 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.

5. On the node mounting a snapshot, the snapped file system cannot be unmounted while the snapshot is mounted.

6. A CFS 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.

7. 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 Cluster File System

The following example shows how to create and mount a snapshot on a two-node cluster using CFS administrative interface commands.

▼ To create a snapshot on a cluster file system

1. Create a VxFS file system on a shared VxVM volume:

```
# mkfs –F vxfs /dev/vx/rdsk/cfsdg/vol1
version 4 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. Mount the file system on all nodes (following previous examples, on `system01` and `system02`):

```
# 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:

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

Note 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:

   ```
   # 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:

   ```
   # cfsumount /mnt1snap
   ```

Fencing Administration 8

I/O Fencing

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.

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. VERITAS 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 persistent reservations. Note that the use of the vxfentsthdw utility to test for SCSI-3 persistent reservation support requires that disks be 1MB or greater. Smaller disks can be tested manually. Contact VERITAS support (http://support.veritas.com) for the procedure.

Using the vxfentsthdw Utility

Use the vxfentsthdw utility to verify that the shared storage arrays support SCSI-3 persistent reservations and I/O fencing. VERITAS recommends testing several disks in each array. Only supported storage devices can be used with I/O fencing of shared storage.

The vxfentsthdw utility, which can be run from a single system in the cluster, tests storage by setting SCSI-3 registrations on the specified disk, verifying the registrations, and removing them from the disk.

Caution The vxfentsthdw utility overwrites and destroys existing data on the disks. Use the **-**r option to ensue non-destructive testing.

In the following example, assume you are checking the shared device that both systems recognize. (It is also possible that each system would use a different name for the same physical device.)

▼ To verify support for SCSI-3 persistent reservations

1. Start the utility on a single system:

```
  # cd /opt/VRTSvcs/vxfen/bin
  # ./vxfentsthdw
```

   The utility provides an overview of its function and behavior. It warns you that the tests it performs overwrites any data on the disks you check. Output resembles:

```
 ******** WARNING!!!!!!!! ******** THIS UTILITY WILL DESTROY THE
DATA ON THE DISK!!
Do you still want to contnue : [y/n] (default: n)
```

2. Enter **y** to continue.

3. When prompted, enter the name of the first cluster node. In this example, “star33.”

4. When prompted, enter the name of the second cluster node. In this example, “star34.”

5. You are then prompted to enter the first and second disk names (see above). Verify the same disk is recognized by star33 and star34. Note the disk names, whether or not they are identical, must refer to the same physical disk. If they do not, testing terminates. The utility begins its check and reports its activities.

6. Run the vxfentsthdw utility on each disk to be verified.

7. Continue to the next section to configure coordinator disks.

Configuring Coordinator Disks

1. Create a disk group `vxfencoorddg` on any cluster node.

2. Deport the disk group:

   ```
   # vxdg deport vxfencoorddg
   ```

3. Import the disk group with the `-t` option to avoid automatically importing it when the systems are rebooted:

   ```
   # vxdg -t import vxfencoorddg
   ```

4. Deport the disk group. Deporting the disk group prevents the coordinator disks from being used for other purposes.

   ```
   # vxdg deport vxfencoorddg
   ```

5. On each system, enter the command:

   ```
   # echo "vxfencoorddg" > /etc/vxfendg
   ```

   No spaces should appear between the quotes in the **`"vxfencoorddg"`** text.

   This command creates the file `/etc/vxfendg`, which includes the name of the coordinator disk group. Based on the `/etc/vxfendg` file, the `rc` script creates the file `/etc/vxfentab` for use by the `vxfen` driver. The `/etc/vxfentab` file is a generated file and should not be modified.

6. Issue the following command on each system to start I/O fencing:

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

   The `rc` startup script automatically creates `/etc/vxfentab` by reading the disks contained in the disk group listed in `/etc/vxfendg`. The script then invokes the `vxfenconfig` command, which configures the `vxfen` driver to start and use the coordinator disks listed in `/etc/vxfentab`. The same disks may be listed using different names on each system.

Adding or Removing Coordinator Disks

Before adding coordinator disks, verify the disks support SCSI-3 persistent reservations. See “Using the vxfentsthdw Utility” on page 85 for instructions.

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. Command options include:

`-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

Registration Key Formatting

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.

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

The keys currently assigned to disks can be displayed by using the command `vxfenadm -g /dev/`*`device_name`* command. For example, from the system with node ID 1, display the key for the *device_name* by entering:

```
# 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, then stop and restart the driver, as instructed below:

```
# echo "vxfen_mode=disabled" > /etc/vxfenmode
# /etc/rc2.d/S97vxfen stop
# /etc/rc2.d/S97vxfen start
```

Additionally, we recommend 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.

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.

Event	Node A: What Happens?	Node B: What Happens?	Action
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.

Event	Node A: What Happens?	Node B: What Happens?	Action
Nodes A and B and private networks lose power. Coordinator and data disks retain power. Power returns to nodes and they reboot, but private networks still have no power.	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: `Potentially a preexisting split brain. Dropping out of the cluster. Refer to the user documentation for steps required to clear preexisting split brain.`	Node B 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: `Potentially a preexisting split brain. Dropping out of the cluster. Refer to the user documentation for steps required to clear preexisting split brain.`	Refer to section in Troubleshooting chapter for instructions on resolving preexisting split brain condition.
Node A crashes while Node B is down. Node B comes up and Node A is still down.	Node A is crashed.	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: `Potentially a preexisting split brain. Dropping out of the cluster. Refer to the user documentation for steps required to clear preexisting split brain.`	Refer to section in Troubleshooting chapter for instructions on resolving preexisting split brain condition.

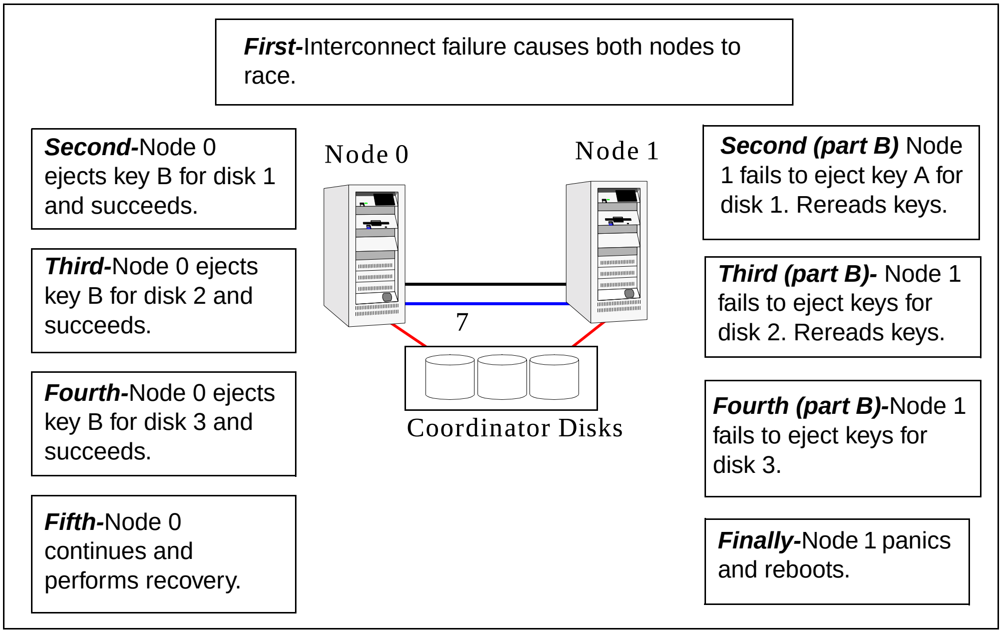
Event	Node A: What Happens?	Node B: What Happens?	Action
The disk array containing two of the three coordinator disks is powered off. Node B leaves the cluster and the disk array is still powered off.	Node A continues to operate as long as no nodes leave the cluster. Node A races for a majority of coordinator disks. Node A fails because only one of three coordinator disks is available. Node A removes itself from the cluster.	Node B continues to operate as long as no nodes leave the cluster. Node B leaves the cluster.	Power on failed disk array and restart I/O fencing driver to enable Node A to register with all coordinator disks.

Troubleshooting Fenced Configurations

The following information provides describes network partitioning in a fenced environment. For a more comprehensive explanation of this topic, see the chapter on VCS communications in the *VERITAS Cluster Server User's Guide*.

Example of a Preexisting Network Partition (Split-Brain)

The scenario illustrated below shows a two-node cluster in which the severed cluster interconnect poses a potential split-brain condition.



First-Interconnect failure causes both nodes to race.

Second-Node 0 ejects key B for disk 1 and succeeds.

Third-Node 0 ejects key B for disk 2 and succeeds.

Fourth-Node 0 ejects key B for disk 3 and succeeds.

Fifth-Node 0 continues and performs recovery.

Second (part B) Node 1 fails to eject key A for disk 1. Rereads keys.

Third (part B)- Node 1 fails to eject keys for disk 2. Rereads keys.

Fourth (part B)-Node 1 fails to eject keys for disk 3.

Finally-Node 1 panics and reboots.

Because the fencing module operates identically on each system, both nodes assume the other is failed, and carry out fencing operations to confirm. 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.

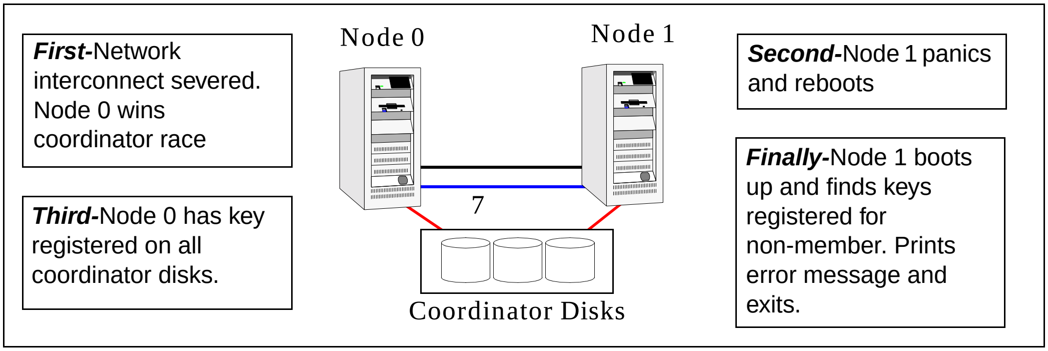
The side that successfully ejects the peer from a majority of the coordinator disks wins. The fencing module on the winning side then passes the membership change up to VCS and other higher-level packages registered with the fencing module, allowing VCS to invoke recovery actions. The losing side forces 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

Another scenario that could cause similar symptoms would be a two-node cluster with one node shut down for maintenance. During the outage, the private interconnect cables are disconnected.


First-Network interconnect severed. Node 0 wins coordinator race
Third-Node 0 has key registered on all coordinator disks.
Node 0
Node 1
7
Coordinator Disks
Second-Node 1 panics and reboots
Finally-Node 1 boots up and finds keys registered for non-member. Prints error message and exits.

In this scenario:

- ✔ 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 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 panics and reboots (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 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.

1. On Node 1, type:

   ```
   # /opt/VRTSvcs/vxfen/bin/vxfenclearpre
   ```

2. Restart the node.

CVM Administration 9

Introduction

Note See the *VERITAS Volume Manager Adminstrator's Guide* for complete information on VxVM and CVM. Online versions of the VxVM documentation set are installed under the `/opt/VRTSvmdoc` directory.

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 VERITAS Cluster Server™ (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 cluster functionality (CVM) of VERITAS Volume Manager (VxVM) allows up to 16 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.

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*.

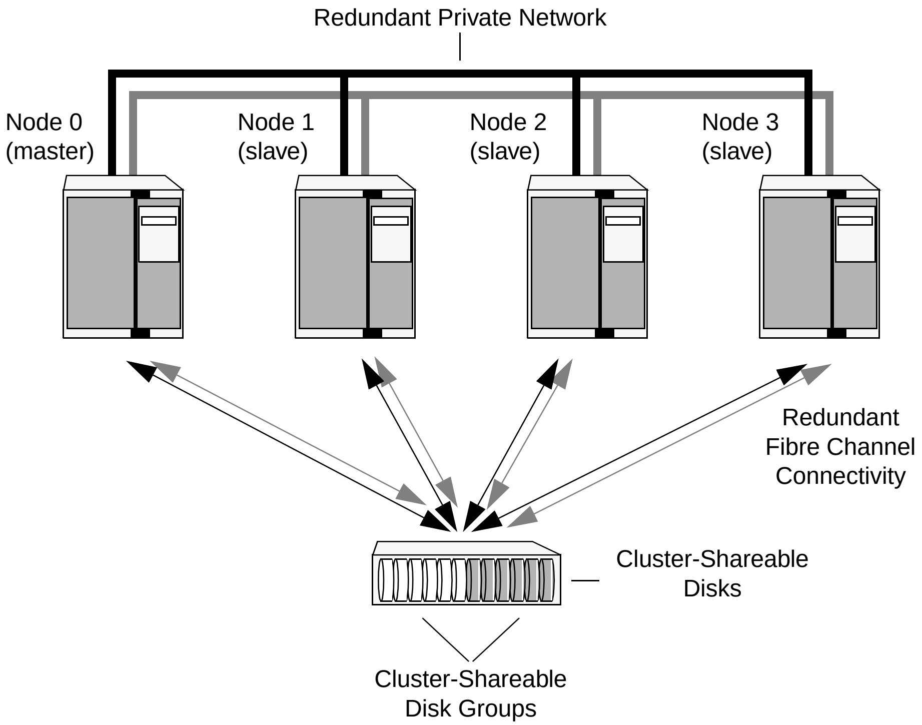
Another major advantage of clustered systems is their ability to reduce contention for system resources caused by activities such as backup, decision support and report generation. Enhanced value can be derived from cluster systems by performing such operations on lightly loaded nodes in the cluster 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.

The figure “Example of a 4-Node Cluster” on page 99 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*.

Example of a 4-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 the master node. If the master node leaves the cluster, one of the slave nodes is chosen to be the new master. In "Example of a 4-Node Cluster," node 0 is the master node and nodes 1, 2 and 3 are slave nodes.

Private and Shared Disk Groups

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

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

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

Whether all members of the cluster have simultaneous read and write access to a cluster-shareable disk group depends on its *activation mode* setting as discussed in "Activation 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).

Activation 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). Activation modes are described in the table "Allowed and Conflicting Activation Modes" on page 102.

Note The default activation mode for shared disk groups is `off` (inactive).

Applications such as high availability (HA) 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.

The table “Allowed and Conflicting Activation Modes” summarizes the allowed and conflicting activation modes for shared disk groups:

Allowed and Conflicting Activation Modes

Disk group activated in cluster as...	**Attempt to activate disk group on another node 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, the defaults file `/etc/default/vxdg` must be created.

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`.

Caution 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.

Caution 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.

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. Either of the following policies for resolving this type of discrepancy may be applied:

- Under the *global* connectivity policy, the detach occurs cluster-wide (globally) if any node in the cluster reports a disk failure. This is the default policy.
- Under the *local* connectivity policy, 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 `vxdg` command can be used to set the disk dettach 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. The following policies are supported:

- *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.

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 (`rootdg`) 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 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.

Agents for SFCFS/SFCFS HA 10

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. Typically 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 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 added when you run the `cfscluster config` command. For more information on VCS agents, see the *VERITAS Cluster Server Bundled Agents Reference Guide*. PDF versions of this guide are located under the `/opt/VRTSvcsdc` directory.

Topics in this appendix include:

List of CFS Agents

The CFS agents include:

- CFSMount Agent
- CFSfsckd Agent
- CVMCluster Agent
- CVMVolDg Agent

VCS Cluster Components

Resources, attributes, and service groups are components integral to cluster functionality (see the *VERITAS Cluster Server User's Guide* in the `/opt/VRTSvcsdc` directory for more information).

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* for instructions on how to install the GUI, and the *VERITAS Cluster Server User’s Guide* for detailed information on how to perform VCS administrative tasks.

Resources and Service Groups for File System Cluster Functionality

Managing cluster mounts through VCS requires various resources types, resources, and service groups. The 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 (see “Cluster File System Administrative Interface” on page 108) to add the CVMCluster resource. The procedure creates a service group named `cvm` and adds the resources to the configuration.

The VCS resource types required for cluster file system (CFS) functionality are:

- CFSfsckd
- CFSMount

CFSfsckd is a mandatory resource type for CFS 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 CFS configuration process (see “The cfscluster Command” on page 109), which adds the resources to the `cvm` service group.

Each CVMVolDg resource controls one shared disk group, and one or more shared volumes of that disk group. CVMVolDg resources enable the disk group and set the disk group activation mode. Each CFSMount resource controls the cluster mount of a shared volume on a specified mount point. CFSMount resources also keep track of mount options associated with the mount points.

These resource instances are not added automatically during the CFS configuration; you must add them as required using the CFS cluster administration commands. Note that CFSMount and CVMVolDg resources are not added to the `cvm` service group; those should be added to a different service group (see "The cfsmntadm Command" on page 109).

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 CFS must follow these dependency (or *link*) rules:

- A CFSMount resource must depend on the corresponding CVMVolDg resource
- A service group containing the CVMVolDg resource must depend on the `cvm` service group

Cluster File System Administrative Interface

The CFS administrative interface provides an easy and convenient way to create resources required for CFS with the correct attributes and the correct links between them.

CFS 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 CFS administrative commands are provided. It is advisable to use only these commands to manage cluster file systems. The commands are:

- `cfscluster`—cluster configuration command
- `cfsmntadm`—adds, deletes, modifies, and sets policy on cluster mounted file systems
- `cfsdgadm`—adds or deletes shared disk groups to/from a cluster configuration
- `cfsmount`/`cfsumount`—mounts/unmounts a cluster file system on a shared volume

The cfscluster Command

The `cfscluster` command is used primarily to configure and unconfigure CVM and CFS, and can be run from any node in the cluster. VCS must be started before you can run the `cfscluster config` command. The `cfscluster config` command adds all the resource type definitions (listed in "Resources and Service Groups for File System Cluster Functionality" on page 107) and adds resource instances, one each of type CVMCluster and CFSfsckd. The `cfscluster config` command also brings the resources online, and `cfscluster status` can be used to query the status of VCS.

The `cfscluster unconfig` command takes resources offline (except CFSMount resources) and removes all the resources and service groups that were used to manage the cluster file system (see the `cfscluster`(1M) manual page for more information). You must manually take CFSMount resources offline (using the `cfsumount` command) before executing the `cfscluster unconfig` command.

The cfsmntadm Command

One CVMVolDg and one CFSMount resource is required to control each cluster mount (see "Resources and Service Groups for File System Cluster Functionality" on page 107). 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.

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 for more information).

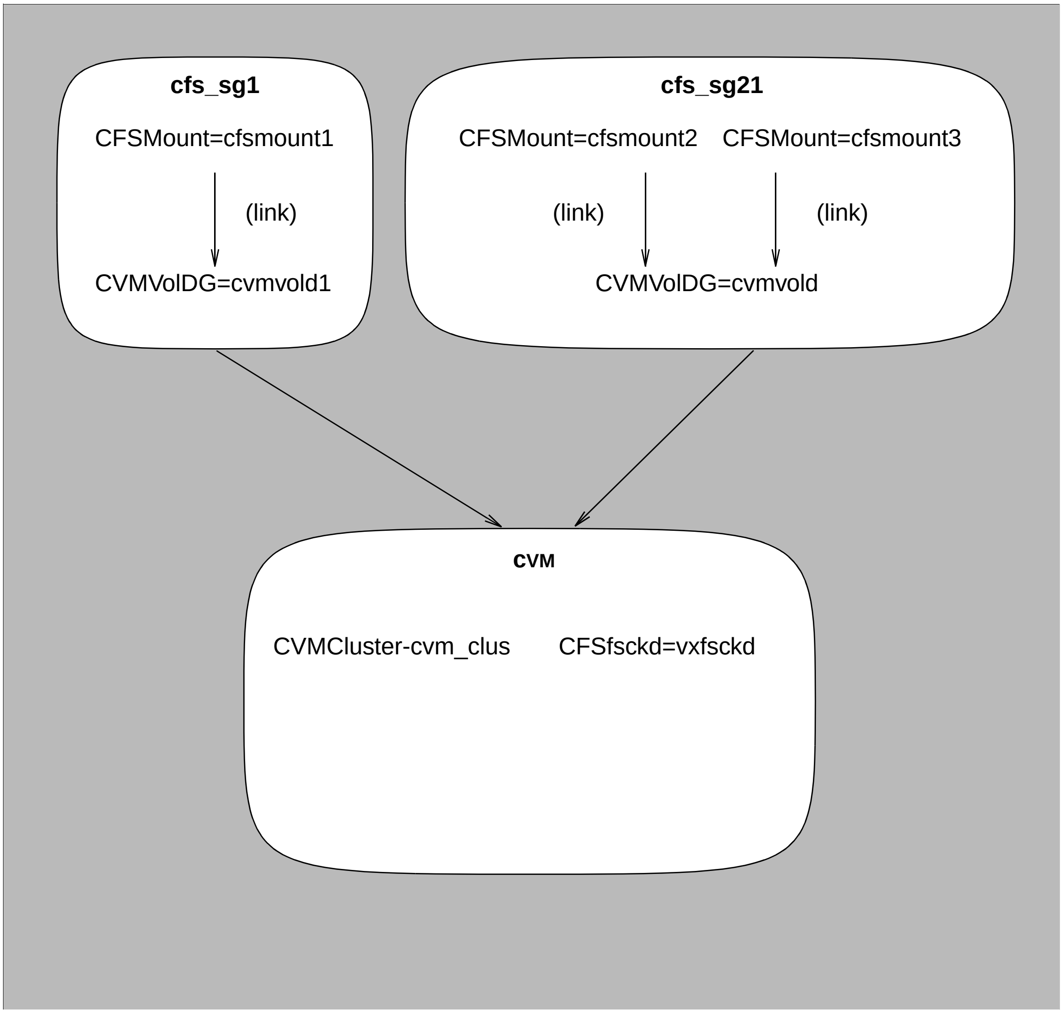
The cfsdgadm Command

The `cfsdgadm` command is the administrative interface for shared disk groups. Using `cfsdgadm`, you can add a shared disk group to a cluster configuration, delete a shared disk group, modify the activation mode, or display the shared disk group's configuration information. A shared disk group must already exist before being specified with `cfsdgadm` (see the `cfsdgadm`(1M) manual page for more information).

The cfsmount/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 for more information).

CFS Service Groups and Resource Dependencies



Example 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/vol1"
  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

```
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

```
type CFSMount (
  static keylist RegList = { MountOpt, Policy, NodeList, ForceOff,
SetPrimary }
  static int FaultOnMonitorTimeouts = 1
  static int OnlineRetryLimit = 16
  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

Description	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 file `/etc/VRTSvcs/conf/config/CFSTypes.cf`.	
Entry Points	- Online—Mounts a block device or file system snapshot in cluster mode. - Offline—Unmounts the file system (doing a forced unmount if necessary). - Monitor—Determines if the file system is mounted. Checks mount status using the `fsclustadm` command. - Clean—A null operation for a cluster file system mount. - attr_change—Remounts file system with new mount option; sets new primary for file system; sets `fsclustadm` policy on file system.	
Required Attributes	**Type and Dimension**	**Definition**
BlockDevice	string-scalar	Block device for mount point.
MountPoint	string-scalar	Directory for mount point.
NodeList	string-keylist	List of nodes on which to mount.
Optional Attributes	**Type and Dimension**	**Definition**
Policy	string-scalar	Node list for the primary file system selection policy.
MountOpt	string-scalar	Options for the `mount` command. To create a valid MountOpt attribute string: - Use VxFS type-specific options only. - Do not use the `-o` flag to specify the VxFS-specific options. - Do not use the `-F vxfs` file system type option. - The `cluster` option is not required. - Specify options in a comma-separated list as in these examples: ro ro,cluster blkclear,mincache=closesync
Internal Attributes		
MountType, Primary, SetPrimary, RemountRes, ForceOff		Not user-configured—used only by system.

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 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

Description	Starts, stops, and monitors the `vxfsckd` process. The CFSfsckd agent executable is `/opt/VRTSvcs/bin/CFSfsckd/CFSfsckdAgent`. The type definition is in the file `/etc/VRTSvcs/conf/config/CFSTypes.cf`. The configuration is added to the `main.cf` file after running the `cfscluster config` command.	
Entry Points	- Online—Starts the `vxfsckd` process. - Offline—Stops the `vxfsckd` process. - Monitor—Checks whether the `vxfsckd` process is running. - Clean—A null operation for a cluster file system mount.	
Required Attributes	**Type and Dimension**	**Definition**
None		
Optional Attributes	**Type and Dimension**	**Definition**
None		

Type Definition

```
type CFSfsckd (
  static int RestartLimit = 2
  static str LogLevel
  static str ArgList[] = { }
  NameRule = ““
)
```

Sample Configuration

```
CFSfsckd vxfsckd (
)
```

CVMCluster Agent

Description	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 file `/etc/VRTSvcs/conf/config/CVMTypes.cf`. The configuration is added to the `main.cf` file after running the `cfscluster config` command.	
Entry Points	- 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.	
Required Attributes	**Type and Dimension**	**Definition**
CVMClustName	string-scalar	Name of the cluster.
CVMNodeAddr	string-association	List of host names and IP addresses.
CVMNodeId	string-association	List of host names and LLT node numbers.
CVMTransport	string-association	The CVM transport mode, either `gab` or `udp`. For SFCFS, `gab` is the only valid transport mode.
PortConfigd	integer-scalar	Port number used by CVM for vxconfigd-level communication.
PortKmsgd	integer-scalar	Port number used by CVM for kernel-level communication.
CVMTimeout	integer-scalar	Timeout used by CVM during cluster reconfigurations.

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 Configuration

```
CVMCluster cvm_clus (
    Critical = 0
    CVMClustName = vcs
    CVMNodeId = { node1 = 1, node2 = 2 }
    CVMTransport = gab
    CVMTimeout = 200
  )
```

CVMVolDg Agent

Description	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 file `/etc/VRTSvcs/conf/config/CVMTypes.cf`.	
Entry Points	- 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.	
Required Attributes	**Type and Dimension**	**Definition**
CVMDiskGroup	string-scalar	Shared disk group name.
CVMVolume	string-keylist	Shared volume names.
CVMActivation	string-scalar	Activation mode for the disk group. Must be set to shared-write (`sw`). This is a localized attribute.

Type Definition

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

Sample Configuration

```
CVMVolDg testdg (
   CVMDiskGroup = testdg
   CVMActivation @node1 = sw
   CVMActivation @node2 = sw
   )
```

Troubleshooting and Recovery A

Installation Issues

If you encounter any issues installing SFCFS / SFCFS HA, refer to the following paragraphs for typical problems and their solutions.

Incorrect Permissions for Root on Remote System

The permissions are inappropriate. Make sure you have remote root access permission on each system to which you are installing.

```
  Checking communication with system01 ................. FAILED
Remote remsh/rcp permissions not available on: system01
Correct permissions and continue
Continue? [Y/N] :
```

Suggested solution: add a line in the file `/.rhosts` giving root access to the system running the install program. A “+” character in the first line of `/.rhosts` will permit any remote host to access the system.

Note Remove remote shell permissions after completing the SFCFS / SFCFS HA installation and configuration.

Resource Temporarily Unavailable

If the installation fails with the following error message on the console:

```
fork() failed: Resource temporarily unavailable
```

The value of HP-UX `nkthread` tunable parameter nay not be large enough. The `nkthread` tunable requires a minimum value of 600 on all systems in the cluster. To determine the current value of `nkthread`, enter:

```
# kctune –q nkthread
```

If necessary, you can change the value of `nkthread` using the SAM (System Administration Manager) interface, or by running the `kctune` command. If you change the value of `nkthread`, the kernel must be rebuilt for the new value to take effect. It is easier to change the value using SAM because there is an option to process the new kernel immediately. See the `kctune`(1M) and `sam`(1M) manual pages for more information on tuning kernel parameters.

Inaccessible System

- The system you specified is not accessible. This could be for a variety of reasons, such as, the system name was entered incorrectly or the system is not available over the network.

  ```
  Checking communication with system01 ................. FAILED
  System not accessible : system01
  ```

  Suggested solution: verify that you entered the system name correctly; use the `ping`(1M) command to verify the accessibility of the host.

- If a system cannot access the software source depot, either `swagentd` is not running on the target system or the `swlist` command cannot see the source depot.

  ```
  Correct /etc/{hosts, nsswitch.conf} and continue from here
  Continue? [Y/N] :
  ```

  Suggested solutions: check that `swagentd` is running. Check whether there is an entry for the target system in `/etc/hosts`. If there is no entry, then ensure the `hosts` file is not the primary lookup for the “hosts” entry.

CFS Problems

If there is a device failure or controller failure to a device, the file system may become disabled cluster-wide. To address the problem, unmount all secondary mounts, unmount the primary, then run a full `fsck`. When the file system check completes, mount all nodes again.

Unmount Failures

The `umount` command can fail for the following reasons:

- When unmounting shared file systems, you must unmount the secondaries before unmounting the primary.
- A reference is being held by an NFS server. Unshare the mount point and try the unmount again.

Mount Failures

Mounting a file system can fail for the following reasons:

- The file system is not using disk layout Version 4, 5 or 6.
- The mount options do not match the options of already mounted nodes.
- A cluster file system is mounted by default with the `qio` option enabled if the node has a Quick I/O for Databases license installed, even if the `qio` mount option was not explicitly specified. If the Quick I/O license is not installed, a cluster file system is mounted *without* the `qio` option enabled. So if some nodes in the cluster have a Quick I/O license installed and others do not, a cluster mount can succeed on some nodes and fail on others due to different mount options. To avoid this situation, ensure that Quick I/O licensing is uniformly applied, or be careful to mount the cluster file system with the `qio`/`noqio` option appropriately specified on each node of the cluster. See the `mount`(1M) manual page for more information on mount options.
- A shared CVM volume was not specified.
- The device is still mounted as a local file system somewhere on the cluster. Unmount the device.
- The `fsck` or `mkfs` command is being run on the same volume from another node, or the volume is mounted in non-cluster mode from another node.
- The `vxfsckd` daemon is not running. This typically happens only if the `CFSfsckd` agent was not started correctly.

- If mount fails with an error message:

  ```
  vxfs mount: cannot open mnttab
  ```

  /etc/mnttab is missing or you do not have root privileges.

- If mount fails with an error message:

  ```
  vxfs mount: device already mounted, ...
  ```

  The device is in use by mount, mkfs or fsck on the same node. This error cannot be generated from another node in the cluster.

- If this error message displays:

  ```
  mount: slow
  ```

  The node may be in the process of joining the cluster.

- If you try to mount a file system that is already mounted without –o cluster option (that is, not in shared mode) on another cluster node,

  ```
  # mount -F vxfs /dev/vx/dsk/share/vol01 /vol01
  ```

  the following error message displays:

  ```
  vxfs mount: /dev/vx/dsk/share/vol01 is already mounted, /vol01
  is busy, allowable number of mount points exceeded, or cluster
  reservation failed for the volume
  ```

- If umount fails with an error message:

  ```
  vxfs umount: /vol01 cannot unmount : Device busy
  ```

- You must unmount the file system on all secondary systems before unmounting it from the primary.

Command Failures

- Manual pages not accessible with the man command. Set the MANPATH environment variable as listed under "Setting PATH and MANPATH Environment Variables" on page 15.
- The mount, fsck, and mkfs utilities reserve a shared volume. They fail on volumes that are in use. Be careful when accessing shared volumes with other utilities such as dd, it is possible for these commands to destroy data on the disk.
- Running some commands, such as fsadm -E /vol02, can generate the following error message:

  ```
  vxfs fsadm: ERROR: not primary in a cluster file system
  ```

  This means that you can run this command only on the primary, that is, the system that mounted this file system first.

Performance Issues

Quick I/O File system performance is adversely affected if a cluster file system is mounted with the qio option enabled and Quick I/O is licensed, but the file system is *not* used for Quick I/O files. Because qio is enabled by default, if you do not intend to use a shared file system for Quick I/O, explicitly specify the noqio option when mounting.

High Availability Issues

Network Partition/Jeopardy

Network partition (or *split brain*) is a condition where a network failure can be misinterpreted as a failure of one or more nodes in a cluster. If one system in the cluster incorrectly assumes that another system failed, it may restart applications already running on the other system, thereby corrupting data. CFS tries to prevent this by having redundant heartbeat links.

At least one link must be active to maintain the integrity of the cluster. If all the links go down, after the last network link is broken, the node can no longer communicate with other nodes in the cluster. Thus the cluster is in one of two possible states. Either the last network link is broken (called a network partition condition), or the last network link is okay, but the node crashed, in which case it is not a network partition problem. It is not possible to identify whether it is the first or second state, so a kernel message is issued to indicate that a network partition may exist and there is a possibility of data corruption.

Jeopardy is a condition where a node in the cluster has a problem connecting to other nodes. In this situation, the link or disk heartbeat may be down, so a *jeopardy* warning may be displayed. Specifically, this message appears when a node has only one remaining link to the cluster and that link is a network link. This is considered a critical event because the node may lose its only remaining connection to the network.

Caution Do not remove the communication links while shared storage is still connected.

Low Memory

Under heavy loads, software that manages heartbeat communication links may not be able to allocate kernel memory. If this occurs, a node halts to avoid any chance of network partitioning. Reduce the load on the node if this happens frequently.

A similar situation may occur if the values in the `/etc/llttab` files on all cluster nodes are not correct or identical.