sgi®

CXFS™ Administration Guide for SGI® InfiniteStorage

007–4016–027

New Features in This Guide

Note: Be sure to read the release notes for your platforms and the late-breaking caveats page on Supportfolio to learn about any changes to the installation and configuration procedures.

This version contains the following:

- Support for clusters with server-capable administration nodes running only SGI ProPack for Linux. IRIX is now exclusively supported as a client-only platform and is generally documented in *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

  Because IRIX is no longer supported as a server-capable administration node, CXFS 5.0 also no longer supports coexecution with FailSafe. Because the client administration node type (`client_admin`) was supported only for coexecution with FailSafe, this node type is no longer supported.

  For information about moving from a cluster with IRIX metadata servers to a cluster with SGI ProPack metadata servers, see Appendix I, "Migration from a Cluster with IRIX Server-Capable Administration Nodes" on page 597.

  Note: CXFS does not support `SGIRDAC` mode. See "Changing `SGIRDAC` Mode to `SGIAVT` Mode for SGI RAID" on page 598.

- New initial installation and update procedures: "Installation from the ISSP Distribution" on page 112.
- The ability to revoke and restore the CXFS kernel membership for the local node, which must be a server-capable administration node, by using the `stop_cxfs` and `start_cxfs` commands in `cxfs_admin`. See "Revoke and Restore CXFS Kernel Membership for the Local Node" on page 248.
- The ability to generate streaming workload for SD/HD/2K/4K formats of video streams by using the `frametest`(1) command. See "Generation of Streaming Workload for Video Streams" on page 323.
- Support for the `framesort` utility, which provides easy file-layout analysis and advanced file-sequence reorganization. See "Frame Files Defragmentation and Analysis" on page 324.

- The new site-changeable static system-tunable parameter "`mtcp_hb_local_options`" on page 481
- Reorganization to improve clarity.
- Information about the `cmgr` command is located primarily in Appendix G, "Reference to `cmgr` Tasks" on page 493 (which has not been updated to reflect CXFS 5.0 support). With the exception of a few administrative `cmgr`commands, the preferred CXFS configuration tools are `cxfs_admin` and the CXFS graphical user interface (GUI). The following `cmgr` commands are still useful:

  ```
  admin fence
  admin nmi
  admin ping
  admin powerCycle
  admin reset
  start/stop cx_services
  test connectivity
  test serial
  ```

  In addition, the `build_cmgr_script` command is still useful.

  In a future release, all of `cmgr` functionality will be provided by the `cxfs_admin` command and the `cmgr` command will not be supported.
- The following commands are now deprecated: `clconf_status` and `cluster_status`. The functionality of these commands is now provided by the `cxfs_admin` command and/or the CXFS GUI. See Appendix J, "Deprecated Commands" on page 613.

Record of Revision

Version	Description
001	September 1999 Supports the CXFS 1.1 product in the IRIX 6.5.6f release.
002	October 1999 Supports the CXFS 1.1 product in the IRIX 6.5.6f release.
003	December 1999 Supports the CXFS product in the IRIX 6.5.7f release.
004	March 2000 Supports the CXFS product in the IRIX 6.5.8f release.
005	June 2000 Supports the CXFS product in the IRIX 6.5.9f release.
006	September 2000 Supports the CXFS product in the IRIX 6.5.10f release.
007	January 2001 Supports the CXFS product in the IRIX 6.5.11f release.
008	March 2001 Supports the CXFS product in the IRIX 6.5.12f release.
009	June 2001 Supports the CXFS product in the IRIX 6.5.13f release.
011	September 2001 Supports the CXFS product in the IRIX 6.5.14f release. (Note, there was no 010 version due to an internal numbering mechanism.)
012	December 2001 Supports the CXFS Version 2 product in IRIX 6.5.15f.
013	March 2002 Supports the CXFS Version 2 product in IRIX 6.5.16f.

014	June 2002 Supports the CXFS Version 2 product in IRIX 6.5.17f.
015	September 2002 Supports the CXFS Version 2 product in IRIX 6.5.18f.
016	December 2002 Supports the CXFS Version 2 product in IRIX 6.5.19f.
017	March 2003 Supports the CXFS Version 2 product in IRIX 6.5.20f.
018	September 2003 Supports the CXFS 3.0 product in IRIX 6.5.22 and CXFS 3.0 for SGI Altix 3000 running SGI ProPack 2.3 for Linux.
019	December 2003 Supports the CXFS 3.1 product in IRIX 6.5.23 and CXFS 3.1 for SGI Altix 3000 running SGI ProPack 2.4 for Linux.
020	March 2004 Supports the CXFS 3.2 product in IRIX 6.5.24 and CXFS 3.2 for SGI Altix 3000 running SGI ProPack 3 for Linux.
021	November 2004 Supports the CXFS 3.2 product in IRIX 6.5.24 and CXFS 3.2 for SGI Altix 3000 running SGI ProPack 3 for Linux.
022	April 2005 Supports the CXFS 3.3 product
023	July 2005 Supports the CXFS 3.4 product
024	May 2006 Supports the CXFS 4.0 product
025	January 2007 Supports the CXFS 4.1 product

026	September 2007 Supports the CXFS 4.2 product
027	March 2008 Supports the CXFS 5.0 product

Contents

Figures

Tables

About This Guide

This guide documents CXFS 5.0 running on a storage area network (SAN). It assumes that you are already familiar with the XFS filesystem and you have access to the *XVM Volume Manager Administrator's Guide*.

You should read through this entire book, especially Chapter 15, "Troubleshooting" on page 353, before attempting to install and configure a CXFS cluster.

Related Publications

The following documents contain additional information:

- *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*
- *XVM Volume Manager Administrator's Guide*
- *SGI InfiniteStorage High Availability Using Linux-HA Heartbeat*
- Storage area network (SAN) documentation:
  - *EL Serial Port Server Installation Guide* (provided by Digi International)
  - *EL Serial Port Server Installation Guide Errata*
  - *FDDIXPress Administration Guide*
  - *SGI InfiniteStorage RM610 and RM660 User's Guide*
  - *SGI® InfiniteStorage TP9400 and SGI® InfiniteStorage TP9500 and TP9500S RAID User's Guide*
  - *SGI InfiniteStorage TP9300 and TP9300S RAID User's Guide*
  - *SGI InfiniteStorage 6700 User's Guide*
  - *SGI Total Performance 9100 Storage System Owner's Guide*
  - *SGI TPSSM Administration Guide*

- SGI ProPack for Linux, SGI Altix, and SGI Altix XE documentation:
  - The user guide and quick start guide for your SGI Altix or SGI Altix XE system
  - *Guide to Programming Environments and Tools Available on SGI Altix XE System*
  - *NIS Administrator's Guide*
  - *Personal System Administration Guide*
  - *Performance Co-Pilot for Linux User's and Administrator's Guide*
  - *SGI ProPack 5 for Linux Service Pack 5 Start Here*
  - *SGI L1 and L2 Controller Software User's Guide*

The following man pages are provided on CXFS server-capable administration nodes:

Server-Capable Administration Node Man Page	Software Product
`cbeutil`(1M)	`cluster_admin`
`cdbBackup`(1M)	`cluster_admin`
`cdbRestore`(1M)	`cluster_admin`
`cdbconfig`(1M)	`cluster_admin`
`cdbutil`(1M)	`cluster_admin`
`cmond`(1M)	`cluster_admin`
`fs2d`(1M)	`cluster_admin`
`cluster_services.man.man`	
`cms_failconf`(1M)	`cluster_services`
`cms_intervene`(1M)	`cluster_control`
`crsd`(1M)	`cluster_services`
`haStatus`(1M)	`cluster_services`

Server-Capable Administration Node Man Page	Software Product
`cxfs_admin`(1M)	`cxfs_admin`
`cxfs-config`(1M)	`cxfs_util`
`cxfscp`(1)	`cxfs_util`
`cxfsdump`(1M)	`cxfs_util`
`cxfslicense`(1M)	`cxfs_util`
`xvm`(1M)	N/A
`xvm`(7M)	N/A
`xvm`(5)	`cxfs-xvm-cmds`
`xvm`(8)	`cxfs-xvm-cmds`
`cxfsmgr`(1M)	`cxfs-sysadm_cxfs-client`
`xvmgr`(1M)	`cxfs-sysadm_xvm-client`

Obtaining Publications

You can obtain SGI documentation as follows:

- See the SGI Technical Publications Library at http://docs.sgi.com. Various formats are available. This library contains the most recent and most comprehensive set of online books, release notes, man pages, and other information.
- You can view most of the release notes in the `/docs` directory.
- You can view man pages by typing `man` *title* at a command line.

Conventions

This guide uses the following terminology abbreviations:

- *Windows* refers to Microsoft Windows 2003, Microsoft Windows XP, and Microsoft Windows Vista
- *SGI ProPack* refers to SGI ProPack 5 for Linux running the `default` kernel on SGI Altix systems and the `smp` kernel of SGI Altix XE systems, as specified in the release notes

The following conventions are used throughout this document:

Convention	**Meaning**
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
variable	Italic typeface denotes variable entries and words or concepts being defined.
`user input`	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. (Output is shown in nonbold, fixed-space font.)
[ ]	Brackets enclose optional portions of a command or directive line.
GUI element	This bold font denotes the names of graphical user interface (GUI) elements, such as windows, screens, dialog boxes, menus, toolbars, icons, buttons, boxes, and fields.
`<TAB>`	Represents pressing the specified key in an interactive session
`admin#`	In an example, this prompt indicates that the command is executed on a server-capable administration node
`client#`	In an example, this prompt indicates that the command is executed on a client-only node
`MDS#`	In an example, this prompt indicates that the command is executed on an active metadata server

`node#`	In an example, this prompt indicates that the command is executed on an any node

This guide uses *Windows* to refer to both Microsoft Windows 2000 and Microsoft Windows XP nodes when the information applies equally to both. Information that applies to only one of these types of nodes is identified.

Reader Comments

If you have comments about the technical accuracy, content, or organization of this publication, contact SGI. Be sure to include the title and document number of the publication with your comments. (Online, the document number is located in the front matter of the publication. In printed publications, the document number is located at the bottom of each page.)

You can contact SGI in any of the following ways:

- Send e-mail to the following address:

  techpubs@sgi.com

- Contact your customer service representative and ask that an incident be filed in the SGI incident tracking system.

- Send mail to the following address:

  SGI
  Technical Publications
  1140 East Arques Avenue
  Sunnyvale, CA 94085–4602

SGI values your comments and will respond to them promptly.

Chapter 1

Introduction to CXFS

This chapter discusses the following:

- "What is CXFS?" on page 1
- "Comparison of XFS and CXFS" on page 3
- "Comparison of Network and CXFS Filesystems" on page 6
- "Cluster Environment Concepts" on page 9
- "CXFS Interoperability With Other Products and Features" on page 28
- "Hardware and Software Requirements for Server-Capable Administration Nodes" on page 34
- "CXFS Software Products Installed on Server-Capable Administration Nodes" on page 35
- "CXFS Tools" on page 36

What is CXFS?

CXFS is clustered XFS, a parallel-access shared clustered filesystem for high-performance computing environments. CXFS allows groups of computers to coherently share XFS filesystems among multiple hosts and storage devices while maintaining high performance. CXFS runs on storage area network (SAN) RAID storage devices, such as Fibre Channel. A *SAN* is a high-speed, scalable network of servers and storage devices that provides storage resource consolidation, enhanced data access, and centralized storage management.

CXFS filesystems are mounted across the cluster by CXFS management software. All files in the filesystem are available to all hosts (called *nodes*) that mount the filesystem. All shared filesystems must be built on top of cluster-owned XVM volumes.

CXFS provides a single-system view of the filesystems; each host in the SAN has equally direct access to the shared disks and common pathnames to the files. CXFS lets you scale the shared-filesystem performance as needed by adding disk channels and storage to increase the direct host-to-disk bandwidth. The CXFS shared-file performance is not limited by LAN speeds or a bottleneck of data passing through a centralized fileserver. It combines the speed of near-local disk access with the flexibility, scalability, and reliability of clustering.

To provide stability and performance, CXFS uses a private network and separate paths for data and *metadata* (information that describes a file, such as the file's name, size, location, and permissions). Each request is handled in a separate thread of execution, without limit, making CXFS execution highly parallel. See "Private Network" on page 22

CXFS provides centralized administration with an intuitive graphical user interface (GUI) and command-line interface. See "CXFS Tools" on page 36.

CXFS provides full cache coherency across heterogeneous nodes running multiple operating systems. CXFS supports:

- Server-capable administration nodes running SGI ProPack for Linux on either all SGI Altix ia64 systems or all SGI Altix XE x86_64 systems (do not mix architectures of server-capable administration nodes, see "Use the Same Architecture for All Server-Capable Administration Nodes" on page 45)
- Client-only nodes running any mixture of the following operating systems:
  - IBM AIX
  - SGI IRIX
  - Apple Mac OS X
  - SGI ProPack for Linux
  - SUSE Linux Enterprise Server (SLES)
  - Red Hat Enterprise Linux (RHEL)
  - Sun Solaris
  - Microsoft Windows

For details about client-only nodes, see the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage* and the release notes.

CXFS provides the following high-availability (HA) features:

- Replicated configuration database
- XVM path failover (failover version 2)

- Server failover
- Network failover

You can use the Linux-HA Heartbeat product in conjunction with CXFS to failover associated applications. See "Highly Available Services" on page 29.

Comparison of XFS and CXFS

CXFS uses the same filesystem structure as XFS. A CXFS filesystem is initially created using the same `mkfs` command used to create standard XFS filesystems.

This section discusses the following:

- "Differences in Filesystem Mounting and Management" on page 3
- "Supported XFS Features" on page 4
- "When to Use CXFS" on page 5
- "Performance Considerations" on page 5

Differences in Filesystem Mounting and Management

The primary difference between XFS and CXFS filesystems is the way in which filesystems are mounted and managed:

- In XFS:
  - XFS filesystems are mounted with the `mount` command directly by the system during boot via an entry in the `/etc/fstab` file.
  - An XFS filesystem resides on only one host.
  - The `/etc/fstab` file contains static information about XFS filesystems. For more information, see the `fstab`(5) man page.
- In CXFS:
  - CXFS filesystems are mounted using the CXFS graphical user interface (GUI) or the `cxfs_admin` command. See "CXFS Tools" on page 36.
  - A CXFS filesystem is accessible to those nodes in the cluster that are defined to mount it. CXFS filesystems are mounted across the cluster by CXFS

management software. All files in the CXFS filesystem are visible to those nodes that are defined to mount the filesystem.

- One node coordinates the updating of metadata on behalf of all nodes in a cluster; this is known as the *metadata server*.
- The filesystem information is stored in the *cluster database* (CDB), which contains persistent static configuration information about the filesystems, nodes, and cluster. The CXFS cluster administration daemons manage the distribution of multiple synchronized copies of the cluster database across the nodes that are capable of becoming metadata servers (the *server-capable administration nodes*). The administrator can view the database and modify it using the CXFS administration tools (see "CXFS Tools" on page 36).
- Information is **not** stored in the `/etc/fstab` file. (However, the CXFS filesystems do show up in the `/etc/mtab` file.) For CXFS, information is instead stored in the cluster database.

Supported XFS Features

XFS features that are also present in CXFS include the following:

- Reliability and fast (subsecond) recovery of a log-based filesystem.
- 64-bit scalability to 9 million terabytes (9 exabytes) per file.
- Speed: high *bandwidth* (megabytes per second), high *transaction rates* (I/O per second), and fast metadata operations.
- Dynamically allocated metadata space.
- Quotas. You can administer quotas from any IRIX, SGI ProPack, or Linux node in the cluster with the suitable software installed, regardless of its role in the cluster, just as if this were a regular XFS filesystem.
- Filesystem reorganizer (defragmenter), which must be run from the CXFS metadata server for a given filesystem. See the `fsr_xfs`(1M) man page.
- Restriction of access to files using file permissions and access control lists (ACLs). You can also use logical unit (LUN) masking or physical cabling to deny access from a specific node to a specific set of LUNs in the SAN.

CXFS preserves these underlying XFS features while distributing the I/O directly between the LUNs and the nodes. The efficient XFS I/O path uses asynchronous buffering techniques to avoid unnecessary physical I/O by delaying writes as long as possible. This allows the filesystem to allocate the data space efficiently and often contiguously. The data tends to be allocated in large contiguous chunks, which yields sustained high bandwidths.

The XFS directory structure is based on B-trees, which allow XFS to maintain good response times even as the number of files in a directory grows to tens or hundreds of thousands of files.

When to Use CXFS

You should use CXFS when you have multiple nodes running applications that require high-bandwidth access to common filesystems. CXFS performs best under the following conditions:

- Data I/O operations are greater than 16 KB
- Large files are being used (a lot of activity on small files will result in slower performance)
- Read/write conditions are one of the following:
  - All processes that perform reads/writes for a given file reside on the same node
  - The same file is read by processes on multiple nodes using buffered I/O, but there are no processes writing to the file
  - The same file is read and written by processes on more than one node using direct-access I/O

For most filesystem loads, the scenarios above represent the bulk of the file accesses. Thus, CXFS delivers fast local file performance. CXFS is also useful when the amount of data I/O is larger than the amount of metadata I/O. CXFS is faster than NFS because the data does not go through the network.

Performance Considerations

CXFS may not give optimal performance under the following circumstances, and you should give extra consideration to using CXFS in these cases:

- When you want to access files only on the local host.
- When distributed applications write to shared files that are memory mapped.

- When exporting a CXFS filesystem via NFS. Because performance will be much better when the export is performed from an active CXFS metadata server than when it is performed from a CXFS client, exporting from a potential metadata server or client is not supported. For more information, see "Node Types, Node Functions, and the Cluster Database" on page 12.
- When access would be as slow with CXFS as with network filesystems, such as with the following:
  - Small files
  - Low bandwidth
  - Lots of metadata transfer

  Metadata operations can take longer to complete through CXFS than on local filesystems. Metadata transaction examples include the following:
  - Opening and closing a file
  - Changing file size (usually extending a file)
  - Creating and deleting files
  - Searching a directory

  In addition, multiple processes on multiple hosts that are reading and writing the same file using buffered I/O can be slower with CXFS than when using a local filesystem. This performance difference comes from maintaining coherency among the distributed file buffers; a write into a shared, buffered file will invalidate data (pertaining to that file) that is buffered in other hosts.

Comparison of Network and CXFS Filesystems

Network filesystems and CXFS filesystems perform many of the same functions, but with important performance and functional differences noted here:

- "Network Filesystems" on page 7
- "CXFS Filesystems" on page 7

Network Filesystems

Accessing remote files over local area networks (LANs) can be significantly slower than accessing local files. The network hardware and software introduces delays that tend to significantly lower the transaction rates and the bandwidth. These delays are difficult to avoid in the client-server architecture of LAN-based network filesystems. The delays stem from the limits of the LAN bandwidth, latency, and shared path through the data server.

LAN bandwidths force an upper limit for the speed of most existing shared filesystems. This can be one to several orders of magnitude slower than the bandwidth possible across multiple disk channels to local or shared disks. The layers of network protocols and server software also tend to limit the bandwidth rates.

A shared fileserver can be a bottleneck for performance when multiple clients wait their turns for data, which must pass through the centralized fileserver. For example, NFS and Samba servers read data from disks attached to the server, copy the data into UDP/IP or TCP/IP packets, and then send it over a LAN to a client host. When many clients access the server simultaneously, the server's responsiveness degrades.

CXFS Filesystems

A CXFS filesystem is a clustered XFS filesystem that allows for logical file sharing similar to network filesystems, but with significant performance and functionality advantages. CXFS runs on top of a SAN, where each node in the cluster has direct high-speed data channels to a shared set of disks.

CXFS Features

CXFS has the following unique features:

- A *peer-to-disk* model for the data access. The shared files are treated as local files by all of the nodes in the cluster. Each node can read and write the disks at near-local disk speeds; the data passes directly from the disks to the node requesting the I/O, without passing through a data server or over a LAN. For the data path, each node is a peer on the SAN; each can have equally fast direct data paths to the shared disks. Therefore, adding disk channels and storage to the SAN can scale the bandwidth. On large systems, the bandwidth can scale to gigabytes and even tens of gigabytes per second. Compare this with a network filesystem where the data is typically flowing over a 1- to 100-MB-per-second LAN.

This peer-to-disk data path also removes the fileserver data-path bottleneck found in most LAN-based shared filesystems.

- Each node can buffer the shared disk much as it would for locally attached disks. CXFS maintains the coherency of these distributed buffers, preserving the advanced buffering techniques of the XFS filesystem.
- A flat, single-system view of the filesystem; it is identical from all nodes sharing the filesystem and is not dependent on any particular node. The pathname is a normal POSIX pathname; for example, `/data/`*username*`/`*directory*.

  Note: A Windows CXFS client uses the same pathname to the filesystem as other clients beneath a preconfigured drive letter.

  The path does not vary if the metadata server moves from one node to another or if a metadata server is added or replaced. This simplifies storage management for administrators and users. Multiple processes on one node and processes distributed across multiple nodes have the same view of the filesystem, with performance similar on each node.

  This differs from typical network filesystems, which tend to include the name of the fileserver in the pathname. This difference reflects the simplicity of the SAN architecture with its *direct-to-disk* I/O compared with the extra hierarchy of the LAN filesystem that goes through a named server to reach the disks.
- A full UNIX filesystem interface, including POSIX, System V, and BSD interfaces. This includes filesystem semantics such as mandatory and advisory record locks. No special record-locking library is required.

CXFS Restrictions

CXFS has the following restrictions:

- Some filesystem semantics are not appropriate and not supported in shared filesystems. For example, the root filesystem is not an appropriate shared filesystem. Root filesystems belong to a particular node, with system files configured for each particular node's characteristics.
- All processes using a named pipe must be on the same node.
- Hierarchical storage management (HSM) applications such as DMF must run on the active metadata server.

The following features are not supported in CXFS:

- The original XFS guaranteed-rate I/O (GRIO) implementation, GRIO version 1. GRIO version 2 is supported, see "Guaranteed-Rate I/O (GRIO) Version 2 Overview" on page 30.
- Swap to a file residing on a CXFS filesystem.
- XVM failover version 1
- External log filesystems
- Real-time filesystems

Cluster Environment Concepts

This section defines the concepts necessary to understand CXFS:

- "Metadata" on page 10
- "Node" on page 10
- "RAID" on page 10
- "LUN" on page 10
- "Cluster" on page 10
- "Pool" on page 11
- "Node Types, Node Functions, and the Cluster Database" on page 12
- "Membership" on page 22
- "Private Network" on page 22
- "Data Integrity Protection" on page 24
- "CXFS Tiebreaker" on page 24
- "Relocation" on page 25
- "Recovery" on page 25
- "CXFS Services" on page 28

Also see the Glossary on page 639.

Metadata

Metadata is information that describes a file, such as the file's name, size, location, and permissions. Metadata tends to be small, usually about 512 bytes per file in XFS. This differs from the *data*, which is the contents of the file. The data may be many megabytes or gigabytes in size.

Node

A *node* is an operating system (OS) image, usually an individual computer. (This use of the term *node* does not have the same meaning as a node in an SGI Origin 3000 or SGI 2000 system.)

RAID

A redundant array of independent disks.

LUN

A logical unit number (LUN) is a representation of disk space. In a RAID, the disks are not individually visible because they are behind the RAID controller. The RAID controller will divide up the total disk space into multiple LUNs. The operating system sees a LUN as a hard disk. A LUN is what XVM uses as its physical volume (*physvol*). For more information, see the *XVM Volume Manager Administrator's Guide*.

Cluster

A *cluster* is the set of nodes configured to work together as a single computing resource. A cluster is identified by a simple name and a cluster identification (ID) number. A cluster running multiple operating systems is known as a *multiOS cluster*. A given node can be a member of only one cluster.

LUNs are assigned to a cluster by recording the name of the cluster on the LUN. Thus, if any LUN is accessible (via a Fibre Channel connection) from nodes in different clusters, then those clusters must have unique names. When members of a cluster send messages to each other, they identify their cluster via the cluster ID. Cluster names and IDs must be unique.

Because of the above restrictions on cluster names and IDs, and because cluster names and IDs cannot be changed after the cluster is created (without deleting the cluster and recreating it), SGI advises that you choose unique names and IDs for each of the clusters within your organization.

Pool

The *pool* is the set of nodes from which a particular cluster may be formed. (All nodes created when using the `cxfs_admin` tool are automatically part of the cluster, so the concept of the pool is obsolete when using `cxfs_admin`.)

Only one cluster may be configured from a given pool, and it need not contain all of the available nodes. (Other pools may exist, but each is disjoint from the other. They share no node or cluster definitions.)

A pool is first formed when you connect to a given server-capable administration node (see "Node Types, Node Functions, and the Cluster Database" on page 12) and define that node in the cluster database using the GUI. You can then add other nodes to the pool by defining them while still connected to the first node. (If you were to connect to a different server-capable administration node and then define it, you would be creating a second pool).

Figure 1-1 shows the concepts of pool and cluster. Node9 and Node10 have been defined as nodes in the CXFS pool, but they have not been added to the cluster definition.

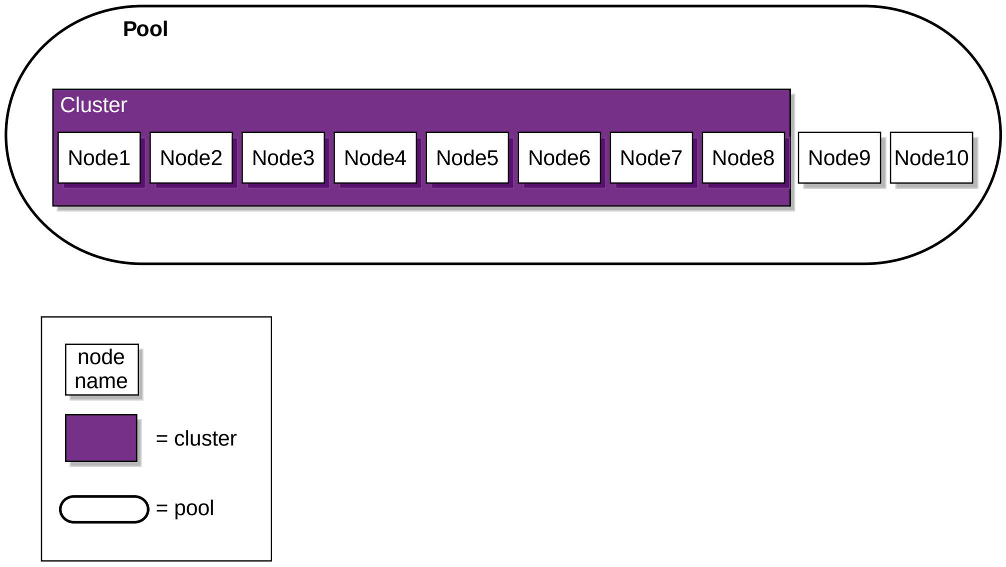


Figure 1-1 Pool and Cluster Concepts

Node Types, Node Functions, and the Cluster Database

The *cluster database* (CDB) contains configuration and logging information. A node is defined in the cluster database as one of the following types:

- *Server-capable administration node*, which is installed with the `cluster_admin` software product and contains the full set of cluster administration daemons (`fs2d`, `crsd`, `cad`, and `cmond`) and the CXFS server-capable administration node control daemon (`clconfd`). Only nodes running SGI ProPack for Linux can be server-capable administration nodes. This node type is capable of coordinating cluster activity and metadata. Multiple synchronized copies of the database are maintained across the server-capable administration nodes in the pool by the *cluster administration daemons*.

  Note: For any given configuration change, the CXFS cluster administration daemons must apply the associated changes to the cluster database and distribute the changes to each server-capable administration node before another change can take place.

For more details, see:

- "Cluster Administration Daemons" on page 40
- "CXFS Control Daemons" on page 41
- Appendix A, "CXFS Software Architecture" on page 419

- *Client-only node*, which is installed with the `cxfs_client` software product and that has a minimal implementation of CXFS that runs the CXFS client control daemon (`cxfs_client`). The client-only nodes in the pool do not maintain a local synchronized copy of the full cluster database. Instead, one of the daemons running on a server-capable administration node provides relevant database information to the client-only nodes. If the set of server-capable administration nodes changes, another node may become responsible for updating the client-only nodes. This node can run any supported operating system.

  This node can safely mount CXFS filesystems but it cannot become a CXFS metadata server or perform cluster administration. Client-only nodes retrieve the information necessary for their tasks by communicating with a server-capable administration node. This node does not contain a copy of the cluster database.

  For more details, see:

  - "CXFS Control Daemons" on page 41
  - Appendix A, "CXFS Software Architecture" on page 419
  - *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*

For each CXFS filesystem, one server-capable administration node is responsible for updating that filesystem's metadata. This node is referred to as the *metadata server*.

Multiple server-capable administration nodes can be defined as *potential metadata servers* for a given CXFS filesystem, but only one node per filesystem is chosen to function as the *active metadata server*, based on various factors. There can be multiple active metadata servers in the cluster, one per CXFS filesystem.

All other nodes that mount a CXFS filesystem function as CXFS *clients*. A server-capable administration node can function at any point in time as either an active metadata server or a CXFS client, depending upon how it is configured and whether it is chosen to function as the active metadata server.

Note: Do not confuse *CXFS client* and *metadata server* with the traditional data-path client/server model used by network filesystems. Only the metadata information passes through the metadata server via the private Ethernet network; the data is passed directly to and from storage on the CXFS client via the Fibre Channel connection.

The metadata server must perform cluster-coordination functions such as the following:

- Metadata logging
- File locking
- Buffer coherency
- Filesystem block allocation

All CXFS requests for metadata are routed over a TCP/IP network and through the metadata server, and all changes to metadata are sent to the metadata server. The metadata server uses the advanced XFS journal features to log the metadata changes. Because the size of the metadata is typically small, the bandwidth of a fast Ethernet local area network (LAN) is generally sufficient for the metadata traffic.

The operations to the CXFS metadata server are typically infrequent compared with the data operations directly to the LUNs. For example, opening a file causes a request for the file information from the metadata server. After the file is open, a process can usually read and write the file many times without additional metadata requests. When the file size or other metadata attributes for the file change, this triggers a metadata operation.

The following rules apply:

- If another potential metadata server exists, recovery will take place. For more information, see "Recovery" on page 25.
- If the last potential metadata server for a filesystem goes down while there are active CXFS clients, all of the clients will be forced out of the filesystem.
- If you are exporting the CXFS filesystem to be used with other NFS clients, the filesystem must be exported from the active metadata server for best performance. For more information on NFS exporting of CXFS filesystems, see "CXFS Mount Scripts" on page 291.

- There should be an odd number of server-capable administration nodes with CXFS services running for quorum calculation purposes. If you have a cluster with more than two nodes, define a CXFS tiebreaker node. See "CXFS Tiebreaker" on page 24 and "Use an Odd Number of Server-Capable Administration Nodes" on page 48.

Figure 1-2 shows nodes in a pool that are installed with the `cluster_admin` software product and others that are installed with the `cxfs_client` software product. Only those nodes with the `cluster_admin` software product have the `fs2d` daemon and therefore a copy of the cluster database. (For more information, see Table 1-2 on page 40 and Table 1-3 on page 41.)

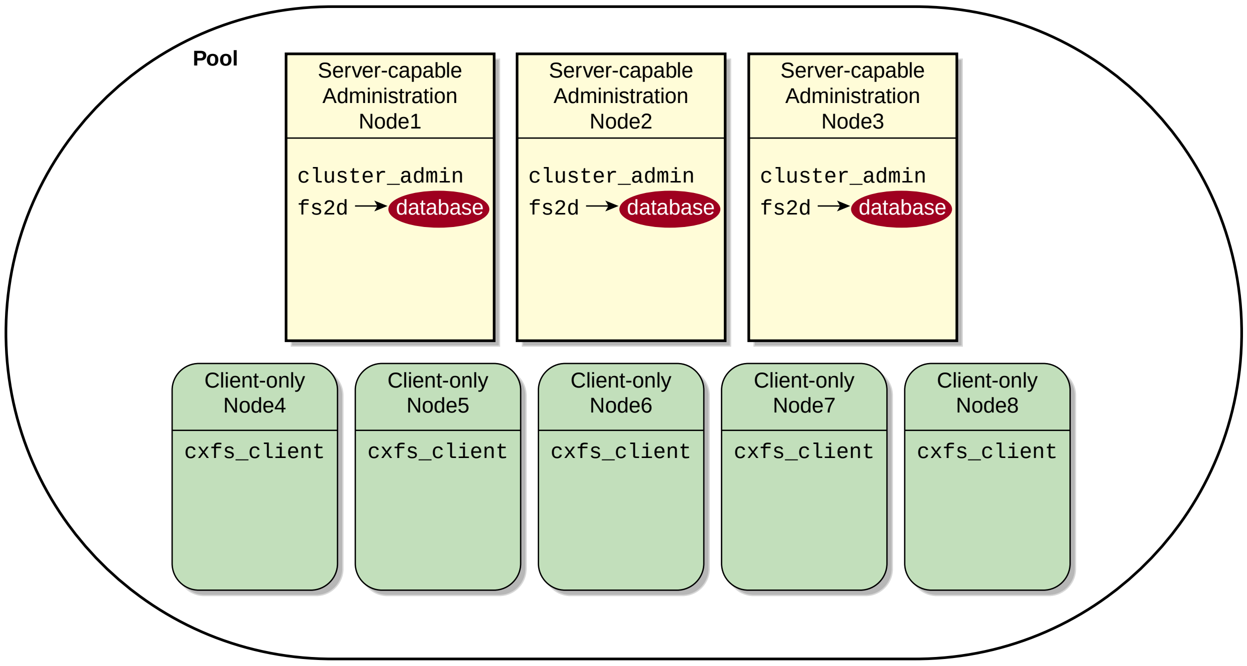


Figure 1-2 Installation Differences

A *standby node* is a server-capable administration node that is configured as a potential metadata server for a given filesystem.

Ideally, all server-capable administration nodes will run the same version of the operating system. However, SGI supports a policy for CXFS that permits a rolling upgrade; see "CXFS Release Versions and Rolling Upgrades" on page 277.

The following figures show a few configuration possibilities. The potential metadata servers are required to be server-capable administration nodes; the other nodes should be client-only nodes.

Figure 1-3 shows a very simple configuration with a single metadata server and no potential (standby) metadata servers; recovery and relocation do not apply to this cluster. The database exists only on the server-capable administration node. The client-only nodes could be running any supported operating system.

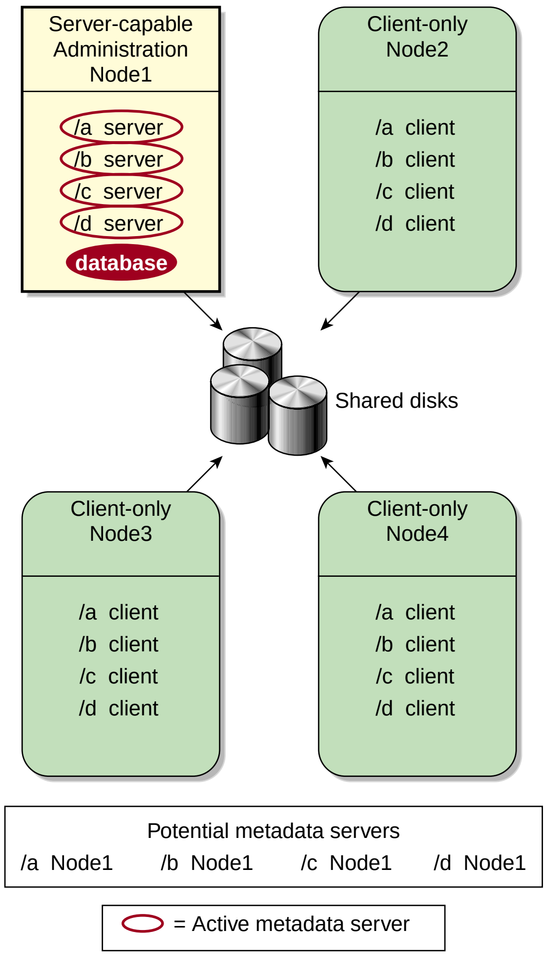


Figure 1-3 One Metadata Server (No Relocation or Recovery)

Figure 1-4 shows a configuration with two server-capable administration nodes, both of which are potential metadata servers for filesystems `/a` and `/b`:

- Node1 is the active metadata server for `/a`
- Node2 is the active metadata server for `/b`

Neither Node1 nor Node2 runs applications because they are both potential metadata servers. For simplicity, the figure shows one client-only node, but there could be many. One client-only node should be the tiebreaker so that the configuration remains stable if one of the server-capable administration nodes is disabled.

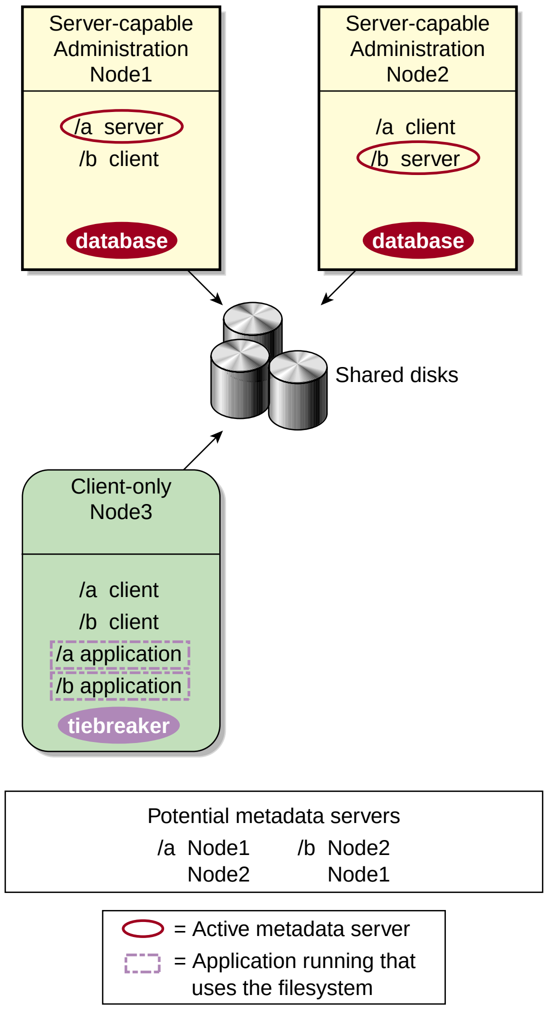


Figure 1-4 Two Metadata Servers for Two Filesystems

Figure 1-5 shows three server-capable administration nodes as active metadata servers. Node1 is the active metadata server for both filesystems `/a` and `/b`. If Node1 failed, Node2 would take over as the active metadata server according to the list of potential metadata servers for filesystems `/a` and `/b`.

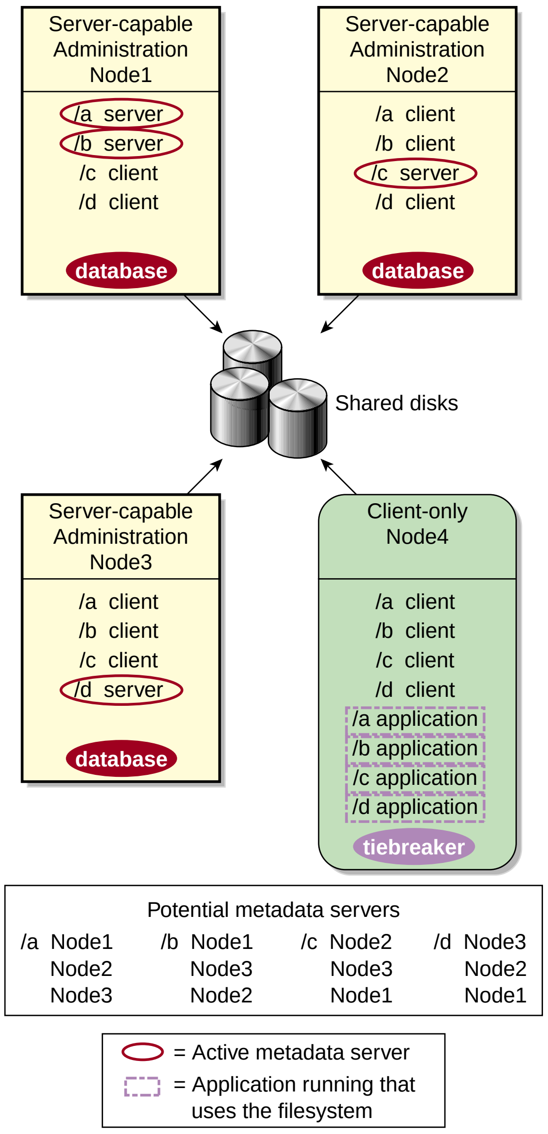


Figure 1-5 Three Metadata Servers for Four Filesystems

Membership

The nodes in a cluster must act together to provide a service. To act in a coordinated fashion, each node must know about all the other nodes currently active and providing the service. The set of nodes that are currently working together to provide a service is called a *membership*:

- *Cluster database membership* is the group of server-capable administration nodes that are accessible to each other. (Client-only nodes are not eligible for cluster database membership.) The nodes that are part of the the cluster database membership work together to coordinate configuration changes to the cluster database. One node is chosen to be the *quorum master*, which is the node that propagates the cluster database to the other server-capable administration nodes in the pool.
- *CXFS kernel membership* is the group of CXFS nodes in the cluster that can actively share filesystems, as determined by the the CXFS kernel, which manages membership and CXFS kernel heartbeating. The CXFS kernel membership may be a subset of the nodes defined in a cluster. All nodes in the cluster are eligible for CXFS kernel membership.

CXFS kernel heartbeat messages are exchanged via a private network so that each node can verify each membership:

- Every second, each server-capable administration node sends a single multicast packet monitored by all of the other server-capable administration and client-only nodes
- Every second, each client-only node sends a single multicast packet monitored by all of the server-capable administration nodes

A *heartbeat timeout* occurs when a node does not receive a heartbeat packet from another node within a predetermined period of time. Timeouts can happen in either direction, although they most frequently are seen when the server-capable administration node fails to receive a multicast heartbeat packet from a client-only node.

Private Network

A *private network* is one that is dedicated to cluster communication and is accessible by administrators but not by users.

Note: A virtual local area network (VLAN) is not supported for a private network.

CXFS uses the private network for the following:

- CXFS kernel heartbeat
- Cluster database heartbeat
- CXFS filesystem metadata traffic
- Cluster database replication
- Communication between the cluster database master and the clients
- Communication between the `cxfs_admin` configuration command and the cluster database master

Even small variations in heartbeat timing can cause problems. If there are delays in receiving heartbeat messages, the cluster software may determine that a node is not responding and therefore revoke its CXFS kernel membership; this causes it to either be reset or disconnected, depending upon the configuration.

Rebooting network equipment can cause the nodes in a cluster to lose communication and may result in the loss of CXFS kernel membership and/or cluster database membership; the cluster will move into a degraded state or shut down if communication among nodes is lost. Using a private network limits the traffic on the network and therefore will help avoid unnecessary resets or disconnects. Also, a network with restricted access is safer than one with user access because the messaging protocol does not prevent *snooping* (illicit viewing) or *spoofing* (in which one machine on the network masquerades as another).

Therefore, because the performance and security characteristics of a public network could cause problems in the cluster and because CXFS kernel heartbeat and cluster database heartbeat are very timing-dependent, **a private network is required**. The private network should be used for metadata traffic only. (Although the primary network must be private, the backup network may be public.)

The private network must be connected to all nodes, and all nodes must be configured to use the same subnet for that network.

For more information, see:

- "Fix Network Issues First" on page 44
- "Use a Private Network" on page 45
- Appendix B, "Memberships and Quorums" on page 429

- "Network Failover Tasks with `cxfs_admin`" on page 262
- Appendix C, "IP Filtering for the CXFS Private Network" on page 445

Data Integrity Protection

A failed node must be isolated from the shared filesystems so that it cannot corrupt data. CXFS uses the following methods in various combinations to isolate failed nodes:

- System reset, which performs a system reset via a system controller. Reset should always be used for server-capable administration nodes.
- I/O fencing, which disables a node's Fibre Channel ports so that it cannot access I/O devices and therefore cannot corrupt data in the shared CXFS filesystem. When fencing is applied, the rest of the cluster can begin immediate recovery.

  Note: I/O fencing differs from zoning. *Fencing* erects a barrier between a node and shared cluster resources. *Zoning* i defines logical subsets of the switch (zones), with the ability to include or exclude nodes and media from a given zone. A node can access only media that are included in its zone. Zoning is one possible implementation of fencing. Because zoning implementation is complex and does not have uniform availability across switches, SGI chose to implement a simpler form of fencing: enabling/disabling a node's Fibre Channel ports.

- System shutdown for client-only nodes that use static CXFS kernel heartbeat monitoring.
- Cluster tiebreaker on a stable client-only node.

For more information, see "Protect Data Integrity on All Nodes" on page 50,

CXFS Tiebreaker

The *CXFS tiebreaker node* is used in the process of computing the CXFS kernel membership for the cluster when exactly half the server-capable administration nodes in the cluster are up and can communicate with each other.

The tiebreaker is required for all clusters with more than one server-capable administration node and at least one client-only node. You should choose a reliable client-only node as the tiebreaker; there is no default. For a cluster that consists of

only four or more server-capable administration nodes, you should choose one of them as the tiebreaker; this is the only situation in which you should choose a server-capable administration node as a tiebreaker.

The tiebreaker is required in addition to I/O fencing or system reset; see "Data Integrity Protection" on page 24.

Relocation

Relocation is the process by which the metadata server moves from one node to another due to an administrative action; other services on the first node are not interrupted.

Note: Relocation is disabled by default.

CXFS kernel membership is not affected by relocation. However, users may experience a degradation in filesystem performance while the metadata server is relocating.

An example of a relocation trigger is when the system administrator uses the GUI or `cxfs_admin` to relocate the metadata server.

To use relocation, you must enable relocation on the active metadata server. See "CXFS Relocation Capability" on page 283.

See also "Use Relocation and Recovery Properly" on page 69.

Recovery

Recovery is the process by which the metadata server moves from one node to another due to an interruption in services on the first node. If the node acting as the metadata server for a filesystem dies, another node in the list of potential metadata servers will be chosen as the new metadata server. This assumes that at least two potential metadata servers are listed when you define a filesystem.

The metadata server that is chosen must be a filesystem client; other filesystem clients will experience a delay during the recovery process. Each filesystem will take time to recover, depending upon the number of active inodes; the total delay is the sum of time required to recover each filesystem. Depending on how active the filesystems are at the time of recovery, the total delay could take up to several minutes per filesystem.

If a CXFS client fails, the metadata server will clean up after the client. Other CXFS clients may experience a delay during this process. A delay depends on what tokens, if any, that the deceased client holds. If the client has no tokens, then there will be no delay; if the client is holding a token that must be revoked in order to allow another client to proceed, then the other client will be held up until recovery returns the failed nodes tokens (for example, in the case where the client has the write token and another client wants to read). The actual length of the delay depends upon the following:

- The total number of exported inodes on the metadata server
- CXFS kernel membership situation
- Whether any metadata servers have died
- Where the metadata servers are in the recovery-order relative to recovering this filesystem

The CXFS client that failed is not allowed to rejoin the CXFS kernel membership until all metadata servers have finished cleaning up after the client.

The following are examples of recovery triggers:

- A metadata server panics
- A metadata server locks up, causing CXFS kernel heartbeat timeouts on metadata clients
- A metadata server loses connection to the CXFS private network

Figure 1-6 describes the difference between relocation and recovery for a metadata server. (Remember that there is one active metadata server per CXFS filesystem. There can be multiple active metadata servers within a cluster, one for each CXFS filesystem.)

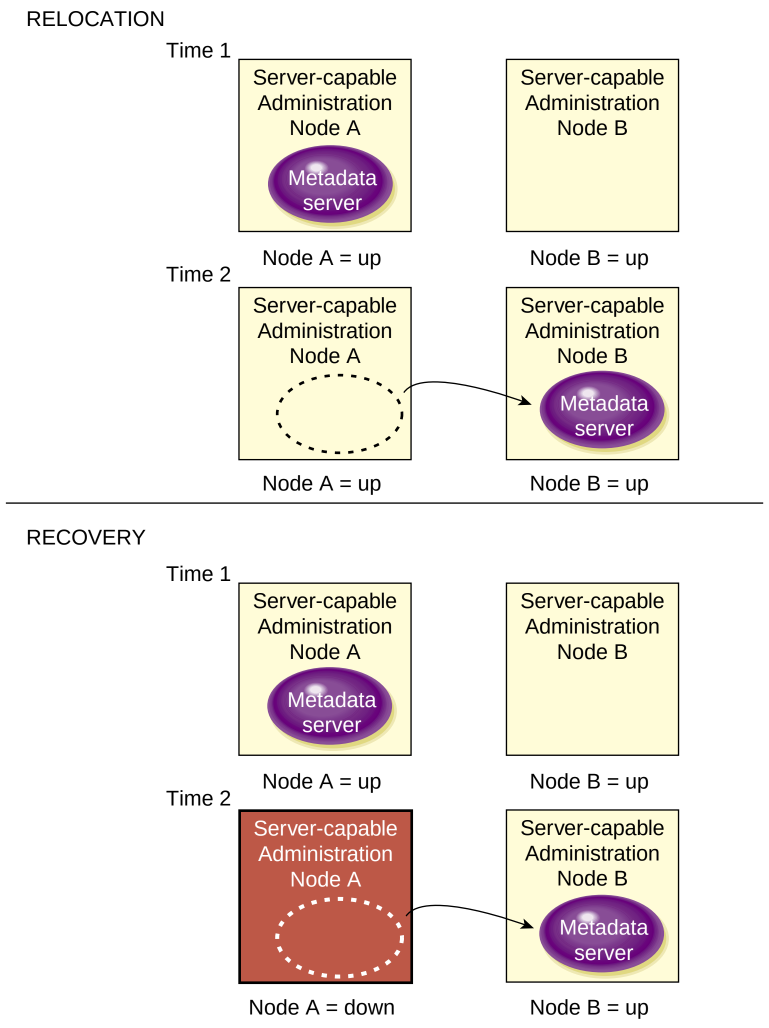

RELOCATION
Time 1
Server-capable Administration Node A
Metadata server
Server-capable Administration Node B
Node A = up
Node B = up
Time 2
Server-capable Administration Node A
Server-capable Administration Node B
Metadata server
Node A = up
Node B = up
RECOVERY
Time 1
Server-capable Administration Node A
Metadata server
Server-capable Administration Node B
Node A = up
Node B = up
Time 2
Server-capable Administration Node A
Server-capable Administration Node B
Metadata server
Node A = down
Node B = up

Figure 1-6 Relocation versus Recovery

See also "Use Relocation and Recovery Properly" on page 69.

CXFS Services

To *start CXFS services* enables a node, which changes a flag in the cluster database by performing an administrative task using the CXFS GUI or `cxfs_admin`.

To *stop CXFS services* disables a node, which changes a flag in the cluster database, by performing an administrative task using the GUI or `cxfs_admin`:

To *shutdown CXFS services* withdraws a node from the CXFS kernel membership, either due to the fact that the node has failed somehow or by issuing an `admin cxfs_stop` command. The node remains enabled in the cluster database. See .

Starting, stopping, or shutting down CXFS services does not affect the daemons involved. For information about the daemons that control CXFS services, see "CXFS Control Daemons" on page 41.

- "CXFS Control Daemons" on page 41
- "Start CXFS Services with the GUI" on page 189
- "Stop CXFS Services with the GUI" on page 189
- "Enable a Node with `cxfs_admin`" on page 244
- "Disable a Node with `cxfs_admin`" on page 244
- "Forced CXFS Shutdown: Revoke Membership of Local Node" on page 299

CXFS Interoperability With Other Products and Features

CXFS is released as part of the SGI InfiniteStorage Software Platform (ISSP). ISSP combines SGI storage software products (such as CXFS, DMF, NFS, Samba, XFS, XVM, and Appliance Manager) into a single distribution that is tested for interoperability.

This section discusses the following:

- "Data Migration Facility (DMF)" on page 29
- "Highly Available Services" on page 29
- "GPT Labels Overview" on page 29
- "Guaranteed-Rate I/O (GRIO) Version 2 Overview" on page 30
- "Storage Management" on page 30

- "Volume Management with XVM" on page 32
- "XVM Failover Version 2 Overview" on page 33

Data Migration Facility (DMF)

CXFS supports the use of hierarchical storage management (HSM) products through the data management application programming interface (DMAPI), also know as X/Open Data Storage Management Specification (XSDM). An example of an HSM product is the Data Migration Facility (DMF). DMF is the only HSM product supported with CXFS.

The DMF server runs on a CXFS server-capable administration node; CXFS client-only nodes may be DMF clients. For more information, see:

- "Using Hierarchical Storage Management (HSM) Products" on page 293
- *DMF Administrator's Guide for SGI InfiniteStorage*

Highly Available Services

Linux-HA Heartbeat from the Linux-HA project provides high-availability services for CXFS clusters. The CXFS plug-in reports on whether the metadata server for a filesystem on a given node is running, not running, or has an error. Linux-HA Heartbeat then moves services accordingly, based upon constraints that the customer sets. For more information about the CXFS plug-in, see *SGI InfiniteStorage High Availability Using Linux-HA Heartbeat*. For more information about Linux-HA Heartbeat, see:

http://linux-ha.org/

GPT Labels Overview

CXFS supports XVM labels on LUNs with GUID partition table (GPT) labels as well LUNs with SGI disk volume header (DVH) labels. A CXFS cluster can contain LUNs that have GPT labels and LUNs that have DVH labels. You can create these labels on SGI ProPack server-capable administration nodes and Linux third-party clients. For more information, see "GPT Labels" on page 322.

Guaranteed-Rate I/O (GRIO) Version 2 Overview

CXFS supports guaranteed-rate I/O (GRIO) version 2 clients on all platforms, with a GRIO server on a server-capable administration node. (GRIO is disabled by default on CXFS Linux client-only nodes. For more information, see the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.)

After GRIO is installed in a cluster, you can run the following commands from any node in the cluster as a superuser:

- `grioadmin` provides stream and bandwidth management
- `grioqos` is the comprehensive stream quality-of-service monitoring tool

Run the above tools with the `-h` (help) option for a full description of all available options.

The paths to the GRIO commands differ by platform. See Appendix D, "Path Summary" on page 449 and the appendix in the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*. For details about GRIO, see the *Guaranteed-Rate I/O Version 2 Guide*. For other platform-specific limitations and considerations, see the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

Storage Management

This section describes the CXFS differences for backups, NFS, quotas, and Samba.

Storage Backups

CXFS enables the use of commercial backup packages such as VERITAS NetBackup and Legato NetWorker for backups that are free from the local area network (LAN), which allows the backup server to consolidate the backup work onto a backup server while the data passes through a storage area network (SAN), rather than through a lower-speed LAN.

For example, a backup package can run on a host on the SAN designated as a backup server. This server can use attached tape drives and channel connections to the SAN storages. It runs the backup application, which views the filesystems through CXFS and transfers the data directly from the LUNs, through the backup server, to the tape drives.

This allows the backup bandwidth to scale to match the storage size, even for very large filesystems. You can increase the number of LUN channels, the size of the backup server, and the number of tape channels to meet the backup-bandwidth requirements.

Note: Do not run backups on a client node because it causes heavy use of non-swappable kernel memory on the metadata server. During a backup, every inode on the filesystem is visited, and if done from a client, it imposes a huge load on the metadata server. The metadata server may experience typical out-of-memory symptoms, and in the worst case can even become unresponsive or crash.

NFS

You can put an NFS server on top of CXFS so that computer systems that are not part of the cluster can share the filesystems. You should run the NFS server on the CXFS active metadata server for optimal performance.

Quotas

XFS quotas are supported. However, the quota mount options must be the same on all mounts of the filesystem. You can administer quotas from any IRIX or Linux node in the cluster.

For more information about setting quotas, see *XFS for Linux Administration* and *IRIX Admin: Disks and Filesystems*.

Samba

You can run Samba on top of CXFS, allowing Windows machines to support CXFS and have access to the filesystem. Samba should run on the active metadata server for optimal performance. You should not serve the same CXFS filesystem from multiple nodes in a cluster.

The architecture of Samba assumes that each share is exported by a single server. Because all Samba client accesses to files and directories in that share are directed through a single Samba server, the Samba server is able to maintain private metadata state to implement the required concurrent access controls (in particular, share modes, write caching and oplock states). This metadata is not necessarily promulgated to the filesystem and there is no protocol for multiple Samba servers exporting the same share to communicate this information between them.

Running multiple Samba servers on one or more CXFS clients exporting a single share that maps to a common underlying filesystem has the following risks:

- File data corruption from writer-writer concurrency
- Application failure due to inconsistent file data from writer-reader concurrency

These problems do not occur when a single Samba server is deployed, because that server maintains a consistent view of the metadata used to control concurrent access across all Samba clients.

It may be possible to deploy multiple Samba servers under one of the following circumstances:

- There are no writers, so a read-only share is exported
- Application-level protocols and/or work-flow guarantee that only one application is ever writing a file, and concurrent file writing and reading does not take place

---

⚠ **Caution:** The onus is on the customer to ensure these conditions are met, as there is nothing in the Samba architecture to verify it. Therefore, SGI recommends that you do not use multiple Samba servers.

Volume Management with XVM

CXFS uses the XVM volume manager. XVM can combine many LUNs into high transaction rate, high bandwidth, and highly reliable filesystems. CXFS uses XVM to provide the following:

- Striping
- Mirroring
- Concatenation
- Advanced recovery features

Note: The `xvm` command must be run on a server-capable administration node. If you try to run an XVM command before starting the CXFS daemons, you will get a warning message and be put into XVM's *local domain*.

When you are in XVM's local domain, you could define your filesystems, but then when you later start up CXFS you will not see the filesystems. When you start up CXFS, XVM will switch to *cluster domain* and the filesystems will not be recognized because you defined them in local domain; to use them in the cluster domain, you would have to use the `give` command. Therefore, it is better to define the volumes directly in the cluster domain.

For more information, see the *XVM Volume Manager Administrator's Guide*.

XVM Failover Version 2 Overview

CXFS supports XVM failover V2 on all platforms. XVM failover V2 allows XVM to use multiple paths to LUNs in order to provide fault tolerance and static load balancing. Paths to LUNs are representations of links from the client HBA ports to the fabric and from the fabric to the RAID controller; they do not represent the path through the fabric itself.

SGI supports `SGIAVT` mode for XVM failover V2 because it will not cause LUN ownership changes, unlike `AVT` mode. `SGIRDAC` and `RDAC` mode do not provide failover support. See "Changing `SGIRDAC` Mode to `SGIAVT` Mode for SGI RAID" on page 598.

In general, you want to evenly distribute the I/O to LUNs across all available host bus adapters and RAID controllers and attempt to avoid blocking in the SAN fabric. The ideal case, from a performance standpoint, is to use as many paths as connection endpoints between two nodes in the fabric as possible with as few blocking paths as possible in the intervening SAN fabric.

For more information, see "XVM Failover V2" on page 314.

Hardware and Software Requirements for Server-Capable Administration Nodes

A CXFS cluster is supported with as many as 64 nodes, of which as many as 16 can be server-capable administration nodes.

CXFS requires the following hardware and software for server-capable administration nodes, as specified in the release notes:

- An odd number of server-capable administration nodes or a tiebreaker with an even number of potential metadata servers. (SGI recommends that you always configure a stable client-only node as a tiebreaker, even for a cluster with an odd number of nodes.)
- Server-capable administration nodes that are dedicated to CXFS and filesystems work. See "Use Server-Capable Administration Nodes that are Dedicated to CXFS Work" on page 48.
- Server-capable administration nodes running the SGI ProPack for Linux operating system listed in the release notes on all SGI Altix ia64 systems or all SGI Altix XE x86_64 systems. (For client-only nodes, see the release notes and *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.)
- At least one host bus adapter (HBA) as listed in the release notes.
- A supported SAN hardware configuration. For details about supported hardware, see the Entitlement Sheet that accompanies the release materials. (Using unsupported hardware constitutes a breach of the CXFS license.)
- Use a network switch of at least 100baseT. (A network hub is not supported.)
- A private 100baseT or Gigabit Ethernet TCP/IP network connected to each node.

  Note: When using Gigabit Ethernet, do not use jumbo frames.

- System reset capability and/or supported Fibre Channel switches. For supported switches, see the release notes. (Either system reset or I/O fencing is required for all nodes.)

  Note: If you use I/O fencing and `ipfilterd` on a node, the `ipfilterd` configuration must allow communication between the node and the `telnet` port on the switch.

- RAID hardware as specified in the release notes.
- Adequate compute power for CXFS nodes, particularly server-capable administration nodes, which must deal with the required communication and I/O overhead. There should be at least 2 GB of RAM on the system. To avoid problems during metadata server recovery/relocation, all potential metadata servers should have as much memory as the active metadata server. See "Provide Enough Memory" on page 45.
- Licenses for CXFS and XVM. See the general release notes Chapter 5, "CXFS License Keys" on page 89.
- The XVM volume manager

CXFS Software Products Installed on Server-Capable Administration Nodes

The following software products are installed on a server-capable administration node:

- Application binaries, documentation, and support tools:

  ```
  cluster_admin
  cluster_control
  cluster_services
  cxfs_admin
  cxfs-doc
  cxfs_cluster
  cxfs_util
  cxfs-xvm-cmds
  ```

- Enhanced XFS:

  `sgi-enhancedxfs-kmp-`*KERNELTYPE*

- GRIOv2 software:

  ```
  grio2-cmds
  grio2-server
  ```

- GUI tools:

  ```
  cxfs-sysadm_base-client
  cxfs-sysadm_base-lib
  cxfs-sysadm_base-server
  ```

```
cxfs-sysadm_cluster_base-client
cxfs-sysadm_cluster_base-server
cxfs-sysadm_cxfs-client
cxfs-sysadm_cxfs-server
cxfs-sysadm_cxfs-web
cxfs-sysadm_xvm-client
cxfs-sysadm_xvm-server
cxfs-sysadm_xvm-web
```

- Kernel module:

  `sgi-cxfs-server-kmp-`*KERNELTYPE*

For information about client-only nodes, see *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

CXFS Tools

This section discusses the following:

- "Configuration Commands" on page 36
- "Cluster Administration Daemons" on page 40
- "CXFS Control Daemons" on page 41
- "Other Commonly Used Administrative Commands" on page 41

See also "CXFS and Cluster Administration Initialization Commands" on page 285.

Configuration Commands

You can perform CXFS configuration tasks using the GUI or `cxfs_admin`, shown in Table 1-1. These tools update the cluster database, which persistently stores metadata and cluster configuration information. After the associated changes are applied to all online database copies in the pool, the view area in the GUI will be updated. You can use the GUI or the `cxfs_admin` command to view the state of the database. (The database is a collection of files, which you cannot access directly.)

Table 1-1 Configuration Commands

Command	Software Product	Description
`cxfs_admin`	`cxfs_admin`	Configures and administers the cluster database.
`cxfsmgr`	`cxfs-sysadm_cxfs-client`	Invokes the CXFS GUI, which provides access to the tasks that help you set up and administer your CXFS filesystems and provides icons representing status and structure.
`xvmgr`	`cxfs-sysadm_xvm-client`	Invokes the XVM GUI, which provides access to the tasks that help you set up and administer your logical volumes and provides icons representing status and structure. You access the XVM GUI as part of the CXFS GUI.

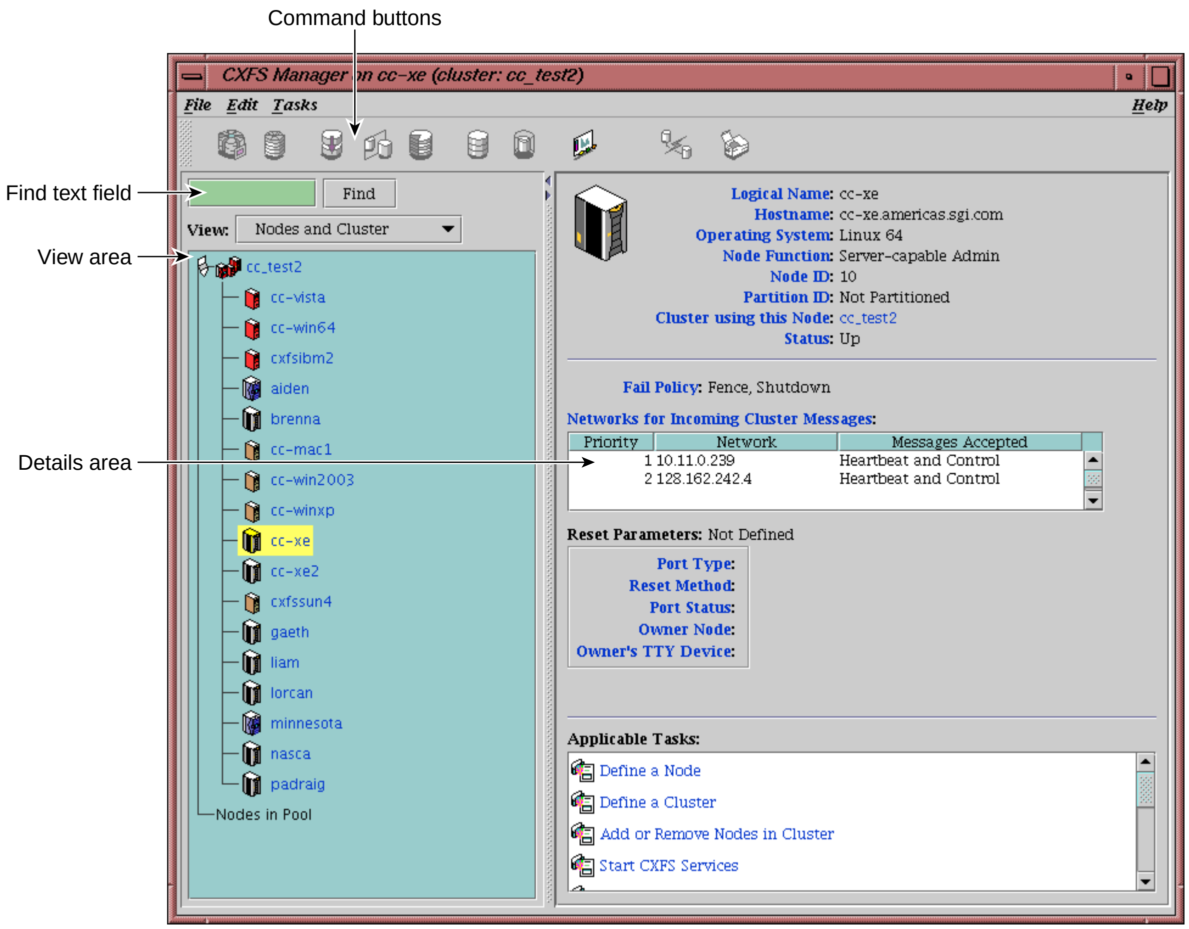
CXFS GUI

The `cxfsmgr` command invokes the CXFS Manager graphical user interface (GUI). The GUI lets you set up and administer CXFS filesystems and XVM logical volumes. It also provides icons representing status and structure. The GUI provides the following features:

- You can click any blue text to get more information about that concept or input field. Online help is also provided with the **Help** button.
- The cluster state is shown visually for instant recognition of status and problems.
- The state is updated dynamically for continuous system monitoring.
- All inputs are checked for correct syntax before attempting to change the cluster configuration information. In every task, the cluster configuration will not update until you click **OK**.
- Tasks take you step-by-step through configuration and management operations, making actual changes to the cluster configuration as you complete a task.
- The graphical tools can be run securely and remotely on any computer that has a Java-enabled web browser, including Windows and Linux computers and laptops.

Figure 1-7 shows an example GUI screen. For more information, see Chapter 10, "Reference to GUI Tasks" on page 147.

Note: The GUI must be connected to a server-capable administration node, but it can be launched elsewhere; see "Starting the GUI" on page 148.



Logical Name: cc-xe
Hostname: cc-xe.americas.sgi.com
Operating System: Linux 64
Node Function: Server-capable Admin
Node ID: 10
Partition ID: Not Partitioned
Cluster using this Node: cc_test2
Status: Up

Fail Policy: Fence, Shutdown

Networks for Incoming Cluster Messages:

Priority	Network	Messages Accepted
1	10.11.0.239	Heartbeat and Control
2	128.162.242.4	Heartbeat and Control

Reset Parameters: Not Defined

Port Type:
Reset Method:
Port Status:
Owner Node:
Owner's TTY Device:

Applicable Tasks:

Define a Node
Define a Cluster
Add or Remove Nodes in Cluster
Start CXFS Services

Figure 1-7 CXFS Manager GUI

`cxfs_admin` Command-Line Configuration Tool

`cxfs_admin` lets you set up and administer CXFS filesystems and XVM logical volumes using command-line mode. It shows both the static and dynamic cluster states. This command is available on nodes that have the appropriate access and network connections. The `cxfs_admin` command provides the following features:

- Waits for a command to be completed before continuing and provides `<TAB>` completion of commands.
- Validates all input before a command is completed.
- Provides a step-by-step mode with auto-prompting and scripting capabilities.
- Provides better state information than the GUI, or `cxfs_info`.
- Provides certain functions that are not available with the GUI.
- Provides a convenient method for performing basic configuration tasks or isolated single tasks in a production environment.
- Provides the ability to run scripts to automate some cluster administration tasks. You can use the `config` command in `cxfs_admin` to output the current configuration to a file and later recreate the configuration by using a command line option.

For more information, see Chapter 11, "Reference to `cxfs_admin` Tasks" on page 217.

Cluster Administration Daemons

Table 1-3 lists the set of daemons that provide cluster infrastructure on a server-capable administration node.

Table 1-2 Cluster Administration Daemons

Daemon	Software Product	Description
`cad`	`cluster_admin`	Provides administration services for the CXFS GUI.
`cmond`	`cluster_admin`	Monitors the other cluster administration and CXFS control daemons on the local node and restarts them on failure.
`crsd`	`cluster_control`	Provides reset connection monitoring and control.
`fs2d`	`cluster_admin`	Manages the cluster database (CDB). Keeps each copy in synchronization on all server-capable administration nodes in the pool and exports configuration to client-only nodes.

You can start these daemons manually or automatically upon reboot by using the `chkconfig` command. For more information, see:

- "`chkconfig` Arguments" on page 288
- "Manually Starting/Stopping CXFS" on page 308
- Appendix A, "CXFS Software Architecture" on page 419

CXFS Control Daemons

Table 1-3 lists the daemons that control CXFS nodes.

Table 1-3 CXFS Control Daemons

Daemon	Software Product	Description
`clconfd`	`cxfs_cluster`	Controls server-capable administration nodes. Does the following: • Obtains the cluster configuration from the `fs2d` daemon and manages the local server-capable administration node's CXFS kernel membership services and filesystems accordingly • Obtains membership and filesystem status from the kernel • Issues reset commands to the `crsd` daemon • Issues I/O fencing commands to configured Fibre Channel switches
`cxfs_client`	`cxfs_client`	Controls client-only nodes. Manages the local kernel's CXFS kernel membership services accordingly.

You can start these daemons manually or automatically upon reboot by using the `chkconfig` command. For more information, see:

- "`chkconfig` Arguments" on page 288
- "Manually Starting/Stopping CXFS" on page 308
- Appendix A, "CXFS Software Architecture" on page 419

Other Commonly Used Administrative Commands

Table 1-4 summarizes the other CXFS commands of most use on a server-capable administration node. For information about client-only nodes, see *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

Table 1-4 Other Administrative Commands

Command	Software Product	Description
`cbeutil`	`cluster_admin`	Accesses the back-end cluster database.
`cdbBackup`	`cluster_admin`	Backs up the cluster database.
`cdbRestore`	`cluster_admin`	Restores the cluster database.
`cdbconfig`	`cluster_admin`	Configures the cluster database.
`cdbutil`	`cluster_admin`	Accesses the cluster database by means of commands that correspond to functions in the `libcdb` library.
`clconf_info`	`cxfs_cluster`	Provides static and dynamic information about the cluster.
`clconf_stats`	`cxfs_cluster`	Provides CXFS kernel heartbeat statistics for cluster.
`cms_failconf`	`cluster_control`	Configures the action taken by the surviving nodes when a CXFS node loses membership. Normally, you will use the GUI or `cxfs_admin` to perform these actions.
`cxfs-config`	`cxfs_util`	Validates configuration information in a CXFS cluster.
`cxfsdump`	`cxfs_util`	Gathers configuration information in a CXFS cluster for diagnostic purposes.
`cxfslicense`	`cxfs_util`	Reports the status of license keys.
`cxfs_shutdown`	`cxfs_cluster`	Shuts down CXFS in the kernel and CXFS daemons.
`haStatus`	`cluster_services`	Obtains I/O fencing configuration and status information.
`hafence`	`cxfs_cluster`	Administer the CXFS I/O fencing configuration stored in the cluster database. Normally, you will perform this task using the GUI or `cxfs_admin`.
`sysadmd`	`cxfs-sysadm_base-server`	Allows clients to perform remote system administration for the GUI server.

Best Practices

This chapter summarizes configuration and administration best-practices information for CXFS:

- "Configuration Best Practices" on page 43
- "Administration Best Practices" on page 65

For the latest information and a matrix of supported CXFS and operating system software, see http://support.sgi.com/content_request/838562/index.html on Supportfolio.

Configuration Best Practices

This section discusses the following configuration topics:

- "Fix Network Issues First" on page 44
- "Save the Current Configuration Before Making Changes" on page 44
- "Use a Private Network" on page 45
- "Use the Same Architecture for All Server-Capable Administration Nodes" on page 45
- "Provide Enough Memory" on page 45
- "Use CXFS Configuration Tools Appropriately" on page 46
- "Ensure Cluster Database Membership Quorum Stability" on page 46
- "Be Consistent in Configuration" on page 47
- "Use the Correct Mix of Software Releases" on page 47
- "Use Server-Capable Administration Nodes that are Dedicated to CXFS Work" on page 48
- "Use an Odd Number of Server-Capable Administration Nodes" on page 48
- "Make Most Nodes Client-Only" on page 49

- "Use a Client-Only Tiebreaker" on page 49
- "Protect Data Integrity on All Nodes" on page 50
- "Avoid CXFS Kernel Heartbeat Issues on Large SGI Altix Systems" on page 60
- "Minimize the Number of Switches" on page 62
- "Form a Small Functional Cluster First" on page 62
- "Configure Filesystems Properly" on page 62
- "Use the Appropriate CXFS Kernel Heartbeat Monitoring" on page 63
- "Verify the Configuration" on page 64
- "Use the Recovery Timeout Mechanism" on page 64
- "Use Proper Storage Management Procedures" on page 65
- "Samba and CXFS" on page 65
- "DMF and CXFS" on page 65

Fix Network Issues First

If there are any network issues on the private network, fix them before trying to use CXFS. A stable private network is important for a stable CXFS cluster network. Ensure that you understand the information in "Hostname Resolution and Network Configuration Rules" on page 103.

Save the Current Configuration Before Making Changes

Before making significant changes to an existing CXFS configuration, run the the `build_cmgr_script` command to create a copy of the current database. If needed, you can then use the generated script to recreate the cluster database after performing a `cdbreinit`. See "Creating a `cmgr` Script Automatically" on page 568.

Use a Private Network

You are required to use a private network for CXFS metadata traffic:

- The private network is used for metadata traffic and should not be used for other kinds of traffic.
- A stable private network is important for a stable CXFS cluster environment.
- Two or more clusters should not share the same private network. A separate private network switch is required for each cluster.
- The private network should contain at least a 100-Mbit network switch. A network hub is not supported and should not be used.
- All cluster nodes should be on the same physical network segment (that is, no routers between hosts and the switch).
- Use private (10.*x.x.x*, 176.16.*x.x*, or 192.168.*x.x*) network addresses (RFC 1918).
- The private network must be configured as the highest priority network for the cluster. The public network may be configured as a lower priority network to be used by CXFS network failover in case of a failure in the private network.
- When administering more than one CXFS cluster, use unique private network addresses for each cluster. If you have multiple clusters using the same public network as the backup CXFS metadata network, use unique cluster names and cluster IDs.
- A virtual local area network (VLAN) is not supported for a private network.

Use the Same Architecture for All Server-Capable Administration Nodes

All server-capable administration nodes within the cluster should have similar capabilities. Use all Altix ia64 systems or all Altix XE x86_64 systems. See also "Provide Enough Memory" on page 45.

Provide Enough Memory

There should be at least 2 GB of RAM on the system. A server-capable administration node must have at least 1 processor and 1 GB of memory more than what it would need for its normal workload (work other than CXFS). In general, this means that the minimum configuration would be 2 processors and 2 GB of memory. If the metadata server is also doing NFS or Samba serving, then more memory is recommended (and the `nbuf` and `ncsize` kernel parameters should be increased from their defaults). CXFS makes heavy use of memory for caching.

If a very large number of files (tens of thousands) are expected to be accessed at any one time, additional memory over the minimum is recommended to avoid throttling memory. Estimate the maximum number of inodes that will be accessed during a 2-minute window and size the server-capable administration node memory for that number. (The inode references are not persistent in memory and are removed after about 2 minutes of non-use.) Use the following general rule to determine the amount of memory required when the number of open files at any one time may be this large:

2 KB x *#inodes* = *server-capable_administration_node_memory*

In addition, about half of a CPU should be allocated for each Gigabit Ethernet interface on the system if it is expected to be run a close to full speed.

To avoid problems during metadata server recovery/relocation, all potential metadata servers should have as much memory as the active metadata server.

Use CXFS Configuration Tools Appropriately

The GUI provides a convenient display of a cluster and its components through the view area. You should use it to see your progress and to avoid adding or removing nodes too quickly. After defining a node, you should wait for it to appear in the view area before adding another node. After defining a cluster, you should wait for it to appear before you add nodes to it. If you make changes too quickly, errors can occur. For more information, see "Starting the GUI" on page 148.

Do not attempt to make simultaneous changes using `cxfs_admin` and the GUI. Use one tool at a time.

Ensure Cluster Database Membership Quorum Stability

The cluster database membership quorum must remain stable during the configuration process. If possible, use multiple windows to display the `fs2d_log` file for each server-capable administration node while performing configuration tasks. Enter the following:

```
admin# tail -f /var/cluster/ha/log/fs2d_log
```

Check the member count when it prints new quorums. Under normal circumstances, it should print a few messages when adding or deleting nodes, but it should stop within a few seconds after a new quorum is adopted. If not enough machines respond, there will not be a quorum. In this case, the database will not be propagated.

If you detect cluster database membership quorum problems, fix them before making other changes to the database. Try restarting the cluster administration daemons on the node that does not have the correct cluster database membership quorum, or on all nodes at the same time.

Enter the following on server-capable administration nodes:

```
admin# service cxfs stop
admin# service cxfs_cluster stop

admin# service cxfs_cluster start
admin# service cxfs start
```

Note: You could also use the `restart` option to stop and start.

Please provide the `fs2d` log files when reporting a cluster database membership quorum problem.

Be Consistent in Configuration

Be consistent in configuration files for all nodes and when configuring networks. Use the same names in the same order. See "Configuring System Files" on page 119.

Use the Correct Mix of Software Releases

Create a new cluster using server-capable administration nodes that have the same version of the OS release and the same version of CXFS installed.

All nodes should run the same level of CXFS and the same level of operating system software, according to platform type. To support upgrading without having to take the whole cluster down, nodes can temporarily run different CXFS releases during the upgrade process.

Caution: You must upgrade all server-capable administration nodes before upgrading any client-only nodes (servers must run the same release as client-only nodes or a later release.) Operating a cluster with clients running a mixture of older and newer CXFS versions will result in a performance loss. Relocation to a server-capable administration node that is running an older CXFS version is not supported.

For details, see the platform-specific release notes and "CXFS Release Versions and Rolling Upgrades" on page 277.

Use Server-Capable Administration Nodes that are Dedicated to CXFS Work

Server-capable administration nodes must be dedicated to CXFS and filesystems work (such as DMF, Samba, or NFS). Standard services (such as DNS and NIS) are permitted, but any other applications (such as analysis, simulation, and graphics) must be avoided.

Only dedicated nodes are supported as CXFS server-capable administration nodes. Running a server-capable administration node in a nondedicated manner will void the support contract. If the use of an application is desired on a server-capable administration node, SGI will provide a quotation to perform the following work:

- Audit the solution
- Design a supportable configuration
- Implement the changes

A statement of work will be created and implementation will begin after mutual agreement with the customer.

If additional products are required from SGI, the customer will be responsible for obtaining a quote and providing a purchase order before any corrective action begins. SGI will not correct unsupported configurations without compensation and reserves the right to terminate or suspend the support agreement.

Use an Odd Number of Server-Capable Administration Nodes

Use an odd number of server-capable administration nodes with CXFS services running and a client-only CXFS tiebreaker node if you have more than two nodes total in the cluster. See "Use a Client-Only Tiebreaker" on page 49.

Make Most Nodes Client-Only

You should define most nodes as client-only nodes and define just the nodes that may be used for CXFS metadata as server-capable administration nodes.

The advantage to using client-only nodes is that they do not keep a copy of the cluster database; they contact a server-capable administration node to get configuration information. It is easier and faster to keep the database synchronized on a small set of nodes, rather than on every node in the cluster. In addition, if there are issues, there will be a smaller set of nodes on which you must look for problem.

See "Transforming a Server-Capable Administration Node into a Client-Only Node" on page 290.

Use a Client-Only Tiebreaker

SGI recommends that you always define a stable client-only node as the CXFS tiebreaker for all clusters with more than one server-capable administration node and at least one client-only node.

Having a tiebreaker is critical when there are an even number of server-capable administration nodes. A tiebreaker avoids the problem of multiple-clusters being formed (a *split cluster*) while still allowing the cluster to continue if one of the metadata servers fails.

As long as there is a reliable client-only node in the cluster, a client-only node should be used as tiebreaker. Server-capable administration nodes are not recommended as tiebreaker nodes because these nodes always affect CXFS kernel membership.

The tiebreaker is of benefit in a cluster even with an odd number of server-capable administration nodes because when one of the server-capable administration nodes is removed from the cluster, it effectively becomes a cluster with an even-number of server-capable administration nodes.

Note the following:

- If exactly two server-capable administration nodes are configured and there are no client-only nodes, **neither** server-capable administration node should be set as the tiebreaker. (If one node was set as the tiebreaker and it failed, the other node would also shut down.)
- If exactly two server-capable administration nodes are configured and there is at least one client-only node, you should specify the client-only node as a tiebreaker.

If one of the server-capable administration nodes is the CXFS tiebreaker in a two-server-capable-node cluster, failure of that node or stopping the CXFS services on that node will result in a cluster-wide forced shutdown. If you use a client-only node as the tiebreaker, either server-capable administration node could fail but the cluster would remain operational via the other server-capable administration node.

- If there are an even number of servers and there is no tiebreaker set, the fail policy must not contain the `shutdown` option because there is no notification that a shutdown has occurred. See "Data Integrity Protection" on page 24.

SGI recommends that you start CXFS services on the tiebreaker client after the server-capable administration nodes are all up and running, and before CXFS services are started on any other clients. See "Restart the Cluster In an Orderly Fashion" on page 73.

Protect Data Integrity on All Nodes

All nodes must be configured to protect data integrity in case of failure. System reset or I/O fencing is required to ensure data integrity for all nodes.

Note: SGI recommends that you use a system reset configuration on server-capable administration nodes in order to protect data integrity and improve server reliability. I/O fencing (or system reset when available) must be used on client-only nodes.

See also "Data Integrity Protection" on page 24.

System Reset

You should configure system reset for all server-capable administration nodes in order to protect data integrity. (I/O fencing is appropriate for client-only nodes.)

Note: If the failure hierarchy contains `reset` or `fencereset`, the reset might be performed before the system kernel core-dump can complete, resulting in an incomplete core-dump.

System reset is recommended because if a server hangs, it must be rebooted as quickly as possbile to get it back in service, which is not available with I/O fencing. In addition, filesystem corruption is more likely to occur with a rogue metadata server, not a rogue client. (If fencing were to be used on a metadata server and fail, the cluster would have to either shutdown or hang. A fencing failure can occur if an administrator is logged into the switch.)

System reset may be either serial reset or, for a system with an L2 system controller or a baseboard management controller (BMC), over the network.

The system reset can use the following methods:

- `powerCycle` shuts off power to the node and then restarts it
- `reset` simulates the pressing of the reset button on the front of the machine
- `NMI` (nonmaskable interrupt) performs a core-dump of the operating system kernel, which may be useful when debugging a faulty machine

Note: NMI is not available on systems containing a BMC.

You would want to use `reset` for I/O protection on a client-only node that has a system controller when CXFS is a primary activity and you want to get it back online fast; for example, a CXFS fileserver.

I/O Fencing

Nodes without system reset capability require I/O fencing. I/O fencing is also appropriate for nodes with system controllers if they are client-only nodes.

You should use the `admin` account when configuring I/O fencing. On a Brocade switch running 4.*x.x.x* or later firmware, modify the `admin` account to restrict it to a single `telnet` session. For details, see the release notes.

If you use I/O fencing, SGI recommends that you use a switched network of at least 100baseT.

You should isolate the power supply for the switch from the power supply for a node and its system controller. You should avoid any possible situation in which a node can continue running while both the switch and the system controller lose power. Avoiding this situation will prevent the possibility of forming split clusters.

You must put switches used for I/O fencing on a network other than the primary CXFS private network so that problems on the CXFS private network can be dealt with by the fencing process and thereby avoid data or filesystem corruption issues. The network to which the switch is connected must be accessible by all server-capable administration nodes in the cluster.

If you manually change the port status, the CXFS database will not be informed and the status output by the `cxfs_admin` command will not be accurate. To update the CXFS database, run the following command:

```
admin# hafence -U
```

I/O fencing does the following:

- Preserves data integrity by preventing I/O from nodes that have been expelled from the cluster
- Speeds the recovery of the surviving cluster, which can continue immediately rather than waiting for an expelled node to reset under some circumstances

To support I/O fencing, platforms require a Fibre Channel switch; for supported switches, see the release notes.

When a node joins the CXFS kernel membership, the worldwide port name (WWPN) of its host bus adapter (HBA) is stored in the cluster database. If there are problems with the node, the I/O fencing software sends a message via the `telnet` protocol to the appropriate switch and disables the port.

Caution: You must keep the `telnet` port free in order for I/O fencing to succeed; **do not** perform a `telnet` to the switch and leave the session connected.

Brocade switches running 4.*x.x.x* or later firmware by default permit multiple `telnet` sessions. However, in the case of a split cluster, a server-capable administration node from each side of the network partition will attempt to issue the fence commands, but only the node that is able to log in will succeed. Therefore, on a Brocade switch running 4.*x.x.x* or later firmware, you must modify the `admin` account to restrict it to a single `telnet` session. See "Keep the `telnet` Port Open on the Switch" on page 76 and the release notes.

The switch then blocks the problem node from communicating with the storage area network (SAN) resources via the corresponding HBA. Figure 2-1 describes this.

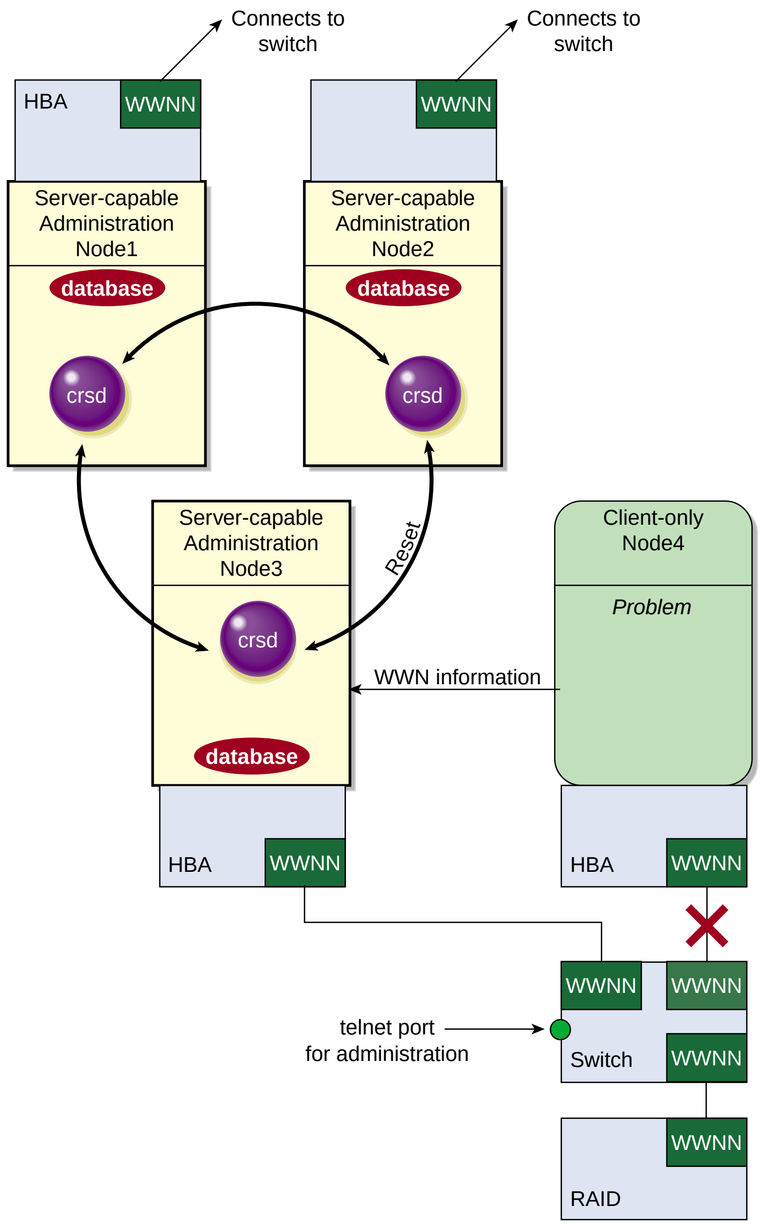


Figure 2-1 I/O Fencing

If users require access to nonclustered LUNs or devices in the SAN, these LUNs/devices must be accessed or mounted via an HBA that has been explicitly masked from fencing. For details on how to exclude HBAs from fencing for server-capable administration nodes, see:

- "Define a Switch with the GUI" on page 194
- "Create a Switch with `cxfs_admin`" on page 263

For nodes running other supported operating systems, see *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

To recover, the affected node withdraws from the CXFS kernel membership, unmounts all filesystems that are using an I/O path via fenced HBA(s), and then rejoins the cluster. This process is called *fencing recovery* and is initiated automatically. Depending on the fail policy that has been configured, a node may be reset (rebooted) before initiating fencing recovery. For information about setting the fail policy, see:

- "Fail Policies" on page 57
- "Define a Node with the GUI" on page 171
- "Create a Switch with `cxfs_admin`" on page 263

In order for a fenced node to rejoin the CXFS kernel membership, the current kernel membership leader must lower its fence, thereby allowing it to reprobe its XVM volumes and then remount its filesystems. If a node fails to rejoin the CXFS kernel membership, it may remain fenced. This is independent of whether the node was rebooted; fencing is an operation that is applied on the switch, not on the affected node. In certain cases, it may therefore be necessary to manually lower a fence. For instructions, see "Lower the I/O Fence for a Node with the GUI" on page 198, and "Switch Manipulation Using `hafence`" on page 286.

Caution: When a fence is raised on an HBA, **no further I/O is possible to the SAN via that HBA until the fence is lowered**. This includes the following:

- I/O that is queued in the kernel driver, on which user processes and applications may be blocked waiting for completion. These processes will return the `EIO` error code under UNIX, or display a warning dialog that I/O could not be completed under Windows.
- I/O issued via the affected HBAs to nonclustered (local) logical units (LUNs) in the SAN or to other Fibre Channel devices such tape storage devices.

On client-only nodes with system reset capability, you would want to use `Fence` for data integrity protection when CXFS is just a part of what the node is doing and therefore losing access to CXFS is preferable to having the system rebooted. An example of this would be a large compute server that is also a CXFS client. However, `Fence` cannot return a nonresponsive node to the cluster; this problem will require intervention from the system administrator.

For more information, see "Switches and I/O Fencing Tasks with the GUI" on page 194 , "Create or Modify a Node with `cxfs_admin`" on page 235, and "Switch Tasks with `cxfs_admin`" on page 262.

Shutdown

You should only use the `shutdown` fail policy for client-only nodes that use static CXFS kernel heartbeat.

If you use dynamic heartbeat monitoring, you must not use the `shutdown` fail policy for client-only nodes; it can be slower to recover because failure detection may take longer if no operations are pending against a node that fails. `shutdown` is not allowed as a fail policy because of the dynamic nature and potentially asymmetric heartbeat monitor between two nodes. For example, the server may begin monitoring heartbeat for a client, but that client may not currently be monitoring heartbeat of the server, and therefore the nodes may not discover they have lost membership in a timely manner.

In the case of a cluster with no tiebreaker node, it is possible that using the `shutdown` setting on server-capable administration nodes could cause a network partition in which split clusters could be formed and data could therefore be corrupted.

Suppose this configuration of server-capable administration nodes:

```
AdminNodeA      AdminNodeB
----------      -----
fence           fence
reset           reset
shutdown        shutdown
```

If the CXFS private network between `AdminNodeA` and `AdminNodeB` fails, the following could occur:

1. Each node will try to fence the other. (That is, `AdminNodeA` will try to fence `AdminNodeB`, and `AdminNodeB` will try to fence `AdminNodeA`).
2. If the fence fails, each node will try to reset the other.

3. If the system reset fails, each assumes that the other will shut itself down. Each will wait for a few moments and will then try to maintain the cluster.

   This will result in two clusters that are unaware of each other (a *split cluster*) and filesystem corruption will occur.

Suppose another configuration, in which neither server-capable administration node has `shutdown` set:

```
AdminNodeA      AdminNodeB
----------      ----------
fence           fence
reset           reset
```

If the CXFS private network between `AdminNodeA` and `AdminNodeB` fails in this situation, each node would first try to fence the other and then try to reset the other, as before. However, if both of those actions fail, each would assume that the state of the other node is unknown. Therefore, neither node would try to maintain the cluster. The cluster will go down, but no filesystem corruption will occur.

The split cluster problem may be avoided by using a tiebreaker node or by not using the `shutdown` setting on any server-capable administration node. You must not use `shutdown` if you use dynamic CXFS kernel heartbeat.

Avoid Split Clusters

The worst scenario is one in which the node does not detect the loss of communication but still allows access to the shared disks, leading to filesystem corruption. For example, it is possible that one node in the cluster could be unable to communicate with other nodes in the cluster (due to a software or hardware failure) but still be able to access shared disks, despite the fact that the cluster does not see this node as an active member.

In this case, the reset will allow one of the other nodes to forcibly prevent the failing node from accessing the disk at the instant the error is detected and prior to recovery from the node's departure from the cluster, ensuring no further activity from this node.

In a case of a split cluster, where an existing CXFS kernel membership splits into two halves (each with half the total number of server-capable administration nodes), the following will happen:

- If the CXFS tiebreaker and system reset or I/O fencing are configured, the half with the tiebreaker node will reset or fence the other half. The side without the tiebreaker will attempt to forcibly shut down CXFS services.
- If there is no CXFS tiebreaker node but system reset or I/O fencing is configured, each half will attempt to reset or fence the other half using a delay heuristic. One half will succeed and continue. The other will lose the reset/fence race and be rebooted/fenced.
- If there is no CXFS tiebreaker node and system reset or I/O fencing is not configured, then both halves will delay, each assuming that one will win the race and reset the other. Both halves will then continue running, because neither will have been reset or fenced, leading to likely filesystem corruption.

  To avoid this situation, you should configure a tiebreaker node, and you must use system reset or I/O fencing. However, if the tiebreaker node (in a cluster with only two server-capable administration nodes) fails, or if the administrator stops CXFS services, the other node will do a forced shutdown, which unmounts all CXFS filesystems.

If the network partition persists when the losing-half attempts to form a CXFS kernel membership, it will have only half the number of server-capable administration nodes and be unable to form an initial CXFS kernel membership, preventing two CXFS kernel memberships in a single cluster.

For more information, contact SGI professional or managed services.

Fail Policies

CXFS uses the following methods to isolate failed nodes. You can specify up to three methods by defining the fail policy. The second method will be completed only if the first method fails; the third method will be completed only if both the first and second methods fail. The possible methods are:

- `Fence`, which disables a node's Fibre Channel ports so that it cannot access I/O devices and therefore cannot corrupt data in the shared CXFS filesystem. When fencing is applied, the rest of the cluster can begin immediate recovery. See "I/O Fencing" on page 51.
- `Reset`, which performs a system reset via a system controller. See "System Reset" on page 50.
- `FenceReset`, which fences the node and then, if the node is successfully fenced, performs an asynchronous system reset; recovery begins without waiting for reset

acknowledgment. If used, this fail policy should be specified first. If the fencing action fails, the reset is not performed; therefore, `reset` alone is also required for all server-capable administration nodes (unless there is a single server-capable administration node in the cluster). See "I/O Fencing" on page 51 and "System Reset" on page 50.

- `Shutdown`, which tells the other nodes in the cluster to wait for a period of time (long enough for the node to shut itself down) before reforming the CXFS kernel membership. (However, there is no notification that the node's shutdown has actually taken place.) See "Shutdown" on page 55.

Caution: You must not use the `shutdown` setting in either of the following circumstances:

– If you have a cluster with no tiebreaker, you must not use the `shutdown` setting for any server-capable administration node in order to avoid split clusters being formed. (This is because there is no notification that a shutdown has occurred.) See "Shutdown" on page 55.
– On client nodes if you choose `dynamic` monitoring.

The following are valid fail policy sets:

Note: If the failure hierarchy contains `reset` or `fencereset`, the reset might be performed before the system kernel core-dump can complete, resulting in an incomplete core-dump.

- Server-capable administration nodes:

  `FenceReset, Reset` (Preferred)
  `FenceReset`
  `Reset`
  `Reset, Fence`
  (none, using the `cmgr` command)

- Client-only nodes with static CXFS kernel heartbeat monitoring:

  `Fence, Shutdown` (Preferred)
  `Fence`
  `Fence, Reset`
  `Fence, Reset, Shutdown`
  `FenceReset`

FenceReset, Reset
FenceReset, Reset, Shutdown
FenceReset, Shutdown
Reset
Reset, Fence
Reset, Fence, Shutdown
Reset, Shutdown
Shutdown
(none, using the `cmgr` command)

- Client-only nodes with dynamic CXFS kernel heartbeat monitoring:

  `Fence` (Most common)
  Fence, Reset
  FenceReset
  FenceReset, Reset
  Reset
  Reset, Fence
  (none, not recommended)

For more information, see "Use the Appropriate CXFS Kernel Heartbeat Monitoring" on page 63.

Note: If you choose no method, or if the fail policy does not include `Shutdown` and all of the other actions fail, CXFS will stall membership until the failed node either attempts to join the cluster again or until the administrator intervenes by using `cms_intervene`. Objects held by the failed node stall until membership finally transitions and initiates recovery. For more information, see the `cms_intervene`(1M) man page.

The rest of this section provides more details. See also "Protect Data Integrity on All Nodes" on page 50. For more information about setting the policies, see:

- "Define a Node with the GUI" on page 171
- "Create or Modify a Node with `cxfs_admin`" on page 235

Avoid CXFS Kernel Heartbeat Issues on Large SGI Altix Systems

To avoid CXFS kernel heartbeat issues on large SGI Altix systems (those with more than 64 processors), do the following:

- Keep current on maintenance levels (especially patches related to `xpmem`).
- Set the CXFS kernel heartbeat timeout to a large value cluster-wide. See "`mtcp_hb_period`" on page 481.
- Use cpusets (for more information, see *Linux Resource Administration Guide*
  - Bootcpuset:
    - On large systems, you generally need 4-8 CPUs in the bootcpuset.
    - The bootcpuset can consist of nonconsecutively numbered CPUs as long as each group is allocated on a node boundary.
    - Set `cpu_exclusive` (**but not** `memory_exclusive`).
  - Batch cpuset, used to schedule work (usually consists of the CPU and nodes that remain after defining the bootcpuset):
    - Set `cpu_exclusive` and `memory_exclusive`.
  - Per-job cpusets (children of the batch cpuset that are generally dynamically allocated by the batch scheduler) :
    - Allocated per-job cpusets on a node boundary.
    - Allocate per-job cpusets that are large enough to meet both the CPU and memory requirements of the job for which they are created. Jobs that exceed available resources should be killed via an automated means.
    - Set `cpu_exclusive` (**but not** `memory_exclusive`).
    - Set `memory_spread_{page,slab}` if specific nodes within a per-job cpuset are being oversubscribed.
- Remember the following when setting kernel memory parameters in `/etc/sysctl.conf`:
  - The kernel (`kswapd`) can lock out other activities on a CPU or the entire system if it is thrashing about, trying to maintain free memory pages to meet unrealistic default kernel tuning specifications on large systems.

– Do not oversubscribe memory on the system.

– Consider job's I/O requirements when estimating a job's memory; buffer cache comes from the same pool of memory.

For example, you could set the following

Note: The following tunable recommendations are generic. They should be evaluated to match a specific system's intended workload.

```
vm.min_free_kbytes= [ Number0fNodes*64*pagesize/1024]
                  # printf "%d\n64\n16\n**p" ‘ls /sys/devices/system/node/ | wc -l‘ | dc
            vm.dirty_ratio=10
            vm.dirty_background_ratio=5
```

- Pin interrupt processing for NICs used for CXFS private networks to CPUs in the bootcpuset.
- Configure the I/O subsystem to meet job requirements for throughput and responsiveness:
  – Maintain the I/O subsystem at peak efficiency. This includes CXFS filesystems as well as locally attached storage where job and system I/O occurs.
  – Maintain the `failover2.conf` file across the cluster to maximize I/O performance.
  – Flush dirty pages at the maximum possible speed. Uncontrolled growth in the number of dirty pages stresses the kernel's memory management functions (for example, `kswapd`) and increases the chance of lengthy kernel lockouts impacting CXFS heartbeat functionality.
- Set the `mtcp_hb_local_options` system tunable parameter to specify a heartbeat generation routine that avoids some memory allocation problems for nodes with large CPU counts that run massively parallel jobs (`0x3`). See "`mtcp_hb_local_options`" on page 481.
- Use dynamic heartbeat monitoring. A cluster defined with dynamic heartbeat starts monitoring only when a node is processing a message from another node (such as for token recall or XVM multicast) or when the client-only node monitors the server-capable administration node because it has a message pending (for example, a token acquire or metadata operation). Once monitoring initiates, it monitors at 1-second intervals and declares a timeout after a given number of

consecutive missed seconds (specified by `mtcp_hb_period`) , just like static monitoring. The intent of dynamic heartbeat monitoring is to avoid inappropriate loss of membership in clusters that have client-only nodes with heavy workloads. However, it may take longer to recover a client's tokens and other state information when there is an actual problem. Dynamic heartbeat monitoring also does not resolve any of the issues noted above if they occur during periods of active heartbeat monitoring.

See also "Inappropriate Node Membership Loss Due to CXFS Kernel Heartbeat Issues" on page 381.

Minimize the Number of Switches

CXFS fencing operations are more efficient with a smaller number of large switches rather than a large number of smaller switches.

Form a Small Functional Cluster First

Ensure that you follow the instructions in "Preliminary Cluster Configuration Steps" on page 128.

For large clusters, SGI recommends that you first form a functional cluster with just server-capable administration nodes and then build up the large cluster in small groups of client-only nodes. This method makes it easier to locate and fix problems, should any occur. See "Configuring a Large Cluster" on page 139.

Configure Filesystems Properly

Configure filesystems properly:

- Use a filesystem block size that is common to all CXFS OS platforms. Each CXFS OS platform supports a unique range of filesystem block sizes, but all of them support a filesystem block size of 4096 bytes. For this reason, SGI recommends 4-KB filesystems for compatibility with all CXFS platforms. For details on the filesystem block sizes supported by each CXFS OS platform, see the “Filesystem and Logical Unit Specifications” appendix in the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.
- Determine whether or not to have all filesystems served off of one metadata server or to use multiple metadata servers to balance the load, depending upon how

filesystems will be accessed. The more often a file is accessed, the greater the stress; a filesystem containing many small files that are accessed often causes greater stress than a filesystem with a few large files that are not accessed often. CXFS performs best when data I/O operations are greater than 16 KB and large files are being accessed. (A lot of activity on small files will result in slower performance.)

- Enable the forced unmount feature for CXFS filesystems, which is off by default. Many sites have found that enabling this feature improves the stability of their CXFS clusters, particularly in situations where the filesystem must be unmounted.

  This function is performed with the `fuser -m -k` command and the `umount` command. See:

  - "Unmount CXFS Filesystems with the GUI" on page 208
  - "Create or Modify a CXFS Filesystem with `cxfs_admin`" on page 254
  - "Unmount a CXFS Filesystem with `cxfs_admin`" on page 259

- If you are using NFS or Samba, you should have the NFS or Samba server run on the active metadata server.

- Do not use nested mount points. Although it is possible to mount other filesystems on top of a CXFS filesystem, this is not recommended.

- Perform reconfiguration (including but not limited to adding and deleting filesystems or nodes) during a scheduled cluster maintenance shift and not during production hours. You should stop CXFS services on a server-capable administration node before performing maintenance on it.

Use the Appropriate CXFS Kernel Heartbeat Monitoring

All nodes send CXFS kernel heartbeat messages once per second. If a node does not receive a heartbeat within a defined period, that node loses membership and is denied access to the cluster's filesystems. The defined period is one of the following:

- `static`: Monitors constantly at 1-second intervals and declares a timeout after 5 consecutive missed seconds (default).

- `dynamic`: Starts monitoring only when the node is processing a message from another node (such as for token recall or XVM multicast) or when the client monitors the server because it has a message pending (for example, a token acquire or metadata operation). After monitoring initiates, it monitors at 1-second intervals and declares a timeout after 5 consecutive missed seconds, just like static monitoring. Dynamic heartbeat monitoring is appropriate for clusters that have clients with heavy workloads; using it avoids inappropriate loss of membership. However, it may take longer to recover a client's tokens and other state information when there is an actual problem.

You can set the CXFS kernel heartbeat monitor period for the entire cluster by using the `cxfs_admin` command. See "Create or Modify a Cluster with `cxfs_admin`" on page 249.

Verify the Configuration

You should always run the following command after any significant configuration change or whenever problems occur in order to validate the configuration:

```
admin# /usr/cluster/bin/cxfs-config -xfs -xvm
```

The CXFS GUI and `cxfs_admin` do not always prevent poor configurations. The `status` command in `cxfs_admin` will indicate some potential problems and the `cxfs-config` tool can detect a large number of potential problems.

Use the Recovery Timeout Mechanism

The recovery timeout mechanism prevents the cluster from hanging and keeps filesystems available in the event that a node becomes unresponsive.

When recovery timeout is enabled, nodes are polled for progress after a recovery has begun. If recovery for a node is not making progress according to the specified polls, the recovery is considered stalled and the node will shut down or panic. For example, to enable the recovery timeout to begin monitoring after 5 minutes, monitor every 2 minutes, declare a node's recovery stalled after 15 minutes without progress, and panic the node with stalled recovery, you would set the following:

```
cxfs_recovery_timeout_start 300
cxfs_recovery_timeout_period 120
cxfs_recovery_timeout_stalled 900
cxfs_recovery_timeout_panic 1
```

For details about the parameters, see "Site-Changeable System Tunable Parameters" on page 471.

Use Proper Storage Management Procedures

You should configure storage management hardware and software according to its documentation and use proper storage mangement procedures, including the following:

- Assign IP addresses to all storage controllers and have them network-connected (but not on the private CXFS metadata network) and manageable via out-of-band management

  Note: Do not use in-band management (which can cause problems if there is a loss of Fibre Channel connectivity).

- Keep a copy of the array configuration
- Monitor for read errors that do not result in drive strikes
- Keep a copy of the XVM volume configuration

Samba and CXFS

If you are using Samba, you should have the Samba server run on the active metadata server. You should not use multiple Samba servers to export the same CXFS filesystem. For more information, see "Samba" on page 31.

DMF and CXFS

Install the DMF server subsystem only on CXFS server-capable administration nodes. Install the DMF client subsystem on CXFS client-only nodes. For more information about DMF, see *DMF Administrator's Guide for SGI InfiniteStorage*.

Administration Best Practices

This section discusses the following administration topics:

- "Do Not Run User Jobs on Server-Capable Administration Nodes" on page 67
- "Do Not Run Backups on a Client Node" on page 67
- "Use `cron` Jobs Properly" on page 67

Do Not Run User Jobs on Server-Capable Administration Nodes

Do not run user jobs on any server-capable administration nodes. These systems must be dedicated to CXFS services for maximum stability. See "Use Server-Capable Administration Nodes that are Dedicated to CXFS Work" on page 48.

Do Not Run Backups on a Client Node

SGI recommends that you perform backups on the active metadata server.

Do not run backups on a client node, because it causes heavy use of non-swappable kernel memory on the metadata server. During a backup, every inode on the filesystem is visited; if done from a client, it imposes a huge load on the metadata server. The metadata server may experience typical out-of-memory symptoms, and in the worst case can even become unresponsive or crash.

Use `cron` Jobs Properly

Jobs scheduled with `cron` can cause severe stress on a CXFS filesystem if multiple nodes in a cluster start the same filesystem-intensive task simultaneously.

Because CXFS filesystems are considered as local on all nodes in the cluster, the nodes may generate excessive filesystem activity if they try to access the same filesystems simultaneously while running commands such as `find` or `ls`. You should build databases for `rfind` and GNU `locate` only on the active metadata server.

Any task initiated using `cron` on a CXFS filesystem should be launched from a single node in the cluster, preferably from the active metadata server. Edit the nodes' `crontab` file to only execute the `find` command on one metadata server of the cluster.

Repair Filesystems with Care

Always contact SGI technical support before using `xfs_repair` on CXFS filesystems. You must first ensure that you have an actual case of filesystem corruption and retain valuable metadata information by replaying the XFS logs before running `xfs_repair`.

⚠ **Caution:** If you run `xfs_repair` without first replaying the XFS logs, you may introduce data corruption. You should run `xfs_ncheck` and capture the output to a file before running `xfs_repair`. If running `xfs_repair` results in files being placed in the `lost+found` directory, the saved output from `xfs_ncheck` may help you to identify the original names of the files.

Only use `xfs_repair` on server-capable administration nodes and only when you have verified that all other cluster nodes have unmounted the filesystem. Make sure that `xfs_repair` is run only on a cleanly unmounted filesystem. If your filesystem has not been cleanly unmounted, there will be uncommitted metadata transactions in the log, which `xfs_repair` will erase. This usually causes loss of some data and messages from `xfs_repair` that make the filesystem appear to be corrupted.

If you are running `xfs_repair` right after a system crash or a filesystem shutdown, your filesystem is likely to have a dirty log. To avoid data loss, you **MUST** mount and unmount the filesystem before running `xfs_repair`. It does not hurt anything to mount and unmount the filesystem locally, after CXFS has unmounted it, before `xfs_repair` is run.

Defragment Filesystems with Care

The `xfs_fsr` tool is useful when defragmenting specific files but not filesystems in general.

Using `xfs_fsr` to defragment CXFS filesystems is not recommended except on read-mostly filesystems because `xfs_fsr` badly fragments the free space. XFS actually does best at maintaining contiguous free space and keeping files from being fragmented if `xfs_fsr` is not run as long as there is a moderate (10% or more) free space available on the filesystem.

⚠ **Caution:** You must use the `xfs_fsr` command **only** on the active metadata server for the filesystem; the `bulkstat` system call has been disabled for CXFS clients. You should use `xfs_fsr` manually, and only on the active metadata server for the filesystem.

Use Relocation and Recovery Properly

Use relocation and recovery only to a standby node that does not currently run any applications (including NFS and Samba) that will use that filesystem. The node can run applications that use other filesystems. See "Node Types, Node Functions, and the Cluster Database" on page 12.

Shut Down Nodes Unobtrusively

Use the proper procedures for shutting down nodes. See "Cluster Member Removal and Restoration" on page 307.

When shutting down, resetting, or restarting a CXFS client-only node, do not stop CXFS services on the node. Rather, let the CXFS shutdown scripts on the node stop CXFS when the client-only node is shut down or restarted. (Stopping CXFS services is more intrusive on other nodes in the cluster because it updates the cluster database. Stopping CXFS services is appropriate only for a server-capable administration node.)

If you are going to perform maintenance on a potential metadata server, you should first shut down CXFS services on it. Disabled nodes are not used in CXFS kernel membership calculations, so this action may prevent a loss of quorum.

Remove Unused Cluster Components

As long as a server-capable administration node remains configured in the cluster database, it counts against cluster database quorum. However, the way it impacts the cluster depends upon the actual node count.

If a server-capable administration node is expected to be down for longer than the remaining mean-time to failure (MTTF) of another server-capable administration node in the cluster, you should remove it from the cluster and the pool in order to avoid problems with cluster database membership and CXFS membership quorum. See the following sections:

- "Modify a Cluster Definition with the GUI" on page 187
- "Delete a Node with `cxfs_admin`" on page 244

You should leave a client-only node in the cluster database unless you are permanently removing it.

You should also remove the definitions for unused objects such as filesystems and switches from the cluster database. This will improve the cluster database performance and reduce the likelihood of cluster database problems.

Use `fam` Properly

If you want to use the file alteration monitor (`fam`), you must remove the `/dev/imon` file from CXFS nodes. Removing this file forces `fam` to poll the filesystem. For more information about the monitor, see the `fam`(3) man page.

Upgrade the Software Properly

Do the following when upgrading the software:

- Save the current CXFS configuration as a precaution before you start an upgrade and acquire new CXFS server-side licenses (if required). See Chapter 13, "Cluster Database Management" on page 331, and Chapter 5, "CXFS License Keys" on page 89.
- Read the release notes and any late-breaking caveats on Supportfolio before installing and/or upgrading CXFS. These contain important information needed for a stable install/upgrade.
- Do not make any other configuration changes to the cluster (such as adding new nodes or filesystems) until the upgrade of all nodes is complete and the cluster is running normally.

SGI recommends the following for server-capable administration nodes in a production cluster:

- Run the latest CXFS release.
- Run a release that is the same or later than the release run by client-only nodes. (The only exception is if the release in question does not apply to the server-capable administration nodes.)
- Run the same minor-level release (such as 4.0.3) on all server-capable administration nodes.

Use Fast Copying for Large CXFS Files

You can use the `cxfscp` command to quickly copy large files (64 KB or larger) to and from a CXFS filesystem. It can be significantly faster than the `cp` command on CXFS filesystems because it uses multiple threads and large I/Os to fully use the bandwidth to the storage hardware.

Files smaller than 64 KB do not benefit from large direct I/Os. For these files, `cxfscp` uses a separate thread using buffered I/O, similar to `cp`.

The `cxfscp` command is available on on IRIX, SGI ProPack, Linux, and Windows platforms. However, some options are platform-specific, and other limitations apply.

By default, `cxfscp` uses direct I/O. To use buffered I/O, use the `--bi` and `--bo` options; for more information and a complete list of options, see the `cxfscp`(1) man page.

Do Not Change Log File Names

You should not change the names of the log files. If you change the names of the log files, errors can occur. If the disk is filling with log messages, see "Log File Management" on page 302.

Rotate Log Files

Periodically, you should rotate log files to avoid filling your disk space; see "Log File Management" on page 302. If you are having problems with disk space, you may want to choose a less verbose log level; see "Configure Log Groups with the GUI" on page 192.

Use System Capacity Wisely

To avoid a loss of connectivity between the metadata server and the CXFS clients, do not oversubscribe the metadata server or the private network connecting the nodes in the cluster. Avoid unnecessary metadata traffic.

If the amount of free memory is insufficient, a node may experience delays in CXFS kernel heartbeat and as a result will be kicked out of the CXFS membership. Examine the `/proc` filesystem for more information and use the commands found in the Linux `procps` RPM to monitor memory usage, in particular:

```
free
slabtop
vmstat
```

See also:

- "Provide Enough Memory" on page 45
- "Use Server-Capable Administration Nodes that are Dedicated to CXFS Work" on page 48
- "Out of Logical Swap Space" on page 387

Reboot Before Changing Node ID or Cluster ID

If you want to change a node ID or the cluster ID, do the following:

1. Remove the current cluster definition for the node and/or cluster.
2. Reboot the node in order to clear the original values for node ID and cluster ID. (If you do not reboot before redefining, the kernel will still have the old values, which prohibits a CXFS membership from forming.) To change the cluster ID, you must reboot all nodes in the cluster.
3. Redefine the node ID or cluster ID

If you use `cdbreinit` on a server-capable administration node to recreate the cluster database, you must reboot it before changing the node IDs or the cluster ID.

See "Recreating the Cluster Database" on page 413.

Restart CXFS on a Node after an Administrative CXFS Stop

If you perform an administrative CXFS stop (forced CXFS shutdown) on a node, you must perform an administrative CXFS start on that node before it can return to the cluster. If you do this while the database still shows that the node is in the cluster and is activated, the node will restart the CXFS membership daemon. Following a forced CXFS shutdown, the node can be prevented from restarting the CXFS membership daemon when CXFS is restarted by stopping CXFS services. (A forced CXFS shutdown alone does not stop CXFS services. A forced CXFS shutdown stops only the kernel membership daemon. Stopping CXFS services disables the node in the cluster database.)

For example, enter the following `cxfs_admin` command on the local node you wish to start:

```
cxfs_admin:mycluster> start_cxfs
```

See:

- "Disable a Node with `cxfs_admin`" on page 244
- "Enable a Node with `cxfs_admin`" on page 244
- "Revoke and Restore CXFS Kernel Membership for the Local Node" on page 248
- "Forced CXFS Shutdown: Revoke Membership of Local Node" on page 299

Restart the Cluster In an Orderly Fashion

SGI recommends that you do the following to restart the cluster in an orderly fashion if you have previously taken the entire cluster down for maintenance or because of server instability. This procedure assumes all nodes have been disabled.

1. Restart CXFS services (using the CXFS GUI or `cxfs_admin`) for the potential metadata servers. Do the following for each potential metadata server if you are using the `cxfs_admin` command:

   ```
   cxfs_admin:clustername> enable node:admin1_nodename
   cxfs_admin:clustername> enable node:admin2_nodename
   ...
   ```

2. Restart CXFS services on the client-only tiebreaker node:

   ```
   cxfs_admin:clustername> enable node:tiebreaker_nodename
   ```

3. Restart CXFS services on the remaining client-only nodes: =

   ```
   cxfs_admin:clustername> enable node:client1_nodename
   cxfs_admin:clustername> enable node:client2_nodename
   ...
   ```

   Repeat this step for each client-only node.

Disable Reset Capability If You Remove Reset Lines

When reset is enabled, CXFS requires a `reset successful` message before it moves the metadata server. Therefore, if you have the reset capability enabled when you must remove the reset lines for some reason, you must first disable the reset capability before removing the reset lines. See the following:

- "Modify a Node Definition with the GUI" on page 181
- "Create or Modify a Node with `cxfs_admin`" on page 235

Avoid Performance Problems with Unwritten Extent Tracking

When you define a filesystem, you can specify whether unwritten extent tracking is on (`unwritten=1`) or off (`unwritten=0`); it is on by default.

In most cases, the use of unwritten extent tracking does not affect performance and you should use the default to provide better security. However, unwritten extent tracking can affect performance when **both** of the following are true:

- A file has been preallocated
- Preallocated extents are written for the first time with records smaller than 4 MB

For optimal performance with CXFS when **both** of these conditions are true, it may be necessary to build filesystems with `unwritten=0` (off).

Caution: There are security issues with using `unwritten=0`.

Avoid Performance Problems with Exclusive Write Tokens

For proper performance, CXFS should not obtain exclusive write tokens. Therefore, use the following guidelines:

- Preallocate the file.
- Set the size of the file to the maximum size and do not allow it to be changed, such as through truncation.
- Do not append to the file. (That is, `O_APPEND` is not true on the open.)

- Do not mark an extent as `written`.
- Do not allow the application to do continual `preallocation` calls.

Use the Appropriate Version of `lcrash` for SGI ProPack

If you want to use `lcrash` for troubleshooting on an SGI ProPack node, you must use the version of `lcrash` that is available from Supportfolio. Use the `-x` option to load the CXFS `kerntypes`:

```
# lcrash -x /boot/sgi-cxfs-kerntypes-kernelversion-architecturetype
```

Note: Do not use the version of `lcrash` that is shipped with SLES 9.

Use Disk Layout Optimization for Approved Media Customers

Approved media customers can use the XFS `filestreams` mount option with CXFS to maximize the ability of storage to support multiple real-time streams of video data. It is appropriate for workloads that generate many files that are created and accessed in a sequential order in one directory.

Caution: SGI must validate that your RAID model and RAID configuration can support the use of the `filestreams` mount option to achieve real-time data transfer and that your application is appropriate for its use. Use of this feature is complex and is reserved for designs that have been approved by SGI.

For more information, see "Disk Layout Optimization for Approved Media Customers" on page 325.

Set System Tunable Parameters Appropriately

You should use care when changing system tunable parameters. Do not change restricted system tunable parameters unless directed to do so by SGI support. For details, see Appendix F, "System Tunable Parameters" on page 471.

There are several configuration files that you can use to set a tunable automatically. SGI recommends that you use the `/etc/modprobe.conf.local` file. This file specifies options for modules and can be used to set options that cannot be set with `sysctl`. To set an option, add a line of the following format to `/etc/modprobe.conf.local`:

```
options  modulename  tunablename=value
```

In this guide, the *modulename* value to be used is given in the "Location" entry in Appendix F, "System Tunable Parameters" on page 471. For example, `sgi-cxfs` is the module name for the tunable `rhelpd_max`. Therefore, to set the value of `rhelpd_max` to `128`, you would add the following line to `/etc/modprobe.conf.local`:

```
options sgi-cxfs rhelpd_max=128
```

There should be only one `options` line per module; if you want to specify multiple parameters, you must place them all on that single line.

Note: SGI does not recommend using `/etc/sysctl.conf` because it is a global configuration file that might be affected by upgrades of non-related software.

Keep the `telnet` Port Open on the Switch

If there are problems with a node, the I/O fencing software sends a message via the `telnet` protocol to the appropriate Fibre Channel switch. The switch only allows one `telnet` session at a time; therefore, if you are using I/O fencing, you must keep the `telnet` port on the Fibre Channel switch free at all times.

Caution: Do not perform a `telnet` to the switch and leave the session connected.

Solve Problems Efficiently

To solve problems efficiently, understand the information in "Troubleshooting Strategy" on page 353 and gather the information discussed in "Reporting Problems to SGI" on page 416.

Chapter 3

SGI RAID for CXFS Clusters

This chapter discusses SGI RAID for CXFS clusters:

- "RAID Hardware" on page 77
- "RAID Firmware" on page 78
- "Number of LUNs Supported" on page 79
- "RAID Verification" on page 80

For additional updates, see the CXFS release notes.

RAID Hardware

CXFS supports the following RAID hardware:

SGI InfiniteStorage 10000
SGI InfiniteStorage 6700
SGI InfiniteStorage 4500
SGI InfiniteStorage 4000
SGI InfiniteStorage 220 (Fibre Channel)
SGI RM610
SGI RM660
SGI TP9700
SGI TP9500S (serial ATA)
SGI TP9500
SGI TP9400
SGI TP9300S (serial ATA)
SGI TP9300
SGI TP9100
SGI S330

The SGI RAID will be initially installed and configured by SGI personnel.

RAID Firmware

SGI RAID supports the following firmware:

Note: SGI InfiniteStorage 220 does not support online updates of the controller firmware.

- SGI RM610 and RM660 running version 5.12b or later.
- SGI InfiniteStorage 6700 supports controller firmware version V3.00.
- The TP9700 9.14 CD contains the required controller firmware and NVSRAM files. The 06.14.*xx.xx* code or later must be installed.
- The TP9500S 8.0 CD contains the required controller firmware and NVSRAM files. The 05.41.*xx.xx* code or later must be installed.
- The TP9400/TP9500 6.0 CD contains the required controller firmware and NVSRAM files. The 05.30.*xx.xx* code or later must be installed.
- The TP9400 4.0 CD contains the required controller firmware and NVSRAM files for the 4774 or 4884 units:
  - If you have a 4774 unit, the 04.01.*xx.xx* , 04.02.*xx.xx*, or 05.30.*xx.xx* code or later must be installed
  - If you have a 4884 unit, the 04.02.*xx.xx* code is installed by default
- The TP9300S 8.0 CD contains the required controller firmware and NVSRAM files. The 05.41.*xx.xx* code or later must be installed if using 2882 controllers, or 05.42.*xx.xx* code or later if using 2822 controllers.

  Note: The initial TP9300S used 2882 controllers in the controller module. This product was later replaced with a 2822 controllers (still using the TP9300S marketing code). With the release of the 2822 controller, SATA disk drives can be installed in the controller module (the 2882 did not have disk drives installed in the controller module).

- The TP9300 7.0 CD contains the required controller firmware and NVSRAM files. The 05.33.*xx.xx* code or later must be installed.

- The TP9100 4.0 CD contains the required version 7.75 controller firmware for the 1-Gbit TP9100. Supported via special request with optical attach (other conditions may apply).
- The TP9100 5.0 CD contains the required version 8.40 firmware for the 2-Gbit TP9100.

  Note: The TP9100 is limited to 64 host connections.

- The TP9300 8.42 CD (TPSSM 8.42) contains the required 8.42 firmware for the S330.

See also "XVM Failover Version 2 Overview" on page 33.

Number of LUNs Supported

By default, the RAID firmware supports a maximum number of logical units (LUNs). If additional LUNs are required, you must obtain a separate software-enabling key; this key will support a larger number of LUNs in separate partitions, which requires that the Fibre Channel ports be mapped to a partition. Contact your SGI sales representative for the SGI software partitioning key.

The maximum depends upon the code installed, as shown in Table 3-1.

Table 3-1 Number of LUNs Supported

Firmware Level	Default LUN Maximum	LUN Maximum with a Partioning Key
04.01.*xx.xx*	32	128
04.02.*xx.xx*	32	128
05.30.*xx.xx*	32	1024
05.33.xx.xx	32	2048
05.40.*xx.xx*	256	2048
06.14.*xx.xx*	32	2048

RAID Verification

To verify that the SGI RAID is properly installed and ready for use with CXFS, you can dump the RAID’s profile and verify the controller software revisions.

Chapter 4

Switches

This chapter discusses the following:

- "Brocade Switch" on page 81
- "QLogic Fibre Channel Switch" on page 86

Brocade Switch

This section discusses the following:

- "Brocade Firmware" on page 81
- "Verifying the Brocade Switch Firmware Version" on page 83
- "Verifying the Brocade License" on page 83
- "Limiting `telnet` Sessions" on page 84
- "Changing the Brocade FC Cable Connections" on page 85

Brocade Firmware

All Brocade switches contained within the SAN fabric must have the appropriate Brocade firmware, shown in Table 4-1.

Note: There are issues when upgrading from firmware v4.1.1. See Technical Information Bulletin 201240 on Supportfolio for details:

http://support.sgi.com

Table 4-1 Brocade Firmware

Switch	Ports	Speed (Gb/s)	Minimum Firmware
200E	8, 16	4	5.2.2
2400	8	1	2.6.2d
2800	16	1	2.6.2d
3200	8	2	3.2.1c
3250	8	2	5.2.2
3252	8	2	5.2.2
3800	16	2	3.2.1c
3850	16	2	5.2.2
3852	16	2	5.2.2
3900	32	2	5.2.2
4100	32	4	5.2,2
4900	16, 32, 64	4	5.2.2
5000	16, 32, 64	4	5.2.2
12000	32, 64, dual 64	2	5.0.5d
24000	32, 64, 128	2	5.2.2
48000	32 through 256	4	5.2.2

If the current firmware level of the switches must be upgraded, please contact your local SGI service representative or customer support center.

The Brocade switch must be configured so that its Ethernet interface is accessible (using `telnet`) from all CXFS administration nodes. The fencing network connected to the Brocade switch must be physically separate from the private heartbeat network.

Caution: The `admin` state must be free in order for I/O fencing to succeed.

Switches using 4.*x.x.x* or later firmware permit multiple `telnet` sessions. However, CXFS I/O fencing requires a `telnet` lockout for global mutual exclusion when a fencing race occurs. Therefore, you must configure these switches to set the maximum allowed simultaneous `telnet` sessions for the `admin` user to 1. (Brocade switches running 3.*x.x.x* firmware are shipped with the required restrictions configured by default).

Verifying the Brocade Switch Firmware Version

To verify the firmware version, log into the switch as user admin and use the version command, as shown in the following example:

```
workstation% telnet brocade1
Trying 169.238.221.224...
Connected to brocade1.example.com
Escape character is '^]'.

Fabric OS (tm)  Release v2.6.0d

login: admin
Password:
brocade1:admin> version
Kernel:     5.4
Fabric OS:  v2.6.0d            <== Firmware Revision
Made on:    Fri May 17 16:33:09 PDT 2002
Flash:      Fri May 17 16:34:55 PDT 2002
BootProm:   Thu Jun 17 15:20:39 PDT 1999
brocade1:admin>
```

Verifying the Brocade License

To verify the Brocade license, log into the switch as user admin and use the licenseshow command, as shown in the following example:

```
brocade:admin> licenseshow
dcRyzyScSedSz0p:
    Web license
    Zoning license
    SES license
    Fabric license
SQQQSyddQ9TRRdUP:
    Release v2.2 license
```

Limiting `telnet` Sessions

You must limit the maximum allowed simultaneous `telnet` session

Brocade 200E/3250/3252/3850/3852/3900/4100/4900/5000 and `telnet`

To limit the maximum allowed simultaneous `telnet` sessions for the `admin` user to 1 on the Brocade 200E/3250/3252/3850/3852/3900/4100/4900/5000, do the following:

1. Connect to the switch via the `telnet` command and log in as `root`.
2. Issue the sync command to avoid filesystem corruption:

   ```
   # sync
   ```

3. Edit the `/etc/profile` file to change the `max_telnet_sessions` from `2` to `1` and place the information in a new file. For example:

```
# cd /etc
# sed -e 's/max_telnet_sessions=2/max_telnet_sessions=1/' profile >profile.new
```

4. Distribute the edited profile file to both partitions on both central processors. For example:

   ```
   # cp profile.new profile
   # cp profile.new /mnt/etc/profile
   ```

5. Issue the `sync` command again to avoid filesystem corruption:

   ```
   # sync
   ```

Brocade 12000/24000/48000 and `telnet`

To limit the maximum allowed simultaneous telnet sessions for the `admin` user to `1` on the Brocade 12000/24000/48000, do the following:

1. Connect to the switch via the `telnet` command and log in as `root`.
2. Use the `haShow` command to make sure that both central processors are up. This is indicated by the message Heartbeat Up within the output of the `haShow` command. If it is not up, wait a few minutes and run haShow again to check for the status.

3. Issue the `sync` command on the filesystems to avoid filesystem corruption:

   ```
   # rsh 10.0.0.5 sync
   # rsh 10.0.0.6 sync
   ```

4. Edit the `/etc/profile` file to change the `max_telnet_sessions` from 2 to 1 and place the information in a new file. For example:

```
# cd /etc
# sed -e 's/max_telnet_sessions=2/max_telnet_sessions=1/' profile >profile.new
```

5. Distribute the new profile to both partitions and central processors. For example:

   ```
   # rcp /etc/profile.new 10.0.0.5:/etc/profile
   # rcp /etc/profile.new 10.0.0.5:/mnt/etc/profile
   # rcp /etc/profile.new 10.0.0.6:/etc/profile
   # rcp /etc/profile.new 10.0.0.6:/mnt/etc/profile
   ```

6. Issue the `sync` command again to avoid filesystem corruption:

   ```
   # rsh 10.0.0.5 sync
   # rsh 10.0.0.6 sync
   ```

Changing the Brocade FC Cable Connections

To change Brocade Fibre Channel cable connections used by nodes in the CXFS cluster, do the following:

1. Cleanly shut down CXFS services on the nodes affected by the cable change. Use the CXFS GUI or `cxfs_admin`.
2. Rearrange the cables as required.
3. Restart CXFS services.
4. Reconfigure I/O fencing if required. You must perform this step if I/O fencing is enabled on the cluster and if you added/removed any Brocade switches. You must use the CXFS GUI or `cxfs_admin` to add or remove switches from the CXFS configuration as required.
5. If any CXFS client nodes are connected to a new (or different) Brocade switch, restart CXFS services on those nodes. This will ensure that the CXFS administration servers can correctly identify the Brocade ports used by all clients.

QLogic Fibre Channel Switch

All QLogic Fibre Channel (FC) switches contained within the SAN fabric must have the appropriate QLogic firmware installed, as shown in Table 4-2.

Table 4-2 QLogic FC Switch Firmware

QLogic FC Switch Model	SANbox Name	Minimum Firmware
SB2A-16A/B	2-16	4.0
SB2B-08A/B	2-8	4.0
SB2C-16BSE	2-64	4.0
SB5200-08/12/16/20A	5200	V5.0.1.10.0
SB9200-32B	9200	V6.2.0.8.0

For more information, see the QLogic *SANbox2-64 Switch Management User's Guide*.

Caution: The `admin` state is required for I/O fencing. To avoid interference with fencing, release `admin` mode as soon as possible. Do not leave `admin` mode sessions open.

The default port configuration on a QLogic 9200 FC switch is not compatible with the CXFS environment. To use the appropriate port configuration, change the following parameters:

`LinkSpeed`	Set to the appropriate value, such as `2` for 2 GB/s. (In some cases, `Auto` does not function properly.)
`PortType`	Enter the appropriate type, usually `F`. (You cannot use the `GL` autonegotiated mode.)
`NoClose`	Set to `True` to prevent the Fibre Channel circuit from shutting down during a host reboot.

`IOStreamGuard`	Set to `Enable` if the port is connected to a host HBA or to `Disable` if the port is connected to a storage HBA. (You cannot use `Auto` mode because most HBAs cannot negotiate this.)

To modify these parameters, use the `admin` command. For example, for a port connected to an SGI Altix:

```
SANbox #> admin start

SANbox (admin) #> config edit
  The config named default is being edited.

SANbox (admin-config) #> set config port 31


  A list of attributes with formatting and current values will follow.
  Enter a new value or simply press the ENTER key to accept the current value.
  If you wish to terminate this process before reaching the end of the list
  press 'q' or 'Q' and the ENTER key to do so.

  Configuring Port Number: 31
  ------------------------

  AdminState       (1=Online, 2=Offline, 3=Diagnostics, 4=Down) [Online]
  LinkSpeed        (1=1Gb/s, 2=2Gb/s, 4=4Gb/s, A=Auto)          [Auto  ]  2
  PortType         (GL / G / F / FL / Donor)                    [GL    ]  F
  SymPortName      (string, max=32 chars)                       [Port31]  Altix45
  ALFairness       (True / False)                               [False ]
  DeviceScanEnable (True / False)                               [True  ]
  ForceOfflineRSCN (True / False)                               [False ]
  ARB_FF           (True / False)                               [False ]
  InteropCredit    (decimal value, 0-255)                       [0     ]
  ExtCredit        (dec value, increments of 15, non-loop only) [0     ]
  FANEnable        (True / False)                               [True  ]
  AutoPerfTuning   (True / False)                               [True  ]
  MSEnable         (True / False)                               [True  ]
  NoClose          (True / False)                               [False ]  True
  IOStreamGuard    (Enable / Disable / Auto)                    [Auto  ]  Enable
  PDISCPingEnable  (True / False)                               [True  ]

  Finished configuring attributes.
  This configuration must be saved (see config save command) and
  activated (see config activate command) before it can take effect.
  To discard this configuration use the config cancel command.
  ....

SANbox (admin-config) #> **config save**
  The config named default has been saved.

SANbox (admin) #> **config activate**

  The currently active configuration will be activated.
  Please confirm (y/n): [n] **y**

SANbox (admin) #> **admin end**

SANbox #> **show config port 31**

  Configuration Name: default
  -------------------

  Port Number: 31
  ------------
  AdminState         Online
  LinkSpeed          2Gb/s
  PortType           F
  SymbolicName       Altix45
  ALFairness         False
  DeviceScanEnabled  True
  ForceOfflineRSCN   False
  ARB_FF             False
  InteropCredit      0
  ExtCredit          0
  FANEnabled         True
  AutoPerfTuning     True
  MSEnabled          True
  NoClose            True
  IOStreamGuard      Enabled
  PDISCPingEnabled   True
```

Chapter 5

CXFS License Keys

Note: On SGI Altix and SGI Altix XE hardware platforms running SGI ProPack 5.0 or later, *CPU count* is the number of processor sockets. On all other hardware platforms, or SGI Altix XE hardware running any other operating system, *CPU count* is the number of processor cores.

The licensing used for SGI ProPack server-capable administration nodes is based on the SGI License Key (LK) software. Only server-side licensing is supported.

For the purposes of licensing, hyperthreaded CPUs are counted as a single processor, while multicore processors are counted as multiple processors. Therefore, a dual-core processor will be counted as 2 CPUs for the purposes of licensing with CXFS. A hyperthreaded CPU or hyperthreaded core would only be counted as a single CPU for licensing purposes.

This section discusses the following:

- "Server-Side Licensing Overview" on page 89
- "Gathering the Host Information" on page 95
- "Obtaining the License Keys from SGI" on page 95
- "Installing the License Keys" on page 95
- "Verifying the License Keys" on page 96
- "For More Information About Licensing" on page 101

See also "Upgrading Licenses from 4.*X* to 5.0" on page 278.

Server-Side Licensing Overview

CXFS 5.0 requires server-side licensing and CXFS 5.0 licenses. These are available to all customers with current support contracts; contact your SGI support person.

CXFS server-side licensing uses license keys on the CXFS server-capable administration nodes; it does not require node-locked license keys on CXFS client-only nodes. The license keys are node-locked to each server-capable administration node and specify the number and size of client-only nodes that may join the cluster membership.

Server-side licensing provides flexibility when changing the CXFS cluster configuration, such as the following: adding nodes, changing the number of CPUs in one host, or using a license key part-time from different nodes connected to the cluster.

Licensing Requirements

Server-side licensing requires the following license keys on each server-capable administration node:

- `CXFS_SS` feature license key. The server license key specifies the maximum number of CPUs on the server. This license key is node-locked to the server.
- Client license keys, which specify the number and/or size of client-only nodes that may join the cluster. See "Server-Side Client License Keys" on page 90.

No license keys are required on the client-only nodes themselves.

Note: Other CXFS-aware products also require license keys:

- XVM cluster mirroring requires a license key on server-capable administration nodes in order for cluster nodes to access the cluster mirror. On CXFS client-only nodes, the user feature where applicable is honored after the `cxfs_client` service is started. XVM cluster mirroring on clients is also honored if it is enabled on the server. All CXFS client nodes need an appropriate mirror license key in order to access local mirrors.
- Guaranteed rate I/O version 2 (GRIOv2) requires a license key on the server-capable administration nodes

---

Server-Side Client License Keys

There are two classes of server-side client license keys:

- *Workstation client license keys* specify the number of nodes with as many as 16 CPUs running one of the following platforms:

  Mac OS X
  Windows

For example, an 8-node workstation client license key will allow up to eight nodes running any combination of the supported workstation platforms to join CXFS membership. On Monday, you could have eight Windows 16-CPU nodes, on Tuesday you could have four Mac OS X nodes and four Windows 16-CPU nodes.

- *Enterprise client license keys* specify the total number of CPUs running one of the following platforms:

  AIX
  IRIX
  Linux on x86_64 or ia64 architecture
  SGI ProPack 5 (CPU count is the number of sockets, not cores)
  Solaris
  Windows (more than 16 CPUs)

  For example, a 32–CPU enterprise license key will allow sixteen 2-CPU nodes, eight 4–CPU nodes, or one 32–CPU node to join membership. If your cluster contained an SGI ProPack node with 4 sockets (4 dual-core CPUs), it would use 4 of the licenses.

License Key Replication on Server Nodes

The purchase of a workstation or enterprise license entitles you to generate a license key on each server-capable administration node in the cluster. Every server-capable administration node in the cluster should install a set of client license keys. A server will generate warnings in the system log if the license keys on one server-capable administration node are not equivalent to other server-capable administration nodes in the cluster.

Note: Server–side licensing does not introduce a point-of-failure in the CXFS cluster. If the metadata server fails and the cluster recovers to a backup server that has fewer/smaller client license keys, the client-only nodes that are currently in the cluster membership will remain in the membership. However, additional client–only nodes that attempt to join membership will fail until the membership count is reduced to below the license key entitlement on the active metadata server.

Cumulative Client License Keys

The number of client license keys is cumulative. To add more client-only nodes, you can purchase additional workstation or enterprise licenses as appropriate (you do not have to upgrade existing license keys).

For example, if you already have a 32-CPU enterprise license key and want to add another 32-CPU enterprise-class machine, you purchase another 32-CPU enterprise license. You must install this new license key key on every server-capable administration node in the cluster.

Examples of License Keys Required for Cluster Changes

The following figures show examples of the license keys that are required for cluster changes.

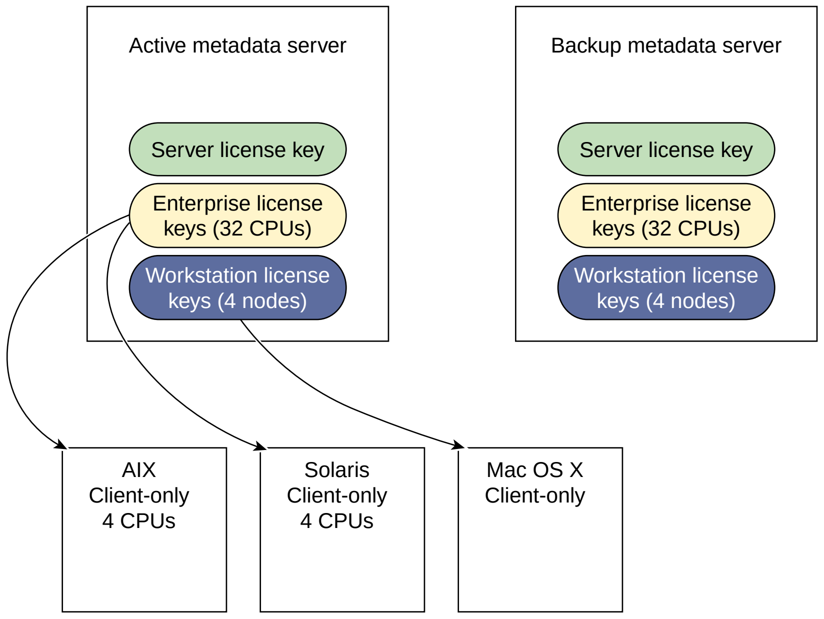

Active metadata server
Server license key
Enterprise license keys (32 CPUs)
Workstation license keys (4 nodes)
Backup metadata server
Server license key
Enterprise license keys (32 CPUs)
Workstation license keys (4 nodes)
AIX Client-only 4 CPUs
Solaris Client-only 4 CPUs
Mac OS X Client-only

Figure 5-1 Server-Side License Keys

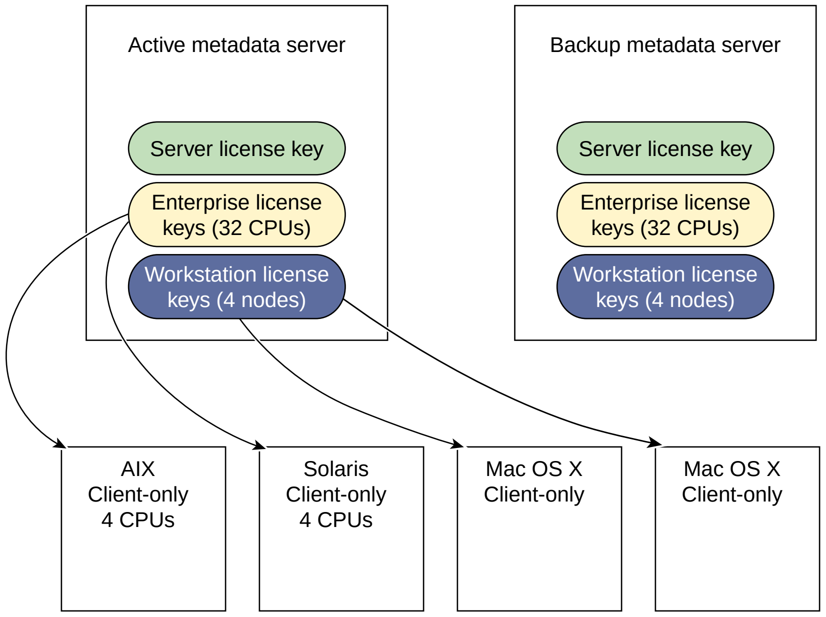

Active metadata server
Server license key
Enterprise license keys (32 CPUs)
Workstation license keys (4 nodes)
Backup metadata server
Server license key
Enterprise license keys (32 CPUs)
Workstation license keys (4 nodes)
AIX Client-only 4 CPUs
Solaris Client-only 4 CPUs
Mac OS X Client-only
Mac OS X Client-only

Figure 5-2 Server-Side License Keys: Adding a New Client-Only Node

The following table further illustrates the progressive changes in license keys required by a cluster as nodes are added and removed.

Table 5-1 Examples of License Keys Required for Cluster Changes

Action	Resulting Configuration	Licensing
Initial configuration	1 x 4-CPU server-capable administration node 4 x 2-CPU Windows clients	Purchase one 8-CPU `CXFS_SS` server license key and one 5-node workstation license key. Generate the 5-node workstation license key (`CXFS_SS_CLIENT_WRK`) for the server-capable administration node (the extra license is for future expansion).
Add a 2-CPU x86 (32-bit) Linux client	1 x 4-CPU server-capable administration node 4 x 2-CPU Windows clients 1 x 2-CPU x86 (32-bit) Linux client	No change, the 5-node workstation license key is now fully utilized.
Add an 8-CPU SGI ProPack 4 client	1 x 4-CPU sever-capable administration node 4 x 2-CPU Windows clients 1 x 2-CPU x86 (32-bit) Linux client 1 x 8-CPU SGI ProPack client	Purchase an 8-CPU enterprise license key. Generate the 8-CPU enterprise license key (`CXFS_SS_CLIENT_ENT`) for the server-capable administration node.
Add another 4-CPU server-capable administration node	2 x 4-CPU server-capable administration node 4 x 2-CPU Windows clients 1 x 2-CPU x86 (32-bit) Linux client 1 x 4-CPU SGI ProPack client	Purchase another 4-CPU server license. Generate both workstation and enterprise client license keys for the new server-capable administration node from the original license keys.
Add an 8-CPU Solaris node	2 x 4-CPU server-capable administration nodes 4 x 2-CPU Windows clients 1 x 2-CPU x86 (32-bit) Linux client 1 x 4-CPU SGI ProPack client 1 x 8-CPU Solaris client	Purchase an 8-CPU enterprise client license key. Generate the 8-CPU enterprise license key on each server-capable administration node.
Add a 4-CPU Mac OS X client for occasional use in the cluster	2 x 4-CPU server-capable administration nodes 4 x 2-CPU Windows clients 1 x 2-CPU x86 (32-bit) Linux client 1 x 4-CPU SGI ProPack client 1 x 4-CPU Mac OS X client	No change if one of the other workstation-class clients is dropped out of the cluster when the Mac OS X client is required.

Gathering the Host Information

When you order CXFS, you will receive an entitlement ID. You must submit the system host ID, host name, and entitlement ID when requesting your permanent CXFS license key.

To obtain the host information for a server-capable administration node, execute the following command (assuming that the LK `rpm` from SGI ProPack has been installed):

```
/usr/sbin/lk_hostid
```

For example, the following shows that the serial number is `N0000302` and the license ID is `e000012e`:

```
admin# /usr/sbin/lk_hostid

N0000302 e000012e socket=16 core=16 processor=16
```

Obtaining the License Keys from SGI

To obtain your CXFS and XVM license keys, see information provided in your customer letter and the following web page:

http://www.sgi.com/support/licensing

Installing the License Keys

You will install the license keys in the following location on the server-capable administration nodes:

```
/etc/lk/keys.dat
```

Do the following:

- Create the `/etc/lk` license key directory if necessary. For example:

  ```
  admin# mkdir -p /etc/lk
  ```

- Copy the keys to the `keys.dat` file.

Verifying the License Keys

This section discusses the following:

- "Verifying the License Keys with `cxfslicense`" on page 96
- "Verifying the License Keys with `lk_verify`" on page 98
- "Displaying the License Keys with `cxfs_admin`" on page 99

For information about verifying the XVM mirror licenses on client-only nodes, see the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

Verifying the License Keys with `cxfslicense`

To verify that the license keys have been installed properly, use the `cxfslicense -d` command after installing the CXFS software. Licensing errors will be reported to the `fs2d` log.

For example:

```
# /usr/cluster/bin/cxfslicense -d
Found 1 XVM_STD_IPF license(s) of version 1.0 for XVM.
Found 1 XVM_PLEX_IPF license(s) of version 1.0 for XVM.
Found 1 XVM_PLEX_CLUSTER license(s) of version 5.0 for XVM.

License(s) found: 1
Found 'CPU 8' serial 071446
Found 8 CPU version 5.0 license for CXFS_SSX
Server-side licensing is available

License(s) found: 1
Found license for 50 of CXFS_SS_CLIENT_WRK 5.0 serial 071822
License(s) found: 1
Found license for 256 of CXFS_SS_CLIENT_ENT 5.0 serial 071952
```

If no valid license is found, `cxfslicense -d` will report:

```
admin# /usr/cluster/bin/cxfslicense -d
Didn’t find XVM_STD_IPF license of version 1.0 for XVM
Didn’t find XVM_PLEX_IPF license(s) of version 1.0 for XVM.
Didn’t find XVM_PLEX_CLUSTER license of version 5.0 for XVM.

Cannot find valid version 5.0 license for CXFS_SSX

No CXFS server-side license, any server-side client licenses will be
ignored.

No licenses available for CXFS_SS_CLIENT_WRK 5.0.
No licenses available for CXFS_SS_CLIENT_ENT 5.0.

Error: No valid CXFS licenses found for this server.
```

If you do not have the CXFS license key properly installed, you will see the following error on the console when trying to run CXFS:

```
Starting CXFS services> ....
CXFS not properly licensed for this host. Run
"/usr/cluster/bin/cxfslicense -d"
for detailed failure information. After fixing the
license, please run "/usr/cluster/bin/cxfs_cluster restart".
```

An error such as the following example will appear in the `SYSLOG` file (line breaks added here for readability):

```
Jan 25 10:24:03 ncc1701:Jan 25 10:24:03 cxfs_client:
cis_main FATAL: cxfs_client failed the CXFS license check.
Use the cxfslicense command to diagnose the license problem
```

The following will appear in the client-log file:

- Successful:
  - Server license key granted, regardless of local client license key:

    ```
    Server-side license granted
    ```

- Unsuccessful (CXFS will not start):
  - Server denies a license key, regardless of local license key presence:

    ```
    A server-side license could not be granted
    ```

On an administration node, the error will appear in the `clconfd` log.

The `cxfs_admin status` command displays the number of server-side license keys that have been issued to clients. See "`cxfs_admin` and Status" on page 344.

Verifying the License Keys with `lk_verify`

You can use the `lk_verify -A` command to verify LK licenses. To see more output, use the `v` option. For example:

```
# lk_verify -A -vvv
lk_check        All     All : total found=10

  1 /etc/lk/keys.dat:004        product=CXFS_SSX, version=5.000, count=0, begDate=1200496489, \
       expDate=1208321999, licenseID=23d5fd92, key=BAYJ2caAnmOga5Z6G4GwRMmXwizlSek1, \
       info='CXFS SVR XE 8 CPU',attr='CPU 8', vendor='Silicon Graphics, Inc.', \
       ref_id='071446'
              Verdict:         SUCCESS. Nodelock.
                               Available since 16 days on 16-Jan-2008 09:14:49.
                               Will expire in 74 days on 15-Apr-2008 23:59:59

              Attribute 1 of 4 : info=CXFS SVR XE 8 CPU
              Attribute 2 of 4 : attr=CPU 8
              Attribute 3 of 4 : vendor=Silicon Graphics, Inc.
              Attribute 4 of 4 : ref_id=071446


  2 /etc/lk/keys.dat:009        product=CXFS_SS_CLIENT_WRK, version=5.000, count=0, begDate=1200496705, \
       expDate=1208321999, licenseID=23d5fd92, key=9Nge+ausgoJkP4Pabv0XuH9C7ybhVA4C, \
       info='CXFS WRK 50 NODE',attr='NODE 50', vendor='Silicon Graphics, Inc.', \
       ref_id='071822'
              Verdict:         SUCCESS. Nodelock.
                               Available since 16 days on 16-Jan-2008 09:18:25.
                               Will expire in 74 days on 15-Apr-2008 23:59:59

              Attribute 1 of 4 : info=CXFS WRK 50 NODE
```

```
                Attribute 2 of 4 : attr=NODE 50
                Attribute 3 of 4 : vendor=Silicon Graphics, Inc.
                Attribute 4 of 4 : ref_id=071822


  3 /etc/lk/keys.dat:014        product=CXFS_SS_CLIENT_ENT, version=5.000, count=0, begDate=1200496795, \
         expDate=1208321999, licenseID=23d5fd92, key=Q0EHNvcpIxiTq4lKUevxQluKFpu6SJBb, \
         info='CXFS ENT 256 CPU',attr='CPU 256', vendor='Silicon Graphics, Inc.', \
         ref_id='071952'
                Verdict:         SUCCESS. Nodelock.
                                 Available since 16 days on 16-Jan-2008 09:19:55.
                                 Will expire in 74 days on 15-Apr-2008 23:59:59

                Attribute 1 of 4 : info=CXFS ENT 256 CPU
                Attribute 2 of 4 : attr=CPU 256
                Attribute 3 of 4 : vendor=Silicon Graphics, Inc.
                Attribute 4 of 4 : ref_id=071952

...

lk_check        All     All : total matched=9
```

Displaying the License Keys with `cxfs_admin`

You can use the `cxfs_admin` command to display license information.

For example, the following is output from the `show licenses` command for the `cc_test2` cluster:

```
cxfs_admin:cc_test2> show licenses
status:licenses:
    cxfs_client:
        enterprise:
            allocated=27
            valid=256
        workstation:
            allocated=2
            valid=50
```

For example, the following is output from the `status` command for the `cc_test2` cluster:

```
cxfs_admin:cc_test2> status
Cluster    : cc_test2
Tiebreaker : minnesota
Licenses   : enterprise   allocated 27 of 256
             workstation  allocated 2 of 50
------------------  -------- ------------------------------------------------
Node                Cell ID   Status
------------------  -------- ------------------------------------------------
cc-xe *             0         Stable
cc-xe2 *            1         Stable
aiden               2         Stable
brenna              3         Stable
cc-mac1             4         Stable
cc-vista            5         Inactive
cc-win2003          6         Stable
cc-win64            7         Inactive
cc-winxp            8         Inactive
cxfsibm2            9         Mounted 0 of 2 filesystems
cxfssun4            10        Stable
gaeth               11        Stable
liam                12        Stable
lorcan              13        Stable
minnesota           14        Stable
nasca               15        Stable
padraig             16        Stable

------------------  ------------------ ---------------------------------------
Filesystem          Mount Point         Status
------------------  ------------------ ---------------------------------------
concatfs            /mnt/concatfs       cxfsibm2 trying to mount
mirrorfs            /mnt/mirrorfs       Unmounted
stripefs            /mnt/stripefs       cxfsibm2 trying to mount

------------------  ---------- ----------------------------------------------
Switch              Port Count  Known Fenced Ports
------------------  ---------- ----------------------------------------------
fcswitch12          32          None
fcswitch13          32          None
```

For More Information About Licensing

To request software keys or information about software licensing, see the following web page:

http://www.sgi.com/support/licensing

If you do not have access to the web, please contact your local Customer Support Center.

Chapter 6

Preinstallation Steps

When you install the CXFS software, you must modify certain system files. The network configuration is critical. Each node in the cluster must be able to communicate with every other node in the cluster by both logical name and IP address without going through any other network routing; proper name resolution is key. SGI recommends static routing.

This section provides an overview of the steps that you should perform on your nodes prior to installing the CXFS software. It contains the following sections:

- "Hostname Resolution and Network Configuration Rules" on page 103
- "Adding a Private Network" on page 104
- "Verifying the Private and Public Networks" on page 106

Hostname Resolution and Network Configuration Rules

Caution: It is critical that you understand these rules before attempting to configure a CXFS cluster.

Use the following hostname resolution rules and recommendations when defining a node:

- The first node you define in the pool must be an administration node.
- Hostnames cannot begin with an underscore (_) or include any white-space characters.
- The private network IP addresses on a running node in the cluster cannot be changed while CXFS services are active.
- You must be able to communicate directly between every node in the cluster (including client-only nodes) using IP addresses and logical names, without routing.
- You must dedicate a private network for control messages, CXFS metadata, CXFS kernel heartbeat messages, and cluster database heartbeat messages. No other load is supported on this network.

- The private must be connected to all nodes, and all nodes must be configured to use the same subnet for that network.
- Because CXFS kernel heartbeat and cluster database heartbeat are done using IP multicast, the private network must be multicast-capable. This means that all of the interfaces must have multicast enabled (which is the default) and all of the external networking hardware (such as switches) must support IP multicast.
- If you change hostname resolution settings in the `/etc/nsswitch.conf` file after you have defined the first administration node (which creates the cluster database), you must re-create the cluster database.

Adding a Private Network

The following procedure provides an overview of the steps required to add a private network.

Note: A private network is required for use with CXFS.

You may skip some steps, depending upon the starting conditions at your site.

1. Edit the `/etc/hosts` file so that it contains entries for every node in the cluster and their private interfaces as well.

   The `/etc/hosts` file has the following format, where *primary_hostname* can be the simple hostname or the fully qualified domain name:

   IP_address *primary_hostname* *aliases*

   You should be consistent when using fully qualified domain names in the `/etc/hosts` file. If you use fully qualified domain names on a particular node, then all of the nodes in the cluster should use the fully qualified name of that node when defining the IP/hostname information for that node in their `/etc/hosts` file.

   The decision to use fully qualified domain names is usually a matter of how the clients are going to resolve names for their client/server programs (such as NFS), how their default resolution is done, and so on.

Even if you are using the domain name service (DNS) or the network information service (NIS), you must add every IP address and hostname for the nodes to `/etc/hosts` on all nodes. For example:

```
190.0.2.1 server1-example.com server1
190.0.2.3 stocks
190.0.3.1 priv-server1
190.0.2.2 server2-example.com server2
190.0.2.4 bonds
190.0.3.2 priv-server2
```

You should then add all of these IP addresses to `/etc/hosts` on the other nodes in the cluster.

For more information, see the `hosts`(5) and `resolve.conf`(5) man pages.

Note: Exclusive use of NIS or DNS for IP address lookup for the nodes will reduce availability in situations where the NIS or DNS service becomes unreliable.

2. Edit the `/etc/nsswitch.conf` file so that local files are accessed before either NIS or DNS. That is, the `hosts` line in `/etc/nsswitch.conf` must list `files` first.

   For example:

   ```
   hosts:      files nis dns
   ```

   (The order of `nis` and `dns` is not significant to CXFS, but `files` must be first.)

3. Configure your private interface according to the instructions in the Network Configuration section of your Linux distribution manual. To verify that the private interface is operational, use the `ifconfig -a` command. For example:

```
admin# ifconfig -a

eth0      Link encap:Ethernet  HWaddr 00:50:81:A4:75:6A
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13782788 errors:0 dropped:0 overruns:0 frame:0
          TX packets:60846 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:826016878 (787.7 Mb)  TX bytes:5745933 (5.4 Mb)
          Interrupt:19 Base address:0xb880 Memory:fe0fe000-fe0fe038
```

```
eth1      Link encap:Ethernet  HWaddr 00:81:8A:10:5C:34
          inet addr:10.0.0.10  Bcast:10.0.0.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:19 Base address:0xef00 Memory:febfd000-febfd038

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:162 errors:0 dropped:0 overruns:0 frame:0
          TX packets:162 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:11692 (11.4 Kb)  TX bytes:11692 (11.4 Kb)
```

This example shows that two Ethernet interfaces, `eth0` and `eth1`, are present and running (as indicated by `UP` in the third line of each interface description).

If the second network does not appear, it may be that a network interface card must be installed in order to provide a second network, or it may be that the network is not yet initialized.

4. *(Optional)* Make the modifications required to use CXFS connectivity diagnostics. See "SGI ProPack Modifications for CXFS Connectivity Diagnostics" on page 117.

Verifying the Private and Public Networks

For each private network on each node in the pool, verify access with the `ping` command. Enter the following, where *nodeIPaddress* is the IP address of the node:

`ping` *nodeIPaddress*

For example:

```
admin# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) from 128.162.240.141 : 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.310 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.122 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.127 ms
```

Also execute a `ping` on the public networks. If `ping` fails, follow these steps:

1. Verify that the network interface was configured up using `ifconfig`. For example:

   ```
   admin# ifconfig eth1
   eth1      Link encap:Ethernet  HWaddr 00:81:8A:10:5C:34
             inet addr:10.0.0.10  Bcast:10.0.0.255  Mask:255.255.255.0
             UP BROADCAST MULTICAST  MTU:1500  Metric:1
             RX packets:0 errors:0 dropped:0 overruns:0 frame:0
             TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:100
             RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
             Interrupt:19 Base address:0xef00 Memory:febfd000-febfd038
   ```

   In the third output line above, `UP` indicates that the interface was configured up.

2. Verify that the cables are correctly seated.

Repeat this procedure on each node.

Chapter 7

Server-Capable Administration Node Installation

On SGI ProPack for Linux nodes, CXFS supports either an *administration node* containing the cluster administration daemons (`fs2d`, `crsd`, `cad`, and `cmond`), the CXFS control daemon (`clconfd`), and the cluster database or a *client-only node* containing the `cxfs_client` daemon. The software you install on a node determines the node type.

Note: SGI ProPack is an overlay product that adds or enhances features in the supported Linux base distributions.

Nodes that you intend to run as metadata servers must be installed as administration nodes; all other nodes should be client-only nodes.

This chapter discusses the following:

- "Installation Overview for Server-Capable Administration Nodes" on page 110
- "SGI ProPack Limitations and Considerations" on page 111
- "Installation from the ISSP Distribution" on page 112
- "Installation Verification" on page 117
- "SGI ProPack Modifications for CXFS Connectivity Diagnostics" on page 117

After completing these steps, see Chapter 9, "Initial Setup of the Cluster" on page 127. For details about specific configuration tasks, see Chapter 10, "Reference to GUI Tasks" on page 147.

Installation Overview for Server-Capable Administration Nodes

Caution: CXFS is a complex product. To ensure that it is installed and configured in an optimal manner, you **must** purchase initial setup services from SGI. You should read through the following chapters before attempting to install and configure a CXFS cluster:

- Chapter 1, "Introduction to CXFS" on page 1
- Chapter 2, "Best Practices" on page 43
- Chapter 3, "SGI RAID for CXFS Clusters" on page 77
- Chapter 4, "Switches" on page 81
- Chapter 5, "CXFS License Keys" on page 89
- Chapter 6, "Preinstallation Steps" on page 103
- Chapter 7, "Server-Capable Administration Node Installation" on page 109
- Chapter 8, "Postinstallation Steps" on page 119
- Chapter 9, "Initial Setup of the Cluster" on page 127

Also see the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

Following is the order of installation and configuration steps:

1. Install the operating system (if not already done). See the CXFS release notes for supported levels.

2. Install and verify the RAID. See Chapter 3, "SGI RAID for CXFS Clusters" on page 77

3. Install and verify the switch. See Chapter 4, "Switches" on page 81.

4. Obtain and install the CXFS license keys and (if needed) XVM license keys. See Chapter 5, "CXFS License Keys" on page 89.

5. Prepare the node, including adding a private network.

6. Install the CXFS software. For details, see Chapter 7, "Server-Capable Administration Node Installation" on page 109.

7. Configure the cluster to define the new node in the pool, add it to the cluster, start CXFS services, and mount filesystems. See "Guided Configuration Tasks" on page 169.

8. Install the client-only nodes, as described in *CXFS Administration Guide for SGI InfiniteStorage*.

SGI ProPack Limitations and Considerations

The following limitations and considerations apply to any SGI ProPack node (client-only or server-capable): See also client-only node information in *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

- By default, DMAPI is turned off on SGI ProPack 5 systems. When you install DMF on a server-capable administration node, it automatically enables DMAPI. However, if you want to mount filesystems on an SGI ProPack 5 client-only node with the `dmi` mount option, you must ensure that the `DMAPI_PROBE` system tunable parameter on the node is set to `yes` in the `/etc/sysconfig/sysctl` file. Changes to the file will be processed on the next reboot. After setting that system configuration file, you can immediately enable DMAPI by executing the following:

  ```
  sysctl -w fs.xfs.probe_dmapi=1
  ```

  If you run a DMAPI application other than DMF, you must also change parameter on the SGI ProPack 5 server-capable administration nodes.

- CXFS does not support external log or real-time filesystems.

- GPT partition tables, often created by operating system installers or the `parted` partitioning tool, store labels in two locations. If you reuse a disk that previously had a GPT label, you must be careful; using tools such as `fdisk` to repartition the drive will not eliminate the backup GPT label. When you reboot, EFI scans the disks before the operating system is started. It assumes any backup labels it finds are valid and restores them. This can corrupt or destroy filesystems. You can use the `parted` tool to detect this situation and fix it.

  Note: The `parted` tool has a `mkpartsect` command that accepts start and end values for partitions being created in sectors rather than MB. For more information, see the *XVM Volume Manager Administrator's Guide* and http://support.sgi.com/content_request/838562/index.html on Supportfolio.

- CXFS filesystems with XFS version 1 directory format cannot be mounted on SGI ProPack nodes.

- Whenever you install a new kernel patch, you must also install the corresponding CXFS package. This is required because the kernel patch causes the kernel version number to be increased. Failure to install the corresponding CXFS package will result in the inability to run CXFS. To obtain the required CXFS package, see your SGI support contact.

- The implementation of file creates using `O_EXCL` is not complete. Multiple applications running on the same node using `O_EXCL` creates as a synchronization mechanism will see the expected behavior (only one of the creates will succeed). However, applications running between nodes may not get the `O_EXCL` behavior they requested (creates of the same file from two or more separate nodes may all succeed).

Installation from the ISSP Distribution

The CXFS software will be initially installed and configured by SGI personnel. This section provides an overview of those procedures.

Note: Version numbers shown here are examples; your installed system may differ.

A node that may be a CXFS metadata server must be installed as a CXFS administration node. All other nodes should be client-only nodes.

Installing the CXFS software for a CXFS administration node requires approximately 65 MB of space.

Do the following to install the software required for an SGI ProPack administration node:

1. If you are upgrading from a previous release, stop the product software (such as DMF) on the server to be upgraded.
2. Log in to the system as `root` using `ssh`.
3. Confirm that you have installed SLES 10 SP1. (See the `/etc/SuSE-release` file.) Ensure that you have the appropriate kernel as listed in the *SGI InfiniteStorage Software Platform* release notes.
4. Confirm that you have installed SGI ProPack 5 SP 5. (See the `/etc/sgi-release` file.) For more information, see the *SGI ProPack 5 for Linux Service Pack 5 Start Here* guide.
5. Install any required patches. See the SGI ProPack `releasenotes/README` file for more information.
6. If you had to install software in one of the above steps, reboot the system:

   ```
   admin# /sbin/reboot
   ```

(Although no kernels are installed at this time, the install process will make some adjustments to the kernel modules that are loaded by default when booting. In addition, certain system tuning adjustments are made. Therefore, it is best to reboot so these changes take effect.) Allow approximately 5 minutes for the system to restart.

7. Install the CXFS software from one of the following sources:

   - DVD:

     1. Insert the DVD for your server architecture:

        – *SGI InfiniteStorage Software Platform 1.2 for Altix ia64*

        – *SGI InfiniteStorage Software Platform 1.2 for Altix XE x86_64*

     2. Read the *SGI InfiniteStorage Software Platform* release notes, CXFS general release notes, and CXFS SGI ProPack release notes from the `/docs` directory on the ISSP DVD and any late-breaking caveats on Supportfolio. (For Supportfolio access, see the following bullet item.)

   - Supportfolio:

     1. Enter Supportfolio Online:

        `https://support.sgi.com/login`

     2. In the Software Services area, select Software Ordering.

     3. Select from your list of registered serial numbers and click **CONTINUE**. This will return a list of entitled products. (Only the base release is listed.)

     4. *SGI InfiniteStorage Software Platform* release notes, CXFS general release notes, and CXFS AIX release notes any late-breaking caveats.

     5. Select the update software by checking **Add to cart** and the download box and clicking **CONTINUE**. The selected product will appear on the screen. Select the software you want to download.

     6. Click **SUBMIT Your Order**. The new screen will contain the list of releases. Click on **/content/request** to the right of the release to display an additional page containing the list of individual downloads.

     7. Download the software onto your system.

Note: If you did not select all of the ISSP software available for download, you will get conflicts when you install it. However, this is expected and you can just acknowledge the conflict.

---

8. Start the YaST software management tool:

   ```
   [root@linux root]# /sbin/yast2
   ```

9. Add **SGI InfiniteStorage Software Platform** for your system to the list of source media, which must include the base OS and SGI ProPack software:

   a. Select the **Software** group in the left pane of the YaST **Control Center** screen.

   b. Click **Installation Source** in the right pane, which opens another window.

   c. If there are any ISSP software installation sources present, remove them by highlighting them one at a time and selecting **Delete**.

   d. Click **Add** and select the DVD or directory. YaST will then add the DVD or directory to the list of media sources.

   e. Click **Finish** to apply the changes and close the window.

10. Install the software from the ISSP distribution:

    a. In the YaST **Control Center** window, click **Software**.

    b. Click **Software Management**.

    c. Click **Patterns** from the pull-down menu to the right of **Filter** on the upper left part of the screen.

    Note: Make sure that you are working with the left-hand window with the **Pattern** title and not the individual package window on the right.

    ---

11. Scroll down the list of product patterns until you can see the SGI patterns at the end.

12. Select your software choices by clicking the small square boxes to the left. The **SGI ISSP Common** pattern will be automatically selected when you make any other choice. You may select one or more options for which you have licenses. To

see what will be installed with any given pattern, see the section about product versions in the ISSP release notes.

Select the following for standalone CXFS[1] :

SGI CXFS Server
SGI CXFS Clients *(install on only one CXFS server)*
SGI DMF Server *(optional)*
SGI HA Server *(optional)*

For example, to install standalone CXFS with HA, you would select both of the following:

SGI CXFS Server
SGI CXFS Clients *(install on only one CXFS server)*
SGI HA Server

13. Verify that the product groups you wish to install now have a check mark in the box. A small triangle may appear to the left of the check box for auto-installed groups. Click **OK** to verify your choices.

    Note: Do not attempt to install software for which you do not have a license.

    You can also verify all of the packages that are going to be installed by choosing the **Installation Summary** option in the **Filter** menu of the **Software Management** view.

14. If you do not want to install GRIO, deselect its RPMs.

15. Click **Accept** to install the packages.

1 For CXFS running in a SAN Solution Server, see the instructions in the *SGI InfiniteStorage Software Platform* release notes.

Note: It is normal to be presented with a **Changed Packages** box. This box lists packages from the OS that are required by the ISSP software. In order for these packages to be installed, you must have your OS install source available.

In some cases, you could be presented with conflicts. These are a result of unresolved dependencies from a previous installation or installation attempt. If you have unresolved dependencies, you must resolve them before the installation of software from the ISSP media kit can proceed. Resolving these conflicts might require that some packages from the OS be installed. In this case, you must have your OS install source available.

Sometimes an out-of-date package can present conflicts. These conflicts should be resolved by ensuring that the correct versions of the OS and SGI ProPack are installed.

---

16. Start the file alteration monitoring (`fam`) service, which is required for the GUI's use of task privileges for users:

    ```
    admin# service fam start
    Starting File Access Monitoring Daemon                       done
    ```

    Enable the `fam` service with `chkconfig` so that the `fam` service automatically starts on a reboot:

    ```
    admin# chkconfig fam on
    ```

17. Copy the license keys for your system to the `/etc/lk/keys.dat` file. For more information, see "Verifying the License Keys with `cxfslicense`" on page 96.

18. If XFS is built into your kernel but your root filesystem **is not** of type `XFS`, run the following commands:

    ```
    admin# depmod
    admin# mkinitrd
    admin# elilo
    ```

19. Reboot the system in order to make the new updates to take effect.

20. Transfer the client-only software for each of your client-only nodes from the server node on which they were installed (above) and install as directed in the *CXFS*

MultiOS Client-Only Guide for SGI InfiniteStorage. The CXFS client packages will be installed on the server-capable administration node in the following directory:

/usr/cluster/client-dist/*CXFS_VERSION*/*CLIENT_PLATFORM*/

Installation Verification

To verify that the CXFS software has been installed properly, use the `rpm -qa` command to display all of the installed packages. You can filter the output by searching for particular package name.

For example, to verify that the `cxfs-sysadm_base-lib` package has installed:

```
admin]# rpm -qa | grep cxfs-sysadm_base-lib
cxfs-sysadm_base-lib-3.0-sgi06092521
```

Note: The output above is an example. The version level may not match the installed software.

To verify the SGI ProPack release, display the `/etc/sgi-release` file.

SGI ProPack Modifications for CXFS Connectivity Diagnostics

If you want to use the cluster diagnostics to test node connectivity, the `root` user on the node running the CXFS diagnostics must be able to access a remote shell using the `rsh` command (as `root`) on all other nodes in the cluster. There are several ways of accomplishing this, depending on the existing settings in the pluggable authentication modules (PAM) and other security configuration files.

Following is one possible method. Do the following on all administration nodes in the cluster:

1. Install the `rsh-server` RPM using YaST.
2. Enable `rsh` by changing `disable yes` to `disable no` in the `/etc/xinetd.d/rsh` file.
3. Restart `xinetd`:

   ```
   admin# service xinetd restart
   ```

4. Add the hostname of the node from which you will be running the diagnostics into the `/root/.rhosts` file. Make sure that the mode of the `.rhosts` file is set to `600` (read and write access for the owner only).

After you have completed running the connectivity tests, you may wish to disable `rsh` on all cluster nodes.

For more information, see the operating system documentation and the `hosts.equiv`(5) man page.

Chapter 8

Postinstallation Steps

This chapter discusses the following:

- "Configuring System Files" on page 119
- "SGI ProPack: Using `cxfs-reprobe` on Client-Only Nodes" on page 125

After completing these step discussed in this chapter, see Chapter 9, "Initial Setup of the Cluster" on page 127. For details about specific configuration tasks, see Chapter 10, "Reference to GUI Tasks" on page 147. For information about upgrades, see "CXFS Release Versions and Rolling Upgrades" on page 277.

Configuring System Files

When you install the CXFS software, there are some system file considerations you must take into account. **The network configuration is critical.** Each node in the cluster must be able to communicate with every other node in the cluster by both logical name and IP address without going through any other network routing; proper name resolution is key. SGI recommends static routing.

This section discusses the following:

- "`/etc/exports` on All Nodes" on page 119
- "Administration Node System Files" on page 120

`/etc/exports` on All Nodes

The optional `/etc/exports` file on each node describes the filesystems that are being exported to NFS clients.

If the `/etc/exports` file contains a CXFS mount point, then when the system is booted NFS will export the empty mount point because the exports are done before CXFS is running. When CXFS on the node joins membership and starts mounting filesystems, the `clconfd-pre-mount` script searches the `/etc/exports` file looking for the mountpoint that is being mounted. If found, the script unexports the mountpoint directory because if it did not the CXFS mount would fail. After successfully mounting the filesystem, the `clconfd-post-mount` script will search the `/etc/exports` file and export the mount point if it is found in the `/etc/exports` file.

For more information, see "CXFS Mount Scripts" on page 291.

Administration Node System Files

This section discusses system files on administration nodes:

- "`/etc/services` on Server-Capable Administration Nodes" on page 120
- "`cad.options` on Server-Capable Administration Nodes" on page 120
- "`fs2d.options` on Server-Capable Administration Nodes" on page 121
- "`clconfd.options` on Server-Capable Administration Nodes" on page 124

`/etc/services` on Server-Capable Administration Nodes

The `/etc/services` file on each CXFS administration contains entries for `sgi-cad` and `sgi-crsd`. The port numbers assigned for these processes must be the same in all nodes in the pool.

The following shows an example of `/etc/services` entries for `sgi-cad` and `sgi-crsd`:

```
sgi-crsd        7500/udp           # Cluster reset services daemon
sgi-cad         9000/tcp           # Cluster Admin daemon
```

`cad.options` on Server-Capable Administration Nodes

The `cad.options` file on each server-capable administration node contains the list of parameters that the cluster administration daemon reads when the `cad` process is started. The file is located as follows:

```
/etc/cluster/config/cad.options
```

`cad` provides cluster information.

The following options can be set in the `cad.options` file:

`--append_log`	Append `cad` logging information to the `cad` log file instead of overwriting it.

`--log_file` *filename*	`cad` log filename. Alternately, this can be specified as `-lf` *filename.*
`-vvvv`	Verbosity level. The number of `v` characters indicates the level of logging. Setting `-v` logs the fewest messages; setting `-vvvv` logs the highest number of messages.

The default file has the following options:

```
-lf /var/cluster/ha/log/cad_log --append_log
```

The following example shows an `/etc/config/cad.options` file that uses a medium-level of verbosity:

```
-vv -lf /var/cluster/ha/log/cad_nodename --append_log
```

The default log file is `/var/cluster/ha/log/cad_log`. Error and warning messages are appended to the log file if log file is already present.

The contents of the `/etc/config/cad.options` file cannot be modified using `cxfs_admin` or the GUI.

If you make a change to the `cad.options` file at any time other than initial configuration, you must restart the `cad` processes in order for these changes to take effect. You can do this by rebooting the nodes or by entering the following command:

```
admin# service cxfs_cluster restart
```

If you execute this command on a running cluster, it will remain up and running. However, the GUI will lose connection with the `cad` daemon; the GUI will prompt you to reconnect.

`fs2d.options` on Server-Capable Administration Nodes

The `fs2d.options` file on each server-capable administration node contains the list of parameters that the `fs2d` daemon reads when the process is started. (The `fs2d` daemon manages the distribution of the cluster database (CDB) across the server-capable administration nodes in the pool.) The file is located as follows:

```
/etc/cluster/config/fs2d.options
```

Table 8-1 shows the options can that can be set in the `fs2d.options` file.

Table 8-1 `fs2d.options` File Options

Option	Description
`-logevents` *event name*	Log selected events. The following event names may be used: `all`, `internal`, `args`, `attach`, `chandle`, `node`, `tree`, `lock`, `datacon`, `trap`, `notify`, `access`, `storage`. The default is `all`.
`-logdest` *log destination*	Set log destination. The following log destinations may be used: `all`, `stdout`, `stderr`, `syslog`, `logfile`. If multiple destinations are specified, the log messages are written to all of them. If `logfile` is specified, it has no effect unless the `-logfile` option is also specified. The default is `logfile`.
`-logfile` *filename*	Set log filename. The default is `/var/cluster/ha/log/fs2d_log`.
`-logfilemax` *maximum size*	Set log file maximum size (in bytes). If the file exceeds the maximum size, any preexisting `filename.old` will be deleted, the current file will be renamed to `filename.old`, and a new file will be created. A single message will not be split across files. If `-logfile` is set, the default is `10000000`.
`-loglevel` *loglevel*	Set log level. The following log levels may be used: `always`, `critical`, `error`, `warning`, `info`, `moreinfo`, `freq`, `morefreq`, `trace`, `busy`. The default is `info`.
`-trace` *trace_class*	Trace selected events. The following trace classes may be used: `all`, `rpcs`, `updates`, `transactions`, `monitor`. If you specify this option, you must also specify `-tracefile` and/or `-tracelog`. No tracing is done, even if it is requested for one or more classes of events, unless either or both of `-tracefile` or `-tracelog` is specified. The default is `transactions`.
`-tracefile` *filename*	Set trace filename. There is no default.
`-tracefilemax` *maximum_size*	Set trace file maximum size (in bytes). If the file exceeds the maximum size, any preexisting `filename.old` will be deleted, the current file will be renamed to `filename.old`, and a new file will be created.
`-[no]tracelog`	[Do not] trace to log destination. When this option is set, tracing messages are directed to the log destination or destinations. If there is also a trace file, the tracing messages are written there as well. The default is `-tracelog.`
`-[no]parent_timer`	[Do not] exit when the parent exits. The default is `-noparent_timer`.

Option	Description
`-[no]daemonize`	[Do not] run as a daemon. The default is `-daemonize`.
`-l`	Do not run as a daemon.
`-h`	Print usage message.
`-o help`	Print usage message.

If you use the default values for these options, the system will be configured so that all log messages of level `info` or less, and all trace messages for transaction events, are sent to the `/var/cluster/ha/log/fs2d_log` file. When the file size reaches 10 MB, this file will be moved to its namesake with the `.old` extension and logging will roll over to a new file of the same name. A single message will not be split across files.

If you make a change to the `fs2d.options` file at any time other than the initial configuration time, you must restart the `fs2d` processes in order for those changes to take effect. You can do this by rebooting the server-capable administration nodes or by entering the following command:

```
admin# service cxfs_cluster restart
```

If you execute this command on a running cluster, it should remain up and running. However, the GUI will lose connection with the `cad` daemon; the GUI will prompt you to reconnect.

Example 1

The following example shows an `/etc/config/fs2d.options` file that directs logging and tracing information as follows:

- All log events are sent to: `/var/log/messages`
- Tracing information for RPCs, updates, and transactions are sent to `/var/cluster/ha/log/fs2d_ops1`.

  When the size of this file exceeds 100,000,000 bytes, this file is renamed to `/var/cluster/ha/log/fs2d_ops1.old` and a new file `/var/cluster/ha/log/fs2d_ops1` is created. A single message is not split across files.

(Line breaks added for readability.)

```
-logevents all -loglevel trace -logdest syslog -trace rpcs
-trace updates -trace transactions -tracefile /var/cluster/ha/log/fs2d_ops1
-tracefilemax 100000000
```

Example 2

The following example shows an `/etc/config/fs2d.options` file that directs all log and trace messages into one file, `/var/cluster/ha/log/fs2d_chaos6`, for which a maximum size of 100,000,000 bytes is specified. `-tracelog` directs the tracing to the log file.

(Line breaks added for readability.)

```
-logevents all -loglevel trace -trace rpcs -trace updates
-trace transactions -tracelog -logfile /var/cluster/ha/log/fs2d_chaos6
-logfilemax 100000000 -logdest logfile.
```

`clconfd.options` on Server-Capable Administration Nodes

You can use the `clconfd.options` file on each server-capable administration node to contain a list of nondefault parameters that the `clconfd` daemon will read when the process is started. To use this feature, create the following file:

```
/etc/cluster/config/clconfd.options
```

Table 8-2 shows the options that can be set in the `fs2d.options` file.

Table 8-2 `clconfd.options` File Options

Option	Description
`-c` *CDBfile*	Reads the cluster database configuration from the specified *CDBfile* file. The default file is `/var/cluster/cdb/cdb.db`.
`-d` *debugfile*	Enables printing `hafence` debug information to the specified file *debugfile*. The default is to print no information.
`-h`	Prints a help message for `clconfd.options`.

Option	Description
`-l`	Runs `clconfd` in the foreground. (For SGI development debugging purposes only. Do not use this option unless directed to do so by SGI support.) The default is to run `clconfd` in the background.
`-s` *loglevel*	Specifies the log level to use for logging to standard error. The default is `0` (no logging). For information about log levels, see "Configure Log Groups with the GUI" on page 192.
`-R`	Disables real-time scheduling. By default, real-time scheduling is enabled.

For example, to print `hafence` debug information to the file `/tmp/hafence.log`, add the following line to the `clconfd.options` file:

```
-d /tmp/hafence.log
```

If you make a change to the `clconfd.options` file at any time other than the initial configuration time, you must restart the `clconfd` processes in order for those changes to take effect. You can do this by rebooting the server-capable administration nodes or by entering the following command:

```
admin# service cxfs restart
```

SGI ProPack: Using `cxfs-reprobe` on Client-Only Nodes

When `cxfs_client` needs to rescan disk buses, it executes the `/var/cluster/cxfs_client-scripts/cxfs-reprobe` script. This requires the use of parameters in SGI ProPack due to limitations in the Linux SCSI layer. You can export these parameters from the `/etc/cluster/config/cxfs_client.options` file.

The `cxfs_reprobe` script detects the presence of the SCSI layer on the system and probes all SCSI layer devices by default. You can override this decision by setting `CXFS_PROBE_SCSI` to `0` to disable the probe or `1` to force the probe (default).

When a SCSI scan is performed, all buses/channels/IDs and LUNs are scanned by default to ensure that all devices are found. You can override this decision by setting one or more of the environment variables listed below. This may be desired to reduce lengthy probe times.

The following summarizes the environment variables (separate multiple values by white space and enclose within single quotation marks):

`CXFS_PROBE_SCSI=`*0*|*1*

Stops (`0`) or forces (`1`) a SCSI probe. Default: `1`

`CXFS_PROBE_SCSI_BUSES=`*BusList*

Scans the buses listed. Default: All buses (`-`)

`CXFS_PROBE_SCSI_CHANNELS=`*ChannelList*

Scans the channels listed. Default: All channels (`-`)

`CXFS_PROBE_SCSI_IDS=`*IDList*

Scans the IDs listed. Default: All IDs (`-`)

`CXFS_PROBE_SCSI_LUNS=`*LunList*

Scans the LUNs listed. Default: All LUNs (`-`)

For example, the following would only scan the first two SCSI buses:

```
export CXFS_PROBE_SCSI_BUSES='0 1'
```

The following would scan 16 LUNs on each bus, channel, and ID combination (all on one line):

```
export CXFS_PROBE_SCSI_LUNS='0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15'
```

Other options within the `/etc/cluster/config/cxfs_client.options` file begin with a `-` character. Following is an example `cxfs_client.options` file:

```
# Example cxfs_client.options file
#
-Dnormal -serror
export CXFS_PROBE_SCSI_BUSES=1
export CXFS_PROBE_SCSI_LUNS='0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20'
```

Note: The `-` character or the term `export` must start in the first position of each line in the `cxfs_client.options` file; otherwise, they are ignored by the `/etc/init.d/cxfs_client` script.

Chapter 9

Initial Setup of the Cluster

Caution: CXFS is a complex product. To ensure that it is installed and configured in an optimal manner, you **must** purchase initial setup services from SGI. You should read through the following chapters, before attempting to install and configure a CXFS cluster:

- Chapter 1, "Introduction to CXFS" on page 1
- Chapter 3, "SGI RAID for CXFS Clusters" on page 77
- Chapter 4, "Switches" on page 81
- Chapter 5, "CXFS License Keys" on page 89
- Chapter 6, "Preinstallation Steps" on page 103
- Chapter 7, "Server-Capable Administration Node Installation" on page 109
- Chapter 8, "Postinstallation Steps" on page 119
- Chapter 2, "Best Practices" on page 43
- Chapter 9, "Initial Setup of the Cluster" on page 127

This chapter provides recommendations and a summary of the basic steps required to initially configure a cluster. It contains the following:

- "Preliminary Cluster Configuration Steps" on page 128
- "Initial Setup Using One of the Configuration Tools" on page 130
- "Configuring a Large Cluster" on page 139
- "Testing the System" on page 141

You should also refer to the information in "Configuration Best Practices" on page 43 and you may wish to use the worksheet provided in Appendix K, "Initial Configuration Checklist" on page 615.

This chapter points to detailed descriptions in the task reference chapters and in the *XVM Volume Manager Administrator's Guide*.

For information about licenses, see Chapter 5, "CXFS License Keys" on page 89.

Preliminary Cluster Configuration Steps

Complete the following steps to ensure that you are ready to configure the initial cluster:

- "Verify the License" on page 128
- "Verify that the Cluster Daemons are Running" on page 128
- "Gather the Required Information" on page 129
- "Configure for `nsd` Use (Optional)" on page 130
- "Verify that the `chkconfig` Arguments are On" on page 130

During the course of configuration, you will see various information-only messages in the log files. See "Normal Messages" on page 383.

Verify the License

Verify that you have the appropriate CXFS licenses by using the `-d` option to the `cxfslicense` command on server-capable administration nodes. See "Verifying the License Keys with `cxfslicense`" on page 96.

Verify that the Cluster Daemons are Running

When you **first install** the software, the following daemons should be running on all server-capable administration nodes:

- `fs2d`
- `cmond`
- `cad`
- `crsd`

To determine which daemons are running on a server-capable administration node, enter the following:

```
admin# service cxfs_cluster status
fs2d is running.
cmond is running.
cad is running.
crsd is running.
```

If you do not see these processes on a server-capable administration node, go to the logs to see what the problem might be. Then restart the daemons by entering the following:

```
admin# service cxfs_cluster start
```

The `cxfs_client` daemon should be running on a client-only node. If it is not, enter the following:

```
client# service cxfs_client start
```

The `ggd2` daemon should be running on a server-capable administration node if you are running GRIOv2. If it is not, enter the following:

```
admin# service grio2 start
```

For more information, see "Stopping and Restarting Cluster Administration Daemons" on page 412 and "Kernel Threads" on page 419.

Gather the Required Information

You should know the fully qualified hostname of the machine from which you will do CXFS administration, which should be the first node you define in the cluster database. If you use `cxfs_admin` (see "Initial Setup Using One of the Configuration Tools" on page 130), you should use the hostname when defining the first node in the pool. (This information is automatically supplied for you in the CXFS GUI.)

You should also know the IP addresses and hostnames of the other machines that will form the cluster and the name by which want to refer to the cluster.

Configure for nsd Use (Optional)

If your system uses `nsd` for hostname resolution, you must configure your system so that local files are accessed before the network information service (NIS) or the domain name service (DNS).

Verify that the `chkconfig` Arguments are On

Ensure that the appropriate `chkconfig` arguments are `on`. For more information, see "`chkconfig` Arguments" on page 288.

For an SGI ProPack node, use `--list` option to `chkconfig` to verify that the `chkconfig` names are set to `on` for the site's normal run levels. For example, if the normal run levels were 3 and 5:

```
admin# /sbin/chkconfig --list | grep cxfs
cxfs_cluster    0:off   1:off   2:off   3:on    4:off   5:on    6:off
cxfs            0:off   1:off   2:off   3:on    4:off   5:on    6:off
```

Note: Your site's normal run levels may differ.

If the normal run levels are set to `off`, set them to `on` and reboot. For example:

```
admin# /sbin/chkconfig cxfs_cluster on
admin# /sbin/chkconfig cxfs on
admin# /sbin/chkconfig grio2 on (if running GRIOv2)
admin# reboot
```

Initial Setup Using One of the Configuration Tools

You can create the cluster and its components using any one of the following tools, which provide similar functionality:

- "Initial Setup with the CXFS GUI" on page 131
- "Initial Setup with the `cxfs_admin` Command" on page 135

Caution: You should only use one configuration tool at a time to make changes.

The following procedures provide an overview of the basic steps to set up a cluster. You will first define a server-capable administration node from which you perform administrative tasks, and then the other components of the cluster.

Initial Setup with the CXFS GUI

Note: For complete details about using the GUI, see "CXFS Tools" on page 36 and Chapter 10, "Reference to GUI Tasks" on page 147.

To initially configure the cluster with GUI, do the following:

- "Start the GUI" on page 131
- "Set Up a New Cluster with the GUI" on page 133
- "Set Up a New CXFS Filesystem with the GUI" on page 134

The server-capable administration node to which you connect the GUI affects your view of the cluster. You should wait for a change to appear in the view area before making another change; the change is not guaranteed to be propagated across the cluster until it appears in the view area. You should only make changes from one instance of the GUI at any given time; changes made by a second GUI instance may overwrite changes made by the first instance.

Start the GUI

Start the CXFS Manager by entering the following:

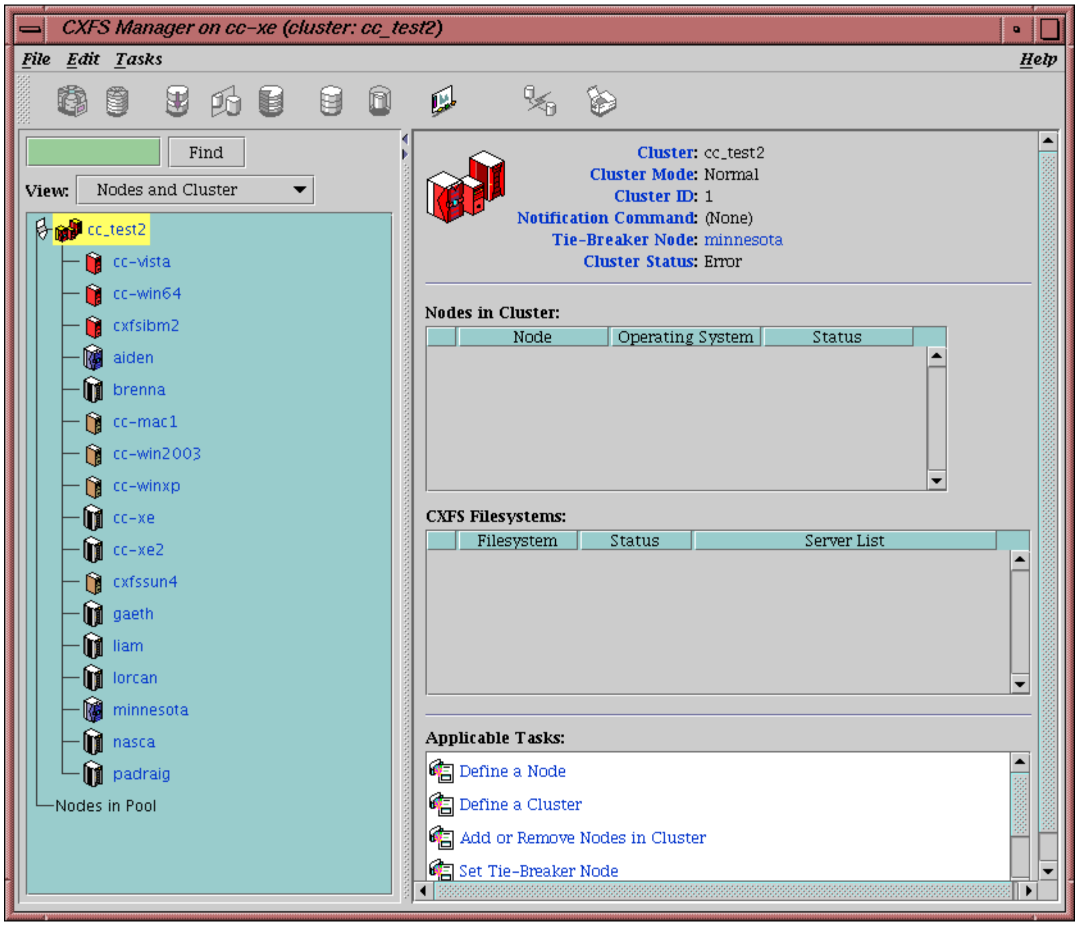
```
admin# /usr/sbin/cxfsmgr
```

You can also start the GUI from your web browser on a Microsoft Windows, Linux, or other platform. To do this, enter `http://`*server*`/CXFSManager/` (where *server* is the name of a server-capable administration node in the pool) and press **Enter**. At the resulting webpage, click the CXFS Manager icon. This method of launching CXFS Manager requires you to have enabled Java in your browser's preferences and have installed the appropriate Java plug-in. (After installing the plug-in, you must close any existing Java windows and restart your browser.) The server-capable administration node must be running a web server, such as Apache, and have the `cxfs-sysadm_cxfs-web` software installed.

There are other methods of starting the GUI. For more information, see "Starting the GUI" on page 148.

Supply the name of the server-capable administration node you wish to connect to and the `root` password.

Figure 9-1 shows an example of the CXFS Manager window.



Cluster: cc_test2
Cluster Mode: Normal
Cluster ID: 1
Notification Command: (None)
Tie-Breaker Node: minnesota
Cluster Status: Error

Nodes in Cluster:

Node	Operating System	Status

CXFS Filesystems:

Filesystem	Status	Server List

Applicable Tasks:

Define a Node
Define a Cluster
Add or Remove Nodes in Cluster
Set Tie-Breaker Node

Figure 9-1 CXFS Manager

Set Up a New Cluster with the GUI

Within the CXFS tasks, you can click any **blue** text to get more information about that concept or input field. In every task, the cluster configuration will not update until you click on **OK**.

Note: To specify reset method that uses Intelligent Platform Management Interface (IPMI) and baseboard management controller (BMC), you must use the `cxfs_admin` configuration tool. See "Create or Modify a Node with `cxfs_admin`" on page 235.

The **Set Up a New Cluster** task in the **Guided Configuration** menu leads you through the steps required to create a new cluster. It encompasses tasks that are detailed elsewhere.

Do the following:

1. Click **Define a Node** to define the server-capable administration node to which you are connected. See "Define a Node with the GUI" on page 171.

   Note: If you attempt to define a cluster or other object before the local node has been defined, you will get an error message that says:

   ```
   No nodes are registered on servername. You cannot define a cluster
   until you define the node to which the GUI is connected. To do so,
   click "Continue" to launch the "Set Up a New Cluster" task.
   ```

   ---

2. (*Optional*) **After** the first node icon appears in the view area on the left, click step 2, **Define a Node**, to define the other nodes in the cluster. To use private network failover, you must use the `cxfs_admin` command's `create failover_net` command to specify the network and mask; see "Network Failover Tasks with `cxfs_admin`" on page 262. See "Define a Node with the GUI" on page 171.

   Note: Do not define another node until this node appears in the view area. If you add nodes too quickly (before the database can include the node), errors will occur.

   Repeat this step for each node. For large clusters, define only the server-capable administration nodes first; see "Configuring a Large Cluster" on page 139.

3. Click **Define a Cluster** to create the cluster definition. See "Define a Cluster with the GUI" on page 186. Verify that the cluster appears in the view area. Choose **View: Nodes and Cluster**.
4. After the cluster icon appears in the view area, click **Add/Remove Nodes in Cluster** to add the nodes to the new cluster. See "Add or Remove Nodes in the Cluster with the GUI" on page 180.

   Click **Next** to move to the second screen of tasks.
5. (*Optional*) Click on **Test Connectivity** to verify that the nodes are physically connected. See "Test Node Connectivity with the GUI" on page 185. (This test requires the proper configuration; see "SGI ProPack Modifications for CXFS Connectivity Diagnostics" on page 117.)
6. If you are using I/O fencing, define the switch in the cluster; see the release notes for supported switches. I/O fencing is required for nodes without system controllers; see "Hardware and Software Requirements for Server-Capable Administration Nodes" on page 34.
7. Click **Start CXFS Services**. See "Start CXFS Services with the GUI" on page 189.
8. Click **Close**. Clicking on **Close** exits the task; it does not undo the task.

Set Up a New CXFS Filesystem with the GUI

Note: Within the CXFS tasks, you can click any blue text to get more information about that concept or input field. In every task, the cluster configuration will not update until you click **OK**.

The **Set Up a New CXFS Filesystem** task leads you through the steps required to create a new filesystem and mount it on all nodes in your cluster. It encompasses tasks that are detailed elsewhere.

Do the following:

1. Click **Start CXFS Services** if the services have not been started already. (The current status is displayed beneath the task link.) See "Start CXFS Services with the GUI" on page 189.
2. Click **Label Disks**.

Note: The disk must be initialized before being labeled. If your disk has not been initialized during factory set-up, use the SGI ProPack `fdisk` command to initialize the disk.

For information about XVM tasks, see the *XVM Volume Manager Administrator's Guide*.

3. Create slices, which define the physical storage, on the labeled disk. Click **Slice Disks**.

4. Create the type of filesystem you want: stripe, mirror, or concat.

5. Click **Make the Filesystem**. If you do not want to use the default options, click **Specify Sizes** and go to the next page. For more information, see the `mkfs`(8) man page and the *XVM Volume Manager Administrator's Guide*.

6. Click **Define a CXFS Filesystem**. This task lets you define a new filesystem, set the ordered list of potential metadata servers, and set the list of client nodes for the filesystem. See "Define CXFS Filesystems with the GUI" on page 202.

7. Click **Mount a CXFS Filesystem**. This task lets you mount the filesystem on all nodes in the cluster. See "Mount CXFS Filesystems with the GUI" on page 207.

Repeat these steps for each filesystem.

Initial Setup with the `cxfs_admin` Command

Note: For the initial installation, SGI highly recommends that you use the GUI guided configuration tasks. See "Initial Setup with the CXFS GUI" on page 131. For complete details about using `cxfs_admin`, see "CXFS Tools" on page 36 andChapter 11, "Reference to `cxfs_admin` Tasks" on page 217.

You can perform configuration with `cxfs_admin` using normal mode (in which you specify each command and attribute) or in prompting mode, in which `cxfs_admin` asks you for the information it requires.

To initially configure the cluster with `cxfs_admin`, do the following (line breaks shown here for readability). A simple example of prompting mode follows the steps.

1. "Preliminary Cluster Configuration Steps" on page 128.

2. Initialize the cluster database and start `cxfs_admin`:

   ```
   admin# cxfs_admin -s
   ```

3. Define the cluster name, where *clustername* is the logical name of the cluster:

   ```
   cxfs_admin> create cluster name=clustername
   ```

   For example:

   ```
   cxfs_admin> create cluster name=mycluster
   ```

4. Create the first server-capable administration node (normally the node on which you are currently running `cxfs_admin`). (You do not need to specify the node type because it must be `server_admin`.) If you use prompting mode, the name of the local node is used as a default for `name`.

   Caution: It is critical that you enter the primary hostname for the first node defined in the pool.

```
cxfs_admin> create node name=server_capable_hostname private_net=private_IPaddress
```

   For example:

```
cxfs_admin> create node name=server1 private_net=10.11.20.114
```

5. Exit `cxfs_admin` and restart the CXFS cluster services:

   ```
   admin# service grio2 stop (if running GRIOv2)
   admin# service cxfs stop
   admin# service cxfs_cluster stop
   admin# service cxfs_cluster start
   admin# service cxfs start
   admin# service grio2 start (if running GRIOv2)
   ```

6. Restart `cxfs_admin`:

   ```
   admin# cxfs_admin
   ```

   Note: If you have multiple clusters using the same public network as the backup CXFS metadata network, use the `-i` option to identify the cluster name:

   ```
   admin# cxfs_admin -i mycluster
   ```

   ---

7. *(Optional)* Create the failover networks:

```
cxfs_admin:cluster> create failover_net network=IPaddress1 mask=netmask
cxfs_admin:cluster> create failover_net network=IPaddress2 mask=netmask
```

   For example:

```
cxfs_admin:mycluster> create failover_net network=192.168.0.0 mask=255.255.255.0
cxfs_admin:mycluster > create failover_net network=10.0.0.0 mask=255.255.255.0
```

8. Create the switches:

```
cxfs_admin:cluster> create switch name=switch_hostname [vendor=brocade|qlogic]
  [user=username password=password]
```

   For example:

```
cxfs_admin:mycluster> create switch name=myswitch vendor=qlogic
```

9. Create other CXFS nodes as required:

```
cxfs_admin:mycluster> create node name=nodename os=OStype private_net=IPaddress
   [type=server_admin|client_only]
```

   For example, for a server-capable administration node:

```
cxfs_admin:mycluster> create node name=server2 os=Linux private_net=10.11.20.115 \
  type=server_admin
```

   For example, for a client-only node, in this case running Windows:

```
cxfs_admin:mycluster> create node name=client1 os=Windows private_net=10.11.20.116 \
```

10. *(Optional)* Define one of the client-only nodes as the CXFS tiebreaker if using multiple server-capable administration nodes:

```
cxfs_admin:cluster> modify clustername tiebreaker=client_only_nodename
```

    For example:

```
cxfs_admin:mycluster> modify mycluster tiebreaker=client1
```

11. Obtain a shell window for one of the server-capable administration nodes in the cluster and use the Linux `parted`(8) command to create a volume header on the disk drive. For information, see the man page and *Linux Configuration and Operations Guide*.

12. Create the XVM logical volumes. In the shell window, use the `xvm` command line interface. For information, see the *XVM Volume Manager Administrator's Guide*.

13. Make the XFS filesystems. In the shell window, use the `mkfs` command. For information, see the *XVM Volume Manager Administrator's Guide*.

14. Create the CXFS filesystems:

```
cxfs_admin:cluster> create filesystem name=XVMvolume [mountpoint=path]
  [options=mount_options]
```

For example:

```
cxfs_admin:cluster> create filesystem name=cxfsvol1
```

15. *(Optional)* Create private network failover:

```
cxfs_admin:cluster> network=IPaddress mask=NetMask
```

For example, to create two private networks, one on the 192.168.0.*x* and the other on the 10.0.0.*x* subnets:

```
cxfs_admin:mycluster> create failover_net network=192.168.0.0 mask=255.255.255.0
cxfs_admin:mycluster > create failover_net network=10.0.0.0 mask=255.255.255.0
```

16. View the cluster status:

```
cxfs_admin:cluster> status
```

Following is a simple example using prompting mode:

```
cxfsopus14:~ # /usr/cluster/bin/cxfs_admin -s
Connecting to the local CXFS server...
cxfs_admin:(no cluster defined)> create cluster
Specify the attributes for create cluster:
 name? mycluster
cxfs_admin:mycluster> create node
Specify the attributes for create node:
 name? cxfsopus14
 private_net? 10.11.20.114
Node "cxfsopus14" has been created, waiting for it to join the cluster...
Please restart all cxfs and cluster services on the server "cxfsopus14"
to make
it join the cluster.
cxfs_admin:mycluster> create filesystem
```

```
Specify the attributes for create filesystem:
 name? thump
 options?
 forced_unmount? false
 mountpoint? /mnt/thump
 mounted? true
Filesystem "thump" has been created, waiting for it to be mounted on all
assigned nodes...
```

For more information, see Chapter 11, "Reference to `cxfs_admin` Tasks" on page 217 and the `help` command within `cxfs_admin`.

Configuring a Large Cluster

When configuring a large cluster, you should ensure that a small cluster containing just the server-capable administration nodes is fully functional before adding client-only nodes. By building up the cluster with client-only nodes in small groups, you will minimize concurrent operational issues and use the database most efficiently.

Do the following:

1. Create the initial cluster with just the server-capable administration nodes and test it:

   a. Define all of the server-capable administration nodes.

   b. Define the cluster.

   c. Add all of the server-capable administration nodes to the cluster.

   d. Create the filesystems as described in "Set Up a New CXFS Filesystem with the GUI" on page 134.

   e. Verify that the nodes are all part of the cluster membership and that the filesystems are mounted and fully functional.

2. Add the client-only nodes to the database:

   a. Define **all** client-only nodes.

   b. Add **all** client-only nodes to the cluster.

3. Gradually build up the functional cluster with subsets of client-only nodes:

   a. Start CXFS services on a **subset** of four client-only nodes.

   b. Ensure that the nodes are part of the cluster membership and that the filesystems are fully functional.

4. Repeat step 3 as needed to complete the cluster membership.

Following is an example `cxfs_admin` script to configure a cluster. The first `node` line creates the first server-capable administration node; you can copy and repeat the second `node` line for each remaining server-capable or client-only node in the cluster:

```
create cluster name=clustername
create node name=nodename private_net=IPaddress
create node name=nodename os=OS private_net=IPaddress      [copy and repeat]
create filesystem name=filesystemname forced_unmount=false mountpoint=/mnt/nodename mounted=true [copy and repeat]
```

Following is an example for configuring a one-node cluster that can be copied and repeated for the number of nodes required:

```
create cluster name=clustername
create node name=nodename private_net=IPaddress
create filesystem name=filesystemname forced_unmount=false mountpoint=/mnt/nodename
mounted=true
```

Testing the System

This section discusses the following:

- "Private Network Interface" on page 141
- "System Reset Connection for Server-Capable Administration Nodes" on page 142
- "Testing Serial Connectivity for the L2 on Altix 350 Systems" on page 144

Private Network Interface

For each private network on each node in the pool, enter the following, where *nodeIPaddress* is the IP address of the node:

```
node# ping -c 3 nodeIPaddress
```

Typical `ping` output should appear, such as the following:

```
PING IPaddress (190.x.x.x: 56 data bytes
64 bytes from 190.x.x.x: icmp_seq=0 tt1=254 time=3 ms
64 bytes from 190.x.x.x: icmp_seq=1 tt1=254 time=2 ms
64 bytes from 190.x.x.x: icmp_seq=2 tt1=254 time=2 ms
```

If `ping` fails, follow these steps:

1. Verify that the network interface was configured up by using `ifconfig`. For example:

```
node# **ifconfig ec3**
ec3: flags=c63<UP,BROADCAST,NOTRAILERS,RUNNING,FILTMULTI,MULTICAST>
inet 190.x.x.x netmask 0xffffff00 broadcast 190.x.x.x
```

   The `UP` in the first line of output indicates that the interface was configured up.

2. Verify that the cables are correctly seated.

Repeat this procedure on each node.

System Reset Connection for Server-Capable Administration Nodes

To test the system reset connections, do the following:

1. Ensure that the nodes and the serial port multiplexer are powered on.

2. Ensure that the `MODULES_LOADED_ON_BOOT` variable in `/etc/sysconfig/kernel` contains the `ioc4_serial` module. This module must be loaded in order for the devices to be present.

3. Start the `cmgr` command on one of the server-capable administration nodes in the pool:

```
admin# /usr/cluster/bin/cmgr
```

4. Stop CXFS services on the entire cluster:

```
stop cx_services for cluster clustername
```

   For example:

```
cmgr> stop cx_services for cluster cxfs6-8
```

   Wait until the node has successfully transitioned to inactive state and the CXFS processes have exited. This process can take a few minutes.

5. Test the serial connections by entering one of the following:

- To test the whole cluster, enter the following:

  `test serial in cluster` *clustername*

  For example:

```
cmgr> test serial in cluster cxfs6-8
Status: Testing serial lines ...
Status: Checking serial lines using crsd (cluster reset services) from node cxfs8
Success: Serial ping command OK.

Status: Checking serial lines using crsd (cluster reset services) from node cxfs6
Success: Serial ping command OK.

Status: Checking serial lines using crsd (cluster reset services) from node cxfs7
Success: Serial ping command OK.

Notice: overall exit status:success, tests failed:0, total tests executed:1
```

- To test an individual node, enter the following:

  `test serial in cluster` *clustername* `node` *machinename*

  For example:

```
cmgr> test serial in cluster cxfs6-8 node cxfs7
Status: Testing serial lines ...
Status: Checking serial lines using crsd (cluster reset services) from node cxfs6
Success: Serial ping command OK.

Notice: overall exit status:success, tests failed:0, total tests executed:1
```

- To test an individual node using just a `ping`, enter the following:

  `admin ping node` *nodename*

  For example:

```
cmgr> admin ping node cxfs7

ping operation successful
```

6. If a command fails, make sure all the cables are seated properly and rerun the command.
7. Repeat the process on other nodes in the cluster.

Testing Serial Connectivity for the L2 on Altix 350 Systems

You can use the `cu`(1) command to test the serial reset lines if you have installed the `uucp` RPM.

The `cu` command requires that the device files be readable and writable by the user `uucp`. The command also requires the `/var/lock` directory be writable by group `uucp`.

Perform the following steps:

1. Change ownership of the serial devices so that they are in group `uucp` and owned by user `uucp`.

   Note: The ownership change may not be persistent across reboots.

   For example, suppose you have the following TTY devices on the IO10:

   ```
   admin# ls -l /dev/ttyIOC*
   crw-rw----  1 root uucp 204, 50 Sep 15 16:20 /dev/ttyIOC0
   crw-rw----  1 root uucp 204, 51 Sep 15 16:20 /dev/ttyIOC1
   crw-rw----  1 root uucp 204, 52 Sep 15 16:20 /dev/ttyIOC2
   crw-rw----  1 root uucp 204, 53 Sep 15 16:20 /dev/ttyIOC3
   ```

   To change ownership of them to `uucp`, you would enter the following:

   ```
   admin# chown uucp.uucp /dev/ttyIOC*
   ```

2. Determine if group `uucp` can write to the `/var/lock` directory and change permissions if necessary.

   For example, the following shows that group `uucp` cannot write to the directory:

   ```
   admin# ls -ld /var/lock
   drwxr-xr-t  5 root uucp 88 Sep 19 08:21 /var/lock
   ```

The following adds write permission for group `uucp`:

```
admin# chmod g+w /var/lock
```

3. Join the `uucp` group temporarily, if necessary, and use `cu` to test the line.

   For example:

```
admin# newgrp uucp
# cu -l /dev/ttyIOC0 -s 38400
Connected
nodeA-001-L2>cfg
L2 192.168.0.1: - 001 (LOCAL)
L1 192.0.1.133:0:0    - 001c04.1
L1 192.0.1.133:0:1    - 001i13.1
L1 192.0.1.133:0:5    - 001c07.2
L1 192.0.1.133:0:6    - 001i02.2
```

For more information, see the `cu`(1) man page and the documentation that comes with the `uucp` RPM.

Chapter 10

Reference to GUI Tasks

This chapter discusses the CXFS Manager graphical user interface (GUI). It contains detailed information about CXFS tasks and an overview of XVM tasks. (For details about XVM tasks, see the *XVM Volume Manager Administrator's Guide*.)

This chapter contains the following sections:

- "GUI Overview" on page 147
- "Guided Configuration Tasks" on page 169
- "Node Tasks with the GUI" on page 170
- "Cluster Tasks with the GUI" on page 186
- "Cluster Services Tasks with the GUI" on page 188
- "Switches and I/O Fencing Tasks with the GUI" on page 194
- "Filesystem Tasks with the GUI" on page 199
- "Privileges Tasks with the GUI" on page 211

Note: CXFS requires a license key to be installed on each server-capable administration node. If you install the software without properly installing the license key, you will get an error and will not be able to use the CXFS Manager GUI. For more information about licensing, see Chapter 5, "CXFS License Keys" on page 89.

GUI Overview

The GUI lets you set up and administer CXFS filesystems and XVM logical volumes. It also provides icons representing status and structure.

This section provides an overview of the GUI:

- "Starting the GUI"
- "GUI Windows" on page 153

- "GUI Features" on page 155
- "Key to Icons and States" on page 165

Note: CXFS is incompatible with the Red Hat cluster manager available in the Red Hat Advanced Server product.

Starting the GUI

There are several methods to start the GUI and connect to a node.

Starting the GUI on IRIX

To start the GUI, use one of the following methods:

- On an IRIX system where the CXFS GUI-client software (`sysadm_cxfs.sw.client`) and desktop support software (`sysadm_cxfs.sw.desktop`) are installed, do one of the following:

  Note: SGI does not recommend this method across a wide-area network (WAN) or virtual private network (VPN), or if the IRIX system has an R5000 or earlier CPU and less than 128-MB memory.

  – Enter the following command line:

  ```
  irix# /usr/sbin/cxfsmgr
  ```

  – Choose the following from the Toolchest:

  System
      > CXFS Manager

  You must restart the Toolchest after installing CXFS in order to see the **CXFS** entry on the Toolchest display. Enter the following commands to restart the Toolchest:

  ```
  irix# killall toolchest
  irix# /usr/bin/X11/toolchest &
  ```

If you are using WAN or VPN, see "Running the Web-based Version" on page 149.

Starting the GUI on SGI ProPack

To start the GUI on an SGI ProPack system where the CXFS GUI-client software (`cxfs-sysadm_cxfs-client`) is installed, do the following:

1. Obtain and install the J2SE 1.4.2 (latest patch) software available from http://java.sun.com
2. Enter the following command line:

   ```
   propack# /usr/sbin/cxfsmgr
   ```

Running the Web-based Version

If you want to use a web-based version of the GUI, do the following:

1. Ensure that the following software products are installed on the server-capable administration nodes that you will connect to (by means of a Java-enabled web browser running on any platform) for performing administrative operations:

   ```
   cxfs-sysadm_xvm-web
   cxfs-sysadm_cxfs-web
   ```

   These software products are part of the software normally installed with CXFS.
2. Ensure that an apache Web server is installed and running on the server-capable administration node.
3. On a PC, install the Java2 v1.4.2 or v1.5 plug-in.

   On an IRIX machine that launches the GUI client from a web browser that supports Java, install the `java_plugin` software product from the IRIX 6.5.*x* CD. This is the Runtime Plug-in for IRIX, Java Edition 1.4.1, which supports JRE 1.4.1. (However, launching the GUI from a web browser is not the recommended method on IRIX. On IRIX, running the GUI client from the desktop is preferred.)
4. Add the following to your `httpd.conf` file:

   ```
   <Location "/CXFSManager">
             Options Includes ExecCGI FollowSymLinks
             DirectoryIndex index.html index.shtml
   </Location>
   ```
5. Close all browser windows and restart the browser.

6. Enter the URL `http://`*server*`/CXFSManager/` where *server* is the name of a server-capable administration node in the pool
7. At the resulting webpage, click the **CXFS Manager** icon.

Note: This method can be used on IRIX systems, but it is not the preferred method unless you are using WAN or VPN. If you load the GUI using Netscape on IRIX and then switch to another page in Netscape, CXFS Manager GUI will not operate correctly. To avoid this problem, leave the CXFS Manager GUI web page up and open a new Netscape window if you want to view another web page.

Running as a Non-Root User on IRIX

Running the CXFS Manager graphical user interface (GUI) from a login other than `root` requires the `sysadmdesktop` package, which is installed by default when you install IRIX. This package provides commands that allow you to give users privileges, including the privileges required to run the CXFS commands. `sysadmdesktop` (located on the *Applications CD 1 of 2 for 6.5.x*) installs the following software products:

```
sysadmdesktop.man.base
sysadmdesktop.man.relnotes
sysadmdesktop.sw.base
sysadmdesktop.sw.data
sysadmdesktop.sw.sysadm
```

Running the GUI from an IRIX Desktop Outside the Cluster

If you want to run the GUI client from an IRIX desktop outside of the cluster, install the following software products on that machine:

```
java2_eoe.sw
java2_eoe.sw32
sysadm_base.man
sysadm_base.sw.client
sysadm_cluster.sw.client
sysadm_cxfs.man
sysadm_cxfs.sw.client
sysadm_cxfs.sw.desktop
sysadm_xvm.sw.client
sysadm_xvm.sw.desktop
```

⚠ **Caution:** The GUI on IRIX only operates with Java2 v1.4.1 Execution Environment (Sun JRE v1.4.1). This is the version of Java that is provided with the supported IRIX 6.5.*x* release.

The SGI website also contains Java1. However, you cannot use this version of Java with the GUI. Using a Java version other than 1.4.1 will cause the GUI to fail.

Summary of GUI Platforms

Table 10-1 describes the platforms where the GUI may be started, connected to, and displayed.

Table 10-1 GUI Platforms

GUI Mode	Where You Start the GUI	Where You Connect the GUI	Where the GUI Displays
`cxfsmgr`	Any IRIX system (such as an SGI 2000 series or SGI O2 workstation) with `sysadm_cxfs.sw.client` and `sysadm_cxfs.sw.desktop` software installed An SGI ProPack system with `cxfs-sysadm_cxfs-client` installed	The server-capable administration node in the pool that you want to use for cluster administration	The system where the GUI was invoked
Toolchest	Any IRIX system (such as an SGI 2000 series or SGI O2 workstation) with `sysadm_cxfs.sw.client` and `sysadm_cxfs.sw.desktop` software installed	The server-capable administration node in the pool that you want to use for cluster administration	The system where the GUI was invoked
Web	Any system with a web browser and Java2 1.4.1 or 1.4.2 plug-in installed and enabled	The server-capable administration node in the pool that you want to use for cluster administration	The same system with the web browser

Logging In

To ensure that the required GUI privileges are available for performing all of the tasks, you should log in to the GUI as `root`. However, some or all privileges can be granted to any other user using the GUI privilege tasks; see "Privileges Tasks with the GUI" on page 211. (Under IRIX, this functionality is also available with the Privilege Manager, part of the IRIX Interactive Desktop System Administration `sysadmdesktop` product. For more information, see the *Personal System Administration Guide*.)

A dialog box will appear prompting you to log in to a CXFS host. You can choose one of the following connection types:

- **Local** runs the server-side process on the local host instead of going over the network
- **Direct** creates a direct socket connection using the `tcpmux` TCP protocol (`tcpmux` must be enabled)

- **Remote Shell** connects to the server via a user-specified command shell, such as `rsh` or `ssh`. For example:

  ```
  ssh -l root servername
  ```

  Note: For secure connection, choose **Remote Shell** and type a secure connection command using a utility such as `ssh`. Otherwise, the GUI will not encrypt communication and transferred passwords will be visible to users of the network.

- **Proxy** connects to the server through a firewall via a proxy server

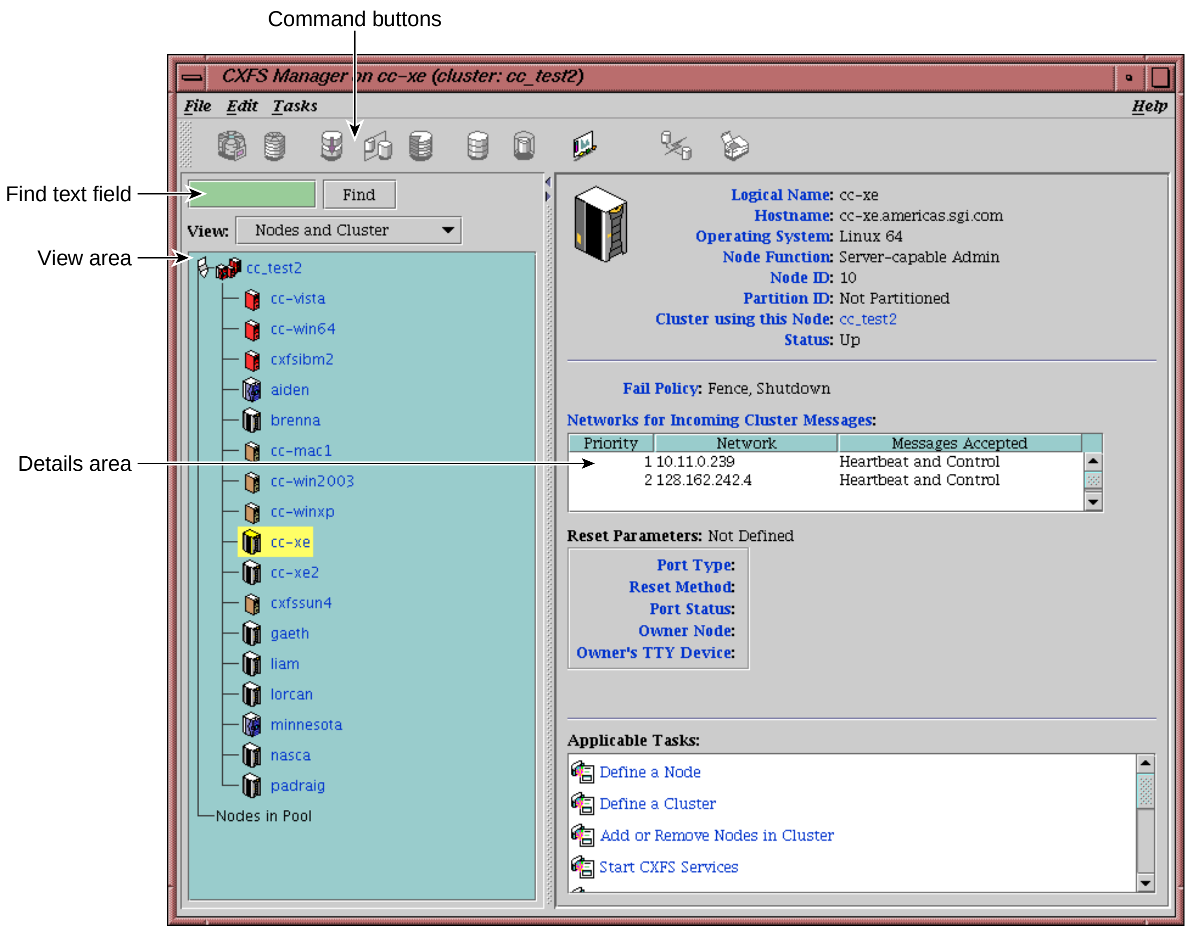
Making Changes Safely

Do not make configuration changes on two different server-capable administration nodes in the pool simultaneously, or use the CXFS GUI, `cxfs_admin`, and `xvm` commands simultaneously to make changes. You should run one instance of the `cxfs_admin` command or the CXFS GUI on a single server-capable administration node in the pool when making changes at any given time. However, you can use any node in the pool when requesting status or configuration information. Multiple CXFS Manager windows accessed via the **File** menu are all part of the same application process; you can make changes from any of these windows.

The server-capable administration node to which you connect the GUI affects your view of the cluster. You should wait for a change to appear in the *view area* before making another change; the change is not guaranteed to be propagated across the cluster until it appears in the view area. (To see the location of the view area, see Figure 10-1 on page 154.) The entire cluster status information is sent to every server-capable administration node each time a change is made to the cluster database.

GUI Windows

Figure 10-1 shows the **CXFS Manager** window displaying information for a specific component in the *details area*. For information about using the *view area* to monitor status and an explanation of the icons and colors, see "Cluster, Node, and CXFS Filesystem Status" on page 343.



Logical Name: cc-xe
Hostname: cc-xe.americas.sgi.com
Operating System: Linux 64
Node Function: Server-capable Admin
Node ID: 10
Partition ID: Not Partitioned
Cluster using this Node: cc_test2
Status: Up

Fail Policy: Fence, Shutdown

Networks for Incoming Cluster Messages:

Priority	Network	Messages Accepted
1	10.11.0.239	Heartbeat and Control
2	128.162.242.4	Heartbeat and Control

Reset Parameters: Not Defined

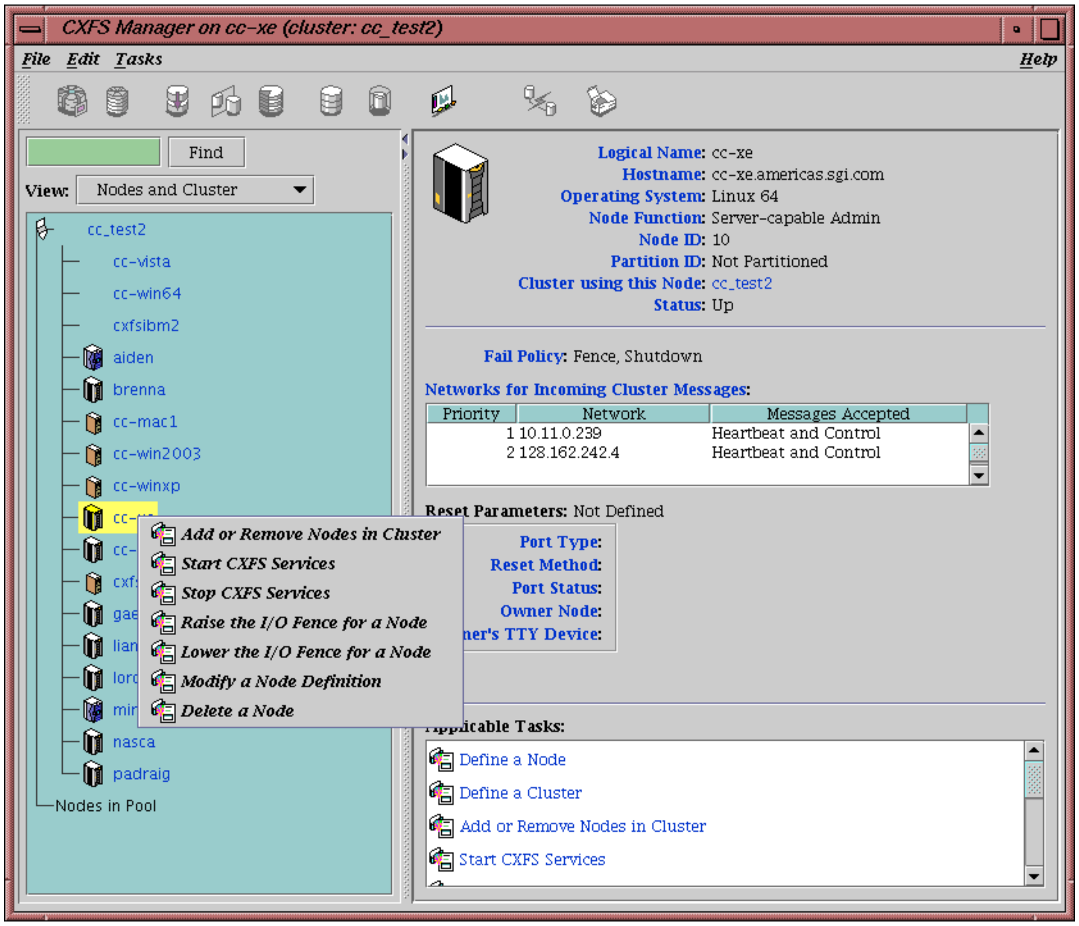
Port Type:
Reset Method:
Port Status:
Owner Node:
Owner's TTY Device:

Applicable Tasks:

- Define a Node
- Define a Cluster
- Add or Remove Nodes in Cluster
- Start CXFS Services

Figure 10-1 CXFS Manager GUI Showing Details for a Node

Figure 10-2 shows an example of the pop-up menu of applicable tasks that appears when you click the right mouse button on a selected item; in this example, clicking on the node name `trinity` displays a list of applicable tasks.



Logical Name: cc-xe
Hostname: cc-xe.americas.sgi.com
Operating System: Linux 64
Node Function: Server-capable Admin
Node ID: 10
Partition ID: Not Partitioned
Cluster using this Node: cc_test2
Status: Up

Fail Policy: Fence, Shutdown

Networks for Incoming Cluster Messages:

Priority	Network	Messages Accepted
1	10.11.0.239	Heartbeat and Control
2	128.162.242.4	Heartbeat and Control

Reset Parameters: Not Defined

Port Type:
Reset Method:
Port Status:
Owner Node:
ner's TTY Device:

Figure 10-2 Pop-up Menu that Appears After Clicking the Right Mouse Button

GUI Features

The **CXFS Manager** GUI allows you to administer the entire CXFS cluster from a single point. It provides access to the tools that help you set up and administer your CXFS cluster:

- *Tasks* let you set up and monitor individual components of a CXFS cluster, including XVM volumes. For details about XVM tasks, see *XVM Volume Manager Administrator's Guide*.

- *Guided configuration tasks* consist of a group of tasks collected together to accomplish a larger goal. For example, **Set Up a New Cluster** steps you through the process for creating a new cluster and allows you to launch the necessary individual tasks by clicking their titles.

This section discusses the following:

- "GUI Window Layout" on page 156
- "File Menu" on page 157
- "Edit Menu" on page 157
- "Tasks Menu" on page 157
- "Help Menu" on page 158
- "Shortcuts Using Command Buttons" on page 158
- "View Menu" on page 160
- "Performing Tasks" on page 161
- "Using Drag-and-Drop" on page 162
- "Structuring Volume Topologies" on page 162
- "Configuring Disks" on page 163
- "Getting More Information" on page 164
- "Important GUI and `xvm` Command Differences" on page 164

GUI Window Layout

By default, the window is divided into two sections: the *view area* and the *details area* (see Figure 10-1 on page 154). The details area shows generic overview text if no item is selected in the view area. You can use the arrows in the middle of the window to shift the display.

File Menu

The **File** menu lets you display the following:

- Multiple windows for this instance of the GUI
- System log file:`/var/log/messages`
- System administration log file: `/var/lib/sysadm/salog`

  The `salog` file shows the commands run directly by this instance of the GUI or some other instance of the GUI running commands on the system. (Changes should not be made simultaneously by multiple instances of the GUI or the GUI and `cxfs_admin`.)

The **File** menu also lets you close the current window and exit the GUI completely.

Edit Menu

The **Edit** menu lets you expand and collapse the contents of the view area. You can choose to automatically expand the display to reflect new nodes added to the pool or cluster. You can also use this menu to select all items in the view menu or clear the current selections.

Tasks Menu

The **Tasks** menu contains the following:

- **Guided Configuration**, which contains the tasks to set up your cluster, define filesystems, create volumes, check status, and modify an existing cluster
- **Nodes**, which contains tasks to define and manage the nodes
- **Cluster**, which contains tasks to define and manage the cluster
- **Cluster Services**, which allows you to start and stop CXFS services, set the CXFS tiebreaker node, set the log configuration, and revoke or allow CXFS kernel membership of the local node
- **Switches and I/O Fencing**, which contains tasks to configure switch definitions and manage I/O fencing
- **Disks**, which contains XVM disk administration tasks

- **Volume Elements**, which contains tasks to create, delete, modify, and administer XVM volume elements
- **Filesystems**, which contains tasks to define and manage filesystems and relocate a metadata server
- **Privileges**, which lets you grant or revoke access to a specific task for one or more users
- **Find Tasks**, which lets you use keywords to search for a specific task

Help Menu

The **Help** menu provides an overview of the GUI and a key to the icons. You can also get help for certain items in blue text by clicking on them.

Shortcuts Using Command Buttons

The command buttons along the top of the GUI window provide a method of performing tasks quickly. When you click a button, the corresponding task executes using default values, usually without displaying a task window. To override the defaults, launch the task from the **Tasks** menu. Table 10-2 summarizes the shortcuts available; for details about these tasks, see the *XVM Volume Manager Administrator's Guide*.

Table 10-2 Command Buttons

Button	Task
	Labels selected unlabeled disks. If the selected disks include foreign and/or labeled disks, the **Label Disks** task will be run.
	Brings up the **Slice Disk** task with the selected disks as default inputs

Button	Task
	Creates a concat with a temporary name
	Creates a mirror with a temporary name
	Creates a stripe with a temporary name
	Creates a volume with a temporary name
	Creates a subvolume with a temporary name
	Detaches the selected volume elements from their current parents
	Deletes the selected non-slice volume elements or unlabels the selected disks directly, or brings up the appropriate delete task for the selected component

View Menu

Choose what you want to view from the **View** menu:

- Nodes and cluster
- Filesystems
- Cluster volume elements
- Local volume elements
- Disks
- Switches
- Users
- Task privileges

Selecting Items to View or Modify

You can use the following methods to select items:

- Click to select one item at a time
- `Shift`+click to select a block of items
- `Ctrl`+click to toggle the selection of any one item

Another way to select one or more items is to type a name into the **Find** text field and then press `Enter` or click the **Find** button.

Viewing Component Details

To view the details on any component, click its name in the view area; see "Selecting Items to View or Modify" on page 160.

The configuration and status details for the component will appear in the details area to the right. At the bottom of the details area will be the **Applicable Tasks** list, which displays tasks you may wish to launch after evaluating the component's configuration details. To launch a task, click the task name; based on the component selected, default values will appear in the task window.

To see more information about an item in the details area, select its name (which will appear in blue); details will appear in a new window. Terms with glossary definitions also appear in blue.

Performing Tasks

To perform an individual task, do the following:

1. Select the task name from the **Task** menu or click the right mouse button within the view area. For example:

   Task
   > Guided Configuration
   > Set Up a New Cluster

   The task window appears.

   As a shortcut, you can right-click an item in the view area to bring up a list of tasks applicable to that item; information will also be displayed in the details area.

   Note: You can click any blue text to get more information about that concept or input field.

2. Enter information in the appropriate fields and click **OK** to complete the task. (Some tasks consist of more than one page; in these cases, click **Next** to go to the next page, complete the information there, and then click **OK**.)

   Note: In every task, the cluster configuration will not update until you click **OK**.

   A dialog box appears confirming the successful completion of the task.

3. Continue launching tasks as needed.

Using Drag-and-Drop

The GUI lets you use drag-and-drop to do the following:

- Move nodes between the pool and the cluster
- Structure volume topologies
- Administer XVM disks

Caution: Always exercise care when restructuring volume elements with drag-and-drop because data that resides on the volume element can be lost. The GUI attempts to warn the user when it can predict that there is a high likelihood of data loss. However, when a volume is not associated with a mounted filesystem, neither the `xvm` command nor the GUI can determine whether that volume holds important data.

To select multiple GUI icons, select the first icon by clicking the left mouse button, then press the `Ctrl` button while clicking on the additional icons. To select consecutive icons, select the first icon and press shift while selecting the last icon.

You cannot drag and drop between two GUI windows. You cannot drag and drop between the CXFS Manager and the IRIX Interactive Desktop Personal System Administration windows. You cannot drag and drop items onto shortcut command buttons.

See the *XVM Volume Manager Administrator's Guide* for more information about using drag-and-drop to structure volume topologies and configure disks.

Structuring Volume Topologies

To reconfigure a logical volume, do the following:

- Select the view you want:

  View
  > Cluster Volume Elements

  or

View
  > Local Volume Elements

- Select a volume element icon
- Drag the icon and drop it on another volume element icon

Icons turn blue as you drag to indicate when it is valid to drop upon them. When you drag, if the mouse cursor reaches the top or the bottom of the view area, the display will scroll automatically.

You can use drag-and-drop to operate on multiple volume elements of different types. For example, you can detach several types of volume elements by selecting items and dragging them to any **Unattached** heading, even if no selected item belongs to that category. You can select multiple items of different types and attach them to a parent. For example, you can select two concats and a stripe and use drag-and-drop to attach them to a parent concat.

You can rename volume elements by clicking a selected (highlighted) volume element and typing a new name into the text field.

Configuring Disks

To label or unlabel disks using drag-and-drop, select the following:

View
  > Disks

Select an unlabeled disk then drag and drop it on the **Labeled Disks** heading, or select a labeled disk then drag and drop it on the **Unlabeled Disks** heading.

You can give away a disk using the task menu or drag-and-drop. In the **Disks** view, select a disk and then drag and drop it on the **Cluster Disks** heading.

Note: Giving away a disk presents less risk of data loss than stealing a disk.

You can label a disk by clicking a selected (highlighted) disk and typing a name into the resulting name text field.

For more information, see the *XVM Volume Manager Administrator's Guide*.

Displaying State

The GUI shows the static and dynamic state of the cluster. For example, suppose the database contains the static information that a filesystem is enabled for mount; the GUI will display the dynamic information showing one of the following:

- A blue icon indicating that the filesystem is mounted (the static and dynamic states match).
- A grey icon indicating that the filesystem is configured to be mounted but the procedure cannot complete because CXFS services have not been started (the static and dynamic states do not match, but this is expected under the current circumstances). See "CXFS Services" on page 28.
- An error (red) icon indicating that the filesystem is supposed to be mounted (CXFS services have been started), but it is not (the static and dynamic states do not match, and there is a problem).

Getting More Information

Click blue text to launch tasks or display one of the following:

- Term definitions
- Input instructions
- Item details
- The selected task window

Important GUI and `xvm` Command Differences

When volume elements other than volumes are created or detached, the system automatically creates a volume and a subvolume that are associated with the volume element. You can explicitly name this generated volume, in which case the volume name is stored in label space and persists across machine reboots.

The GUI does not display volumes and subvolumes that were not named explicitly. The GUI displays the children of these volumes and subvolumes as available for use or as unattached. In contrast, the `xvm` command shows all volumes and subvolumes.

The GUI displays filesystems that are on volumes that were not named explicitly, but lists the volumes as **None**. Volumes and subvolumes that the system generated automatically with temporary names are mentioned in the full paths of unattached volume elements (for example, `/vol96/datav`), but the GUI ignores them otherwise.

To reduce the risk of data loss, SGI recommends that you name volumes explicitly when using the GUI. If you have created volumes using the `xvm` command that you did not name explicitly, you can use the `xvm` tool to assign these volumes permanent names before proceeding. This can reduce the risk of data loss.

Key to Icons and States

The following tables show keys to the icons and states used in the CXFS Manager GUI.

Table 10-3 Key to Icons

Icon	Entity
	IRIX node
	SGI ProPack node (server-capable or client-only)
	AIX, Linux third-party, Mac OS X, Solaris, or Windows node (client-only)
	Cluster
	Expanded tree in view area

Icon	Entity
	Collapsed tree in view area
	Switch
	XVM disk
	Unlabeled disk
	Foreign disk
	Slice
	Volume
	Subvolume
	Concat

Icon	Entity
	Mirror
	Stripe
	Slot
	Local filesystem
	CXFS filesystem
	Copy on write
	Repository
	Snapshot
	User account

Icon	Entity
	GUI task for which execution privilege may be granted or revoked
🔑	Privileged command executed by a given GUI task

Table 10-4 Key to States

Icon	State
	(grey icon) Inactive, unknown, offline — CXFS services may not be active
	(blue icon) Enabled for mount — CXFS services may not be active
	(blue icon) Online, ready for use, up, or mounted without error
	(green swatch) Open, in use

Icon	State
	(blinking orange arrow) Mirror reviving
	(red icon) Error detected, down or mounted with error

Guided Configuration Tasks

This section discusses the following guided configuration tasks:

- "Make Changes to Existing Cluster" on page 169
- "Fix or Upgrade Cluster Nodes" on page 170

Also see "Set Up a New Cluster with the GUI" on page 133, "Set Up a New CXFS Filesystem with the GUI" on page 134, and "CXFS GUI and Status" on page 343. For information about XVM guided configuration tasks, see the *XVM Volume Manager Administrator's Guide*.

Make Changes to Existing Cluster

This task lists different ways to edit an existing cluster. You can make changes while the CXFS services are active, such as changing the way the cluster administrator is notified of events; however, your must first stop CXFS services before testing connectivity. You must unmount a filesystem before making changes to it.

See the following:

- "Modify a Cluster Definition with the GUI" on page 187
- "Set Up a New CXFS Filesystem with the GUI" on page 134
- "Modify a CXFS Filesystem with the GUI" on page 206
- "Define a Node with the GUI" on page 171

- "Test Node Connectivity with the GUI" on page 185
- "Add or Remove Nodes in the Cluster with the GUI" on page 180

Fix or Upgrade Cluster Nodes

This task leads you through the steps required to remove a server-capable administration node from a cluster. It covers the following steps:

- "Stop CXFS Services with the GUI" on page 189.
- Perform the necessary maintenance on the node. Only if required, see "Reset a Node with the GUI " on page 180.
- "Start CXFS Services with the GUI" on page 189.
- Monitor the state of the cluster components in the view area. See "CXFS GUI and Status" on page 343.

When shutting down, resetting, or restarting a CXFS client-only node, do not stop CXFS services on the node. (Stopping CXFS services is more intrusive on other nodes in the cluster because it updates the cluster database. Stopping CXFS services is appropriate only for a server-capable administration node.) Rather, let the CXFS shutdown scripts on the node stop CXFS when the client-only node is shut down or restarted.

Node Tasks with the GUI

This section discusses the following:

- "Define a Node with the GUI" on page 171
- "Examples of Defining a Node with the GUI" on page 177
- "Add or Remove Nodes in the Cluster with the GUI" on page 180
- "Reset a Node with the GUI " on page 180
- "Modify a Node Definition with the GUI" on page 181
- "Delete a Node with the GUI" on page 185
- "Test Node Connectivity with the GUI" on page 185

- "Display a Node with the GUI" on page 185

Note: The **Set Up a New Cluster** guided configuration task leads you through the tasks required to set up the cluster and nodes. See "Set Up a New Cluster with the GUI" on page 133.

Define a Node with the GUI

Note: Within the CXFS tasks, you can click any blue text to get more information about that concept or input field. In every task, the cluster configuration will not update until you click **OK**.

To define a node, do the following:

1. **Hostname**: Enter the hostname of the node you are defining. You can use a simple hostname, such as `lilly`, if it can be resolved by the name server or `/etc/hosts` on all nodes in the cluster; otherwise, use a fully qualified domain name such as `lilly.example.com`. Use the `ping` command to display the fully qualified hostname. Do not enter an IP address.

   If you attempt to define a cluster or other object before the local node has been defined, you will get an error message that says:

   ```
   No nodes are registered on servername. You cannot define a cluster
   until you define the node to which the GUI is connected. To do so,
   click "Continue" to launch the "Set Up a New Cluster" task.
   ```

2. **Logical Name**: Enter the simple hostname (such as `lilly`) or an entirely different name (such as `nodeA`). If you entered in the simple hostname for the **Hostname** field, the same name will be entered into the **Logical Name** field by default. Logical names cannot begin with an underscore (_) or include any whitespace characters, and can be at most 255 characters.

   Note: To rename a node, you must delete it and then define a new node.

   ---

3. **Operating System:** Choose the name of the operating system that is running on the node being defined. Choose **Windows** for Windows 2000, Windows 2003, or Windows XP. Choose **Linux 64** when defining an x86_64 or ia64 architecture. (Use the `uname -i` command to determine the architecture type.)

An SGI ProPack node can be a server-capable administration node or a CXFS client-only node, depending upon the node function selected and the software installed. AIX, IRIX, Linux third-party, Mac OS X, Solaris, and Windows nodes are always CXFS client-only nodes.

If you select a fail policy that includes reset, you will be given an opportunity to provide reset information on a second page. Any potential metadata server should include reset in its fail policy hierarchy.

You cannot later modify the operating system for a defined node. To change the operating system, you would have to delete the node and then define a new node with the new name.

4. **Node Function**: Select one of the following:

   - **Server-capable Admin** is an SGI ProPack node on which you will execute cluster administration commands and that you also want to be a CXFS metadata server. (You will use the **Define a CXFS Filesystem** task to define the specific filesystem for which this node can be a metadata servers.) Use this node function only if the node will be a metadata servers. You must install the `cluster_admin` product on this node.

   - **Client-only** is a node that shares CXFS filesystems but on which you will not execute cluster administration commands and that will not be a CXFS metadata server. Use this node function for all nodes other than those that will be metadata servers. You must install the product on this node. This node can run on any platform. (Nodes other than SGI ProPack are **required** to be client-only nodes.)

5. **Networks for Incoming Cluster Messages**: Do the following:

   - **Network**: Enter the IP address or hostname of the NIC. (The hostname must be resolved in the `/etc/hosts` file.) The priorities of the NICs must be the same for each node in the cluster. For information about why a private network is required, see "Private Network" on page 22.
   - **Messages to Accept**: Select **Heartbeat and Control**.

     You can use the **None** setting if you want to temporarily define a NIC but do not want it to accept messages.
   - Click **Add** to add the NIC to the list.

     If you later want to modify the NIC, click the NIC in the list to select it, then click **Modify**.

     To delete a NIC from the list, click the NIC in the list to select it, then click **Delete**.

   By default, the priority `1` NICs are used as the private network; they must be on the same subnet. To allow one network to fail over to another, you must group the NICs into failover networks manually by using `cxfs_admin`. See Chapter 11, "Reference to `cxfs_admin` Tasks" on page 217.

6. **Node ID**: (*Optional for server-capable administration nodes*) An integer in the range 1 through 32767 that is unique among the nodes in the pool. If you do not specify a number for a server-capable administration node, CXFS will calculate an ID for you.

   For server-capable administration nodes, the default ID is a 5-digit number based on the machine's serial number and other machine-specific information; it is not sequential. For client-only nodes, you must supply the node ID.

   You must not change the node ID number after the node has been defined. (There is no default CXFS tiebreaker; for more information, see "CXFS Kernel Membership, Quorum, and Tiebreaker" on page 432.)

7. **Partition ID**: *(Optional)* Uniquely defines a partition in a partitioned Origin 3000 system, Altix 3000 series system, or Altix 4700 system. If your system is not partitioned, leave this field empty. Use the IRIX `mkpart` command or the SGI ProPack `proc` command to determine the partition ID value (see below).

   Click **Next** to move to the next screen.

8. **Fail Policy:** Specify the set of actions that determines what happens to a failed node: the second action will be followed only if the first action fails; the third action will be followed only if the first and second fail.

   The available actions depend upon the operating system value selected for the node:

   - **Fence:** disables access to the SAN from the problem node. Fencing provides faster recovery of the CXFS kernel membership than reset.
   - **FenceReset:** performs a fence and then, if the node is successfully fenced, **also** performs an asynchronous reset of the node via a system controller (according to the chosen reset method); recovery begins without waiting for reset acknowledgement.

     Note: A server-capable administration node should also include **Reset** in its fail policy hierarchy (unless it is the only server-capable administration node in the cluster).

   - **Reset:** performs a system reset via a system controller. A server-capable administration node should include **Reset** in its fail policy hierarchy.
   - **Shutdown:** tells the other nodes in the cluster to wait for a period of time (long enough for the node to shut itself down) before reforming the CXFS kernel membership. (However, there is no notification that the node's shutdown has actually taken place.) The default fail policy hierarchy for IRIX or SGI ProPack nodes is **Reset, Shutdown**. The default for other nodes is **Shutdown**.

     Caution: There are issues when using **Shutdown** with server-capable administration nodes; for more information and for a list of valid fail policy sets, see "Data Integrity Protection" on page 24. If you are using dynamic CXFS kernel heartbeat monitoring, you must not use the **Shutdown** setting on a client-only node. For information about heartbeat monitoring, see "Use the Appropriate CXFS Kernel Heartbeat Monitoring" on page 63. To specify a fail policy without **Shutdown** you must define or modify the node with `cxfs_admin`. See Chapter 11, "Reference to `cxfs_admin` Tasks" on page 217.

Note: If the failure hierarchy contains **Reset** or **FenceReset**, the reset might be performed before the system kernel core-dump can complete, resulting in an incomplete core-dump.

---

9. If you have chosen a failure hierarchy that includes **Reset** or **FenceReset**, provide the following information.

   - This node:

     – **Port Type:** select one of the following:

       - **L1** (Origin/Onyx 300/350, Origin/Onyx 3200C)
       - **L2** (Any Altix with an L2, Prism, Origin/Onyx 3000 series, Origin 300/350 over a direct-connect serial line)
       - **BMC** (Altix XE systems
       - **MSC** (Origin 200, Onyx2 Deskside, SGI 2100/2200 deskside systems)
       - **MMSC** (Rackmount SGI 2400/2800, Onyx2).

       Note: Altix XE systems use baseboard management controller (BMC) for reset. To configure reset via BMC or L2 over the network, use the `cxfs_admin` configuration tool. See "Create or Modify a Node with `cxfs_admin`" on page 235.

     – **Reset Method**: The type of reset to be performed:

       - **Power Cycle** shuts off power to the node and then restarts it
       - **Reset** simulates the pressing of the reset button on the front of the machine
       - **NMI** (nonmaskable interrupt) performs a core-dump of the operating system kernel, which may be useful when debugging a faulty machine

         ---

         Note: NMI is not available on systems containing a BMC.

     – **Port Password**: The password for the system controller port, **not** the node's `root` password or PROM password. On some nodes, the system administrator may not have set this password. If you wish to set or change the system controller port password, consult the hardware manual for your node.

  – **Temporarily Disable Port**: If you want to provide reset information now but do not want to allow the reset capability at this time, check this box. If this box is checked, CXFS cannot reset the node.

- Owner (node that sends the reset command):

  – **Logical Name**: Name of the node that sends the reset command. If you use serial cables, they must physically connect the node being defined and the owner node through the system controller port. At run time, the node must be defined in the CXFS pool.

    You can select a logical name from the pull-down list or enter the logical name of a node that is not yet defined. However, you must define the node in CXFS before you run the node connectivity diagnostics task or start CXFS services.

  – **TTY Device**: Name of the terminal port (TTY) on the owner node to which the system controller is connected (the node being reset). `/dev/ttyd2` is the most commonly used port, except on Origin 300 and Origin 350 systems (where `/dev/ttyd4` is commonly used) and Altix 350 systems (where `/dev/ttyIOC0` is commonly used). The other end of the cable connects to this node's (the node being reset) system controller port, so the node can be controlled remotely by the owner node. Check the owner node's specific hardware configuration to verify which tty device to use.

    Note: To specify reset method that uses Intelligent Platform Management Interface (IPMI) or L2 over the network, use the `cxfs_admin` configuration tool.See "Create or Modify a Node with `cxfs_admin`" on page 235.

10. Click **OK**.

Note: Do not add a second node until the first node icon appears in the view area. The entire cluster status information is sent to each server-capable administration node each time a change is made to the cluster database; therefore, the more server-capable administration nodes in a configuration, the longer it will take.

You can use the IRIX `mkpart` command to determine the partition ID:

- The `-n` option lists the partition ID (which is 0 if the system is not partitioned).
- The `-l` option lists the bricks in the various partitions (use *rack#.slot#* format in the GUI).

  On SGI ProPack, you can find the partition ID by reading the `proc` file. For example:

  ```
  propack# cat /proc/sgi_sn/partition_id
  0
  ```

  The `0` indicates that the system is not partitioned. If the system is partitioned, the number of partitions (such as `1`, `2`, etc.) is displayed.

For example (output truncated here for readability):

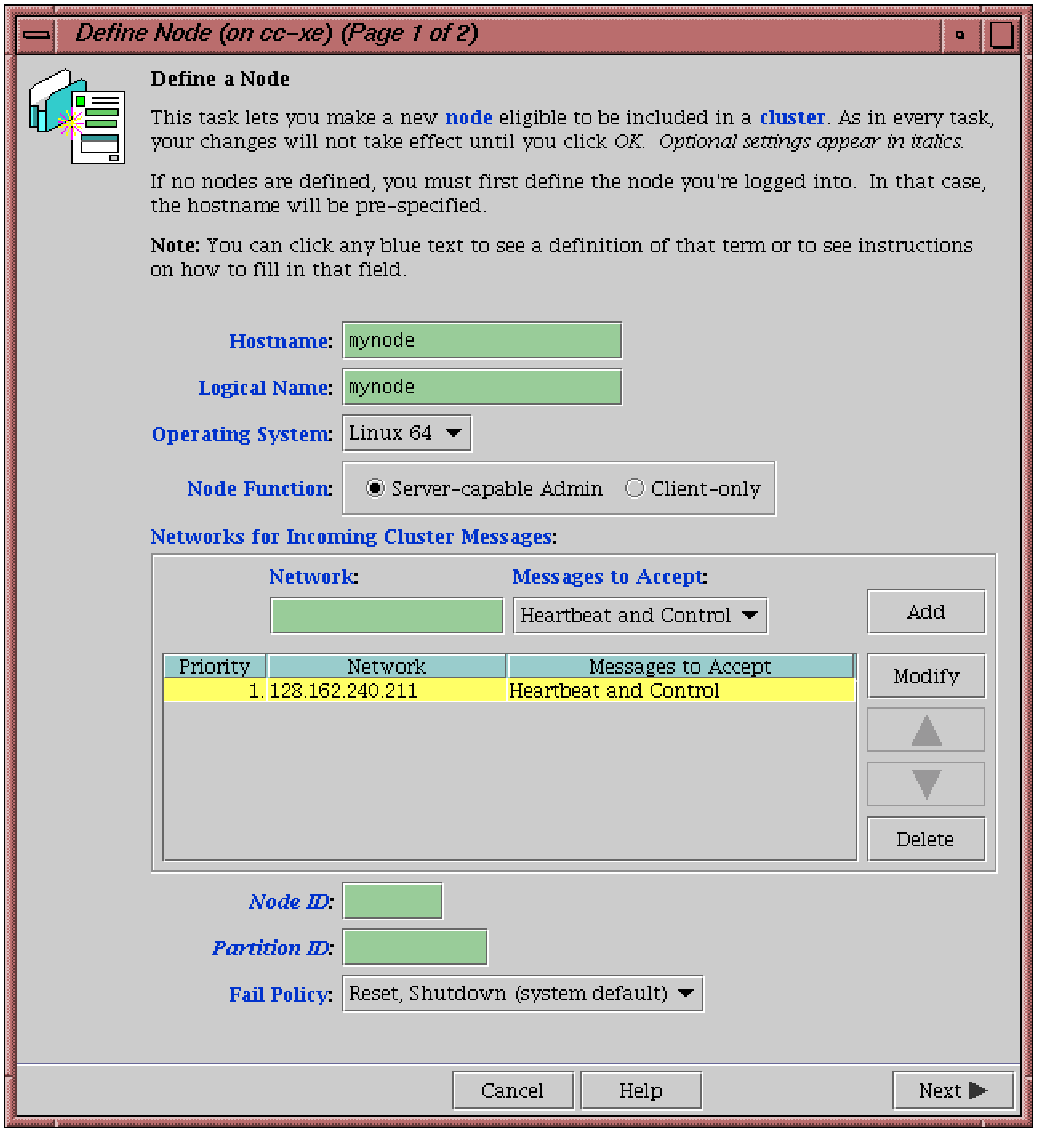
```
# mkpart -n
Partition id = 1
irix# mkpart -l
partition: 3 = brick: 003c10 003c13 003c16 003c21 ...
partition: 1 = brick: 001c10 001c13 001c16 001c21 ...
```

You could enter one of the following for the **Partition ID** field:

```
1
001.10
```

Examples of Defining a Node with the GUI

The following figures show an example of defining a new node.



Define a Node

This task lets you make a new **node** eligible to be included in a **cluster**. As in every task, your changes will not take effect until you click *OK*. *Optional settings appear in italics.*

If no nodes are defined, you must first define the node you're logged into. In that case, the hostname will be pre-specified.

Note: You can click any blue text to see a definition of that term or to see instructions on how to fill in that field.

Hostname: mynode

Logical Name: mynode

Operating System: Linux 64

Node Function: [✓ Server-capable Admin] Client-only

Networks for Incoming Cluster Messages:

Network:

Messages to Accept: Heartbeat and Control

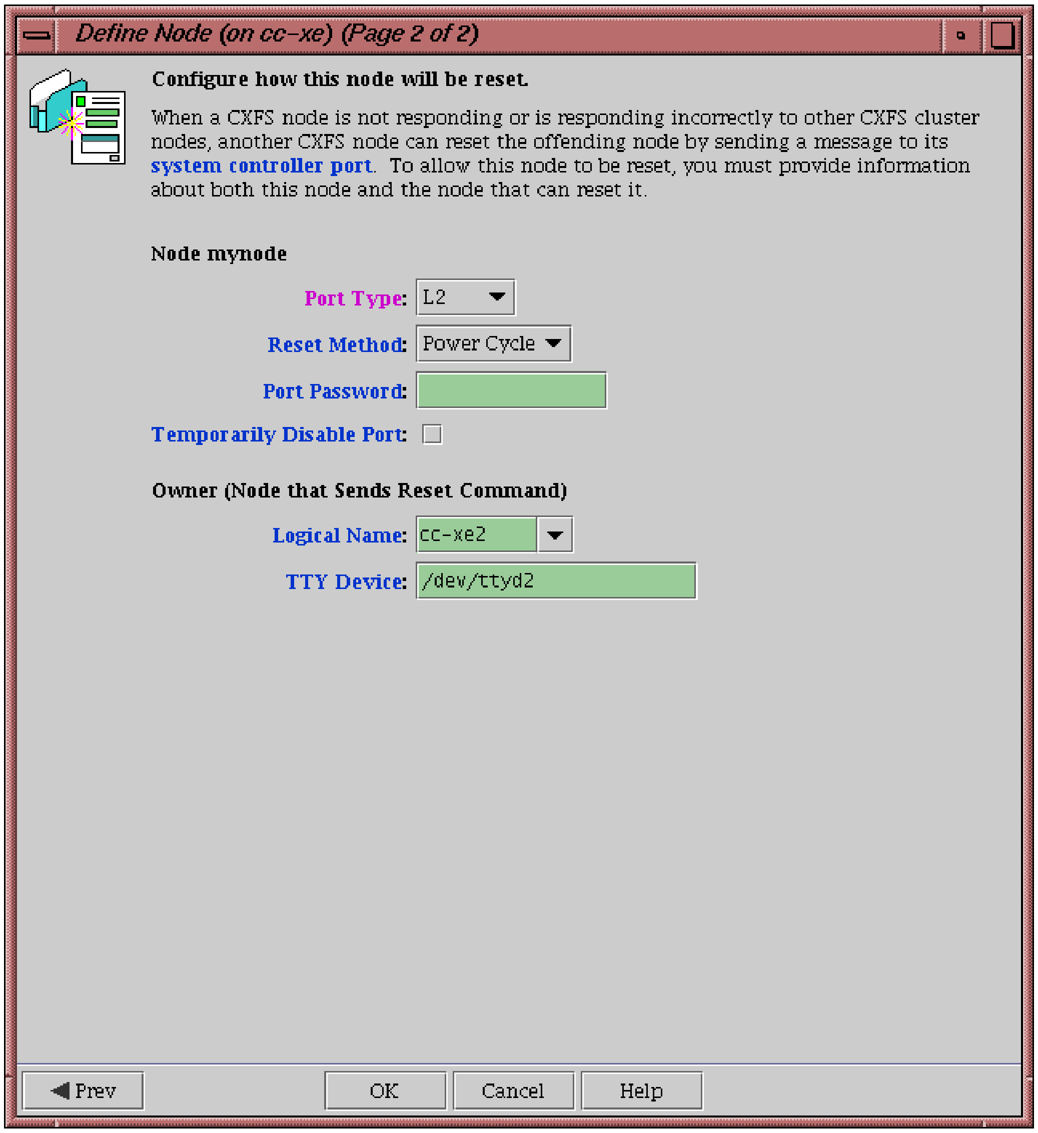
Priority	Network	Messages to Accept
1.	128.162.240.211	Heartbeat and Control

Node ID:

Partition ID:

Fail Policy: Reset, Shutdown (system default)

Figure 10-3 Example Node Definition



Configure how this node will be reset.

When a CXFS node is not responding or is responding incorrectly to other CXFS cluster nodes, another CXFS node can reset the offending node by sending a message to its system controller port. To allow this node to be reset, you must provide information about both this node and the node that can reset it.

Node mynode

Port Type: L2

Reset Method: Power Cycle

Port Password:

Temporarily Disable Port: ☐

Owner (Node that Sends Reset Command)

Logical Name: cc-xe2

TTY Device: /dev/ttyd2

Figure 10-4 Example System Reset Settings

Add or Remove Nodes in the Cluster with the GUI

After you have added nodes to the pool and defined the cluster, you can indicate which nodes to include in the cluster.

Note: Do not add or remove nodes until the cluster icon appears in the view area; set the **View** selection to **Nodes and Cluster**.

Do the following:

1. Add or remove the desired nodes:
   - To add a node, select its logical name from the **Available Nodes** pull-down menu and click **Add**. The node name will appear in the **Nodes to Go into Cluster** list. To select all of the available nodes, click **Add All**.
   - To delete a node, click its logical name in the **Nodes to Go into Cluster** screen. (The logical name will be highlighted.) Then click **Remove**.
2. Click **OK**.

Reset a Node with the GUI

You can use the GUI to reset IRIX or SGI ProPack nodes in a cluster. This sends a reset command to the system controller port on the specified node. When the node is reset, other nodes in the cluster will detect the change and remove the node from the active cluster. When the node reboots, it will rejoin the CXFS kernel membership.

To reset a node, do the following:

1. **Node to Reset:** Choose the node to be reset from the pull-down list.
2. Click **OK**.

Modify a Node Definition with the GUI

To rename a node or change its operating system, you must delete it and then define a new node.

To modify other information about a node, do the following:

1. **Logical Name**: Choose the logical name of the node from the pull-down list. After you do this, information for this node will be filled into the various fields.

2. **Networks for Incoming Cluster Messages**: The priorities of the NICs must be the same for each node in the cluster.

   - **Network**: To add a NIC for incoming cluster messages, enter the IP address or hostname into the **Network** text field and click **Add**.

   - To modify a NIC that is already in the list, click the network in the list in order to select it. Then click **Modify**. This moves the NIC out of the list and into the text entry area. You can then change it. To add it back into the list, click **Add**.

   - To delete a NIC, click the NIC in the priority list in order to select it. Then click **Delete**.

   - To change the priority of a NIC, click the NIC in the priority list in order to select it. Then click the up and down arrows in order to move it to a different position in the list.

     You can use the **None** setting if you want to temporarily define a NIC but do not want it to accept messages.

   By default, the priority `1` NICs are used as the private network; they must be on the same subnet. To allow the one network to fail over to another, you must group the NICs into networks manually by using `cxfs_admin`. See Chapter 11, "Reference to `cxfs_admin` Tasks" on page 217.

   Click **Next** to move to the next page.

3. **Partition ID**: *(Optional)* Uniquely defines a partition in a partitioned Origin 3000 system, Altix 3000 series system, or Altix 4700 system. If your system is not partitioned, leave this field empty. You can use the IRIX `mkpart` command or the SGI ProPack `proc` command to determine the partition ID value; see below.

4. **Fail Policy:** Specify the set of actions that determines what happens to a failed node: the second action will be followed only if the first action fails; the third action will be followed only if the first and second fail.

The available actions depend upon the operating system value selected for the node:

- **Fence:** disables access to the SAN from the problem node. Fencing provides faster recovery of the CXFS kernel membership than reset.
- **FenceReset:** performs a fence and then, if the node is successfully fenced, **also** performs an asynchronous reset of the node via a system controller (according to the chosen reset method); recovery begins without waiting for reset acknowledgement.

  Note: A server-capable administration node should also include **Reset** in its fail policy hierarchy (unless it is the only server-capable administration node in the cluster).

- **Reset:** performs a system reset via a system controller. A server-capable administration node should include **Reset** in its fail policy hierarchy.
- **Shutdown:** tells the other nodes in the cluster to wait for a period of time (long enough for the node to shut itself down) before reforming the CXFS kernel membership. (However, there is no notification that the node's shutdown has actually taken place.) The default fail policy hierarchy for IRIX or SGI ProPack nodes is **Reset, Shutdown**. The default for other nodes is **Shutdown**.

  Caution: There are issues when using **Shutdown** with server-capable administration nodes; for more information and for a list of valid fail policy sets, see "Data Integrity Protection" on page 24. If you are using dynamic CXFS kernel heartbeat monitoring, you must not use the **Shutdown** setting on a client-only node. For information about heartbeat monitoring, see "Use the Appropriate CXFS Kernel Heartbeat Monitoring" on page 63. To specify a fail policy without **Shutdown** you must define or modify the node with `cxfs_admin`. See Chapter 11, "Reference to `cxfs_admin` Tasks" on page 217.

  ---

5. If you have chosen a failure hierarchy that includes **Reset** or **FenceReset**, provide the following information.

   - This node:

     – **Port Type:** select one of the following:

       - **L1** (Origin/Onyx 300/350, Origin/Onyx 3200C)

    - **L2** (Any Altix with an L2, Prism, Origin/Onyx 3000 series, Origin 300/350 over a direct-connect serial line)
    - **BMC** (Altix XE systems
    - **MSC** (Origin 200, Onyx2 Deskside, SGI 2100/2200 deskside systems)
    - **MMSC** (Rackmount SGI 2400/2800, Onyx2).
  - **Reset Method**: The type of reset to be performed:
    - **Power Cycle** shuts off power to the node and then restarts it
    - **Reset** simulates the pressing of the reset button on the front of the machine
    - **NMI** (nonmaskable interrupt) performs a core-dump of the operating system kernel, which may be useful when debugging a faulty machine

      Note: NMI is not available on systems containing a BMC.
  - **Port Password**: The password for the system controller port, **not** the node's `root` password or PROM password. On some nodes, the system administrator may not have set this password. If you wish to set or change the system controller port password, consult the hardware manual for your node.
  - **Temporarily Disable Port**: If you want to provide reset information now but do not want to allow the reset capability at this time, check this box. If this box is checked, CXFS cannot reset the node.
- Owner (node that sends the reset command):
  - **Logical Name**: Name of the node that sends the reset command. Serial cables must physically connect the node being defined and the owner node through the system controller port. At run time, the node must be defined in the CXFS pool.

    You can select a logical name from the pull-down list or enter the logical name of a node that is not yet defined. However, you must define the node in CXFS before you run the node connectivity diagnostics task.
  - **TTY Device**: Name of the terminal port (TTY) on the owner node to which the system controller is connected. `/dev/ttyd2` is the most commonly

used port, except on Origin 300 and Origin 350 systems (where `/dev/ttyd4` is commonly used) and Altix 350 systems (where `/dev/ttyIOC0` is commonly used). The other end of the cable connects to this node's system controller port, so the node can be controlled remotely by the other node.

Note: To specify reset method that uses Intelligent Platform Management Interface (IPMI) or L2 over the network, use the `cxfs_admin` configuration tool.See "Create or Modify a Node with `cxfs_admin`" on page 235.

6. Click **OK**.

You can use the IRIX `mkpart` command to determine the partition ID value:

- The `-n` option lists the partition ID (which is 0 if the system is not partitioned).
- The `-l` option lists the bricks in the various partitions (use *rack#.slot#* format in the GUI).

For example (output truncated here for readability):

```
irix# mkpart -n
Partition id = 1
irix# mkpart -l
partition: 3 = brick: 003c10 003c13 003c16 003c21 ...
partition: 1 = brick: 001c10 001c13 001c16 001c21 ...
```

You could enter one of the following for the **Partition ID** field:

```
1
001.10
```

On SGI ProPack, you can find the partition ID by reading the `proc` file. For example:

```
propack# cat /proc/sgi_sn/partition_id
0
```

The `0` indicates that the system is not partitioned. If the system is partitioned, the number of partitions (such as `1`, `2`, etc.) is displayed.

Delete a Node with the GUI

You must remove a node from a cluster before you can delete the node from the pool. For information, see "Modify a Cluster Definition with the GUI" on page 187.

To delete a node, do the following:

1. **Node to Delete**: Select the logical name of the node to be deleted from the pull-down list.
2. Click **OK**.

Test Node Connectivity with the GUI

The **Test Node Connectivity** screen requires `rsh` access between hosts. The `/.rhosts` file must contain the hosts and local host between which you want to test connectivity.

To test connectivity, do the following from the CXFS Manager:

1. Choose whether to test by network or serial connectivity by clicking the appropriate radio button.
2. Choose a node to be tested from the pull-down list and add it to the test list by clicking **Add**.

   To delete a node from the list of nodes to be tested, click the logical name to select it and then click **Delete**.
3. To start the tests, click **Start Tests**. To stop the tests, click **Stop Tests**.
4. To run another test, click **Clear Output** to clear the status screen and start over with step 3.
5. To exit from the window, click **Close**.

Display a Node with the GUI

After you define nodes, you can use the **View** selection in the view area to display the following:

- **Nodes and Cluster** shows the nodes that are defined as part of a cluster or as part of the pool (but not in the cluster)

Click any name or icon to view detailed status and configuration information.

Cluster Tasks with the GUI

This section discusses the following:

- "Define a Cluster with the GUI" on page 186
- "Modify a Cluster Definition with the GUI" on page 187
- "Delete a Cluster with the GUI" on page 188
- "Display a Cluster with the GUI" on page 188

Note: The **Set Up a New Cluster** guided configuration task leads you through the tasks required to set up the cluster and nodes. See "Set Up a New Cluster with the GUI" on page 133.

Define a Cluster with the GUI

A *cluster* is a collection of nodes coupled to each other by a private network. A cluster is identified by a simple name. A given node may be a member of only one cluster.

To define a cluster, do the following:

1. Enter the following information:

   - **Cluster Name**: The logical name of the cluster. Logical names cannot begin with an underscore (_) or include any whitespace characters, and can be at most 255 characters. Clusters must have unique names.
   - **Cluster ID**: A unique number within your network in the range 1 through 255. The cluster ID is used by the operating system kernel to make sure that it does not accept cluster information from any other cluster that may be on the network. The kernel does not use the database for communication, so it requires the cluster ID in order to verify cluster communications. This information in the kernel cannot be changed after it has been initialized; therefore, you must not change a cluster ID after the cluster has been defined. Clusters must have unique IDs.
   - **Cluster Mode**: Usually, you should set the cluster to the default `Normal` mode.

Setting the mode to `Experimental` turns off CXFS kernel heartbeat in the CXFS kernel membership code so that you can debug the cluster without causing node failures. For example, this can be useful if you just want to disconnect the network for a short time (provided that there is no other cluster networking activity, which will also detect a failure even if there is no heartbeat) or if you want to enter the kernel debugger (which stops heartbeat) on a CXFS node. You should only use `Experimental` mode when debugging with the approval of SGI support.

- **Notify Administrator** (of cluster and node status changes):
  - **By e-mail**: This choice requires that you specify the e-mail program (`/usr/sbin/Mail` by default) and the e-mail addresses of those to be identified. To specify multiple addresses, separate them with commas. CXFS will send e-mail to the addresses whenever the status changes for a node or cluster. If you do not specify an address, notification will not be sent.
  - **By other command**: This choice requires that you specify the command to be run whenever the status changes for a node or cluster.
  - **Never**: This choice specifies that notification is not sent.

2. Click **OK**.

Modify a Cluster Definition with the GUI

To change how the cluster administrator is notified of changes in the cluster's state, do the following:

1. Enter the following information:
   - **Cluster Name**: Choose from the pull-down list.
   - **Cluster Mode**: Usually, you should set the cluster to the default `Normal` mode. See "Define a Cluster with the GUI" on page 186, for information about `Experimental` mode.
   - **Notify Administrator**: Select the desired notification. For more information, see "Define a Cluster with the GUI" on page 186.
2. Click **OK**.

To modify the nodes that make up a cluster, see "Add or Remove Nodes in the Cluster with the GUI" on page 180.

Note: If you want to rename a cluster, you must delete it and then define a new cluster. If you have started CXFS services on the node, you must either reboot it or reuse the cluster ID number when renaming the cluster.

However, be aware that if you already have CXFS filesystems defined and then rename the cluster, CXFS will not be able to mount the filesystems. For more information, see "Cannot Mount Filesystems" on page 372.

Delete a Cluster with the GUI

You cannot delete a cluster that contains nodes; you must move those nodes out of the cluster first. For information, see "Add or Remove Nodes in the Cluster with the GUI" on page 180.

To delete a cluster, do the following:

1. **Cluster to Delete**: The name of the cluster is selected for you.
2. Click **OK**.

Display a Cluster with the GUI

From the **View** selection, you can choose elements to examine. To view details of the cluster, click the cluster name or icon; status and configuration information will appear in the details area on the right.

Cluster Services Tasks with the GUI

This section discusses the following:

- "Start CXFS Services with the GUI" on page 189
- "Stop CXFS Services with the GUI" on page 189
- "Set Tiebreaker Node with the GUI" on page 190
- "Set Log Configuration with the GUI" on page 190

- "Revoke Membership of the Local Node with the GUI" on page 192
- "Allow Membership of the Local Node with the GUI" on page 193

Start CXFS Services with the GUI

To start CXFS services, and set the configuration to automatically restart CXFS services whenever the system is rebooted, do the following:

1. **Node(s) to Activate**: Select `All Nodes` or the individual node on which you want to start CXFS services.
2. Click **OK**.

Stop CXFS Services with the GUI

When CXFS services are stopped on a node, filesystems are automatically unmounted from that node. You can stop CXFS on a specified node or for the cluster, and prevent CXFS services from being restarted by a reboot, by performing the following steps.

Note: If you stop CXFS services using this method, they will not restart when the node is rebooted. To stop CXFS services temporarily (that is, allowing them to restart with a reboot if so configured), see "Manually Starting/Stopping CXFS" on page 308

1. Enter the following information:
   - **Force**: If you want to forcibly stop CXFS services even if there are errors (which would normally prevent the stop operation), click the **Force** checkbox.
   - **Node(s) to Deactivate**: Select `All Nodes` or the individual node on which you want to stop CXFS services.

     If you stop CXFS services on one node, that node will no longer have access to any filesystems. If that node was acting as the metadata server for a filesystem, another node in the list of potential metadata servers will be chosen. Clients of the filesystem will experience a delay during this process.
2. Click **OK**. It may take a few minutes to complete the process.

After you have stopped CXFS services on a node, the node is no longer an active member of the cluster. CXFS services will not be restarted when the system reboots.

Caution: You should stop CXFS services before using the `shutdown` or `reboot` commands. If you execute `shutdown` or `reboot` when CXFS services are active, the remaining nodes in the cluster will view it as a node failure and be forced to run recovery against that node.

Set Tiebreaker Node with the GUI

A *CXFS tiebreaker node* determines whether a CXFS kernel membership quorum is maintained when exactly half of the server-capable administration nodes are up and can communicate with each other. There is no default CXFS tiebreaker. For more information, see "CXFS Kernel Membership, Quorum, and Tiebreaker" on page 432.

Caution: If one of the server-capable administration nodes is the CXFS tiebreaker in a two server-capable cluster, failure of that node or stopping the CXFS services on that node will result in a cluster-wide forced shutdown. Therefore SGI recommends that you use client-only nodes as tiebreakers so that either server could fail but the cluster would remain operational via the other server.

To ensure data integrity, SGI recommends that you use system reset for all potential metadata servers and reset or or I/O fencing for all client-only nodes.

The current CXFS tiebreaker node is shown in the detailed view of the cluster.

To set the CXFS tiebreaker node, do the following:

1. **Tie-Breaker Node**: Select the desired node from the list. If there currently is a CXFS tiebreaker, it is selected by default.

   To unset the CXFS tiebreaker node, select `None`.

2. Click **OK**.

Set Log Configuration with the GUI

CXFS maintains logs for each of the CXFS daemons. CXFS logs both normal operations and critical errors to individual log files for each log group and the `/var/log/messages` system log file on server-capable administration nodes.

You can customize the logs according to the level of logging you wish to maintain.

Caution: Do not change the names of the log files. If you change the names, errors can occur.

When you define a log configuration, you specify the following information:

- **Log Group**: A set of one or more CXFS processes that use the same log configuration. A log group usually corresponds to one CXFS daemon, such as `crsd`.
- **Log Level**: A number controlling the amount of log messages that CXFS will write into an associated log group's log file.
- **Log File**: The file in which to log messages.

See also "Status in Log Files" on page 342.

Display Log Group Definitions with the GUI

To display log group definitions, do the following:

1. **Log Group**: Choose the log group to display from the menu.

   The current log level and log file for that log group will be displayed in the task window, where you can change those settings if you desire.

2. Click **OK**.

Configure Log Groups with the GUI

To configure a log group, do the following in the **Set Log Configuration** task:

1. Enter the appropriate information:

   - **Log Group**: Select the log group from the pull-down list. A *log group* is a set of processes that log to the same log file according to the same logging configuration. Each CXFS daemon creates a log group. Settings apply to all nodes in the pool for the `cli` and `crsd` log groups, and to all nodes in the cluster for the `clconfd` and `diags` log groups.

   - **Log Level**: Select the log level, which specifies the amount of logging.

   Caution: The **Default** log level is quite verbose; using it could cause space issues on your disk. You may wish to select a lower log level. Also see "Log File Management" on page 302, "`cad.options` on Server-Capable Administration Nodes" on page 120, and "`fs2d.options` on Server-Capable Administration Nodes" on page 121.

   The values are as follows:

   – **Off** gives no logging

   – **Minimal** logs notifications of critical errors and normal operation (these messages are also logged to the `/var/log/messages` file)

   – **Info** logs **Minimal** notifications plus warnings

   – **Default** logs all **Info** messages plus additional notifications

   – **Debug 0** through **Debug 9** log increasingly more debug information, including data structures

2. **Log File**: Do not change this value.

3. Click **OK**.

Revoke Membership of the Local Node with the GUI

You should revoke CXFS kernel membership of the local node only in the case of error, such as when you need to perform a forced CXFS shutdown (see "Shutdown of the Database and CXFS" on page 296).

To revoke CXFS kernel membership for the local node, do the following:

1. **Local Node:** Verify the name of the local node, which will be displayed in the pop-up window.
2. Click **OK** to complete the task.

This result of this task will be considered as a node failure by the rest of the cluster. The rest of the cluster may then fail due to a loss of CXFS kernel membership quorum, or it may decide to reset the failed node. To avoid the reset, you can modify the node definition to disable the system controller status.

Allow Membership of the Local Node with the GUI

You must allow CXFS kernel membership for the local node (the node to which the GUI is connected) after fixing the problems that required a forced CXFS shutdown; doing so allows the node to reapply for CXFS kernel membership in the cluster. A forced CXFS shutdown can be performed manually or can be triggered by the kernel. For more information, see "Shutdown of the Database and CXFS" on page 296.

You must actively allow CXFS kernel membership of the local node in the following situations:

- After a manual revocation as in "Revoke Membership of the Local Node with the GUI" on page 192.
- When instructed to by an error message on the console or the `/var/log/messages` system log file.
- After a kernel-triggered revocation. This situation is indicated by the following message in the `/var/log/messages` system log file:

  ```
  Membership lost - withdrawing from cluster
  ```

To allow CXFS kernel membership for the local node, do the following:

1. **Local Node:** Verify the name of the local node, which will be displayed in the pop-up window.
2. Click **OK** to complete the task.

Switches and I/O Fencing Tasks with the GUI

This section discusses the following:

- "Define a Switch with the GUI" on page 194
- "Modify a Switch Definition with the GUI" on page 197
- "Update Switch Port Information with the GUI" on page 197
- "Delete a Switch Definition with the GUI" on page 197
- "Raise the I/O Fence for a Node with the GUI" on page 198
- "Lower the I/O Fence for a Node with the GUI" on page 198

See the release notes for supported switches.

Note: Nodes without system controllers require I/O fencing to protect data integrity.

Define a Switch with the GUI

This task lets you define a new switch to support I/O fencing in a cluster.

Do the following:

1. Enter the following information:
   - **Switch Name:** Enter the hostname of the switch; this is used to determine the IP address of the switch.
   - **Username:** Enter the user name to use when sending a `telnet` message to the switch. By default, this value is `admin`.
   - **Password:** Enter the password for the specified **Username** field.
   - **Mask:** Enter one of the following:
     – A list of ports in the switch that will never be fenced. The list has the following form, beginning with the # symbol and separating each port number with a comma:

       #port`,`*port*`,`*port*`. . .`

Each *port* is a decimal integer in the range 0 through 1023. Use a hyphen to specify an inclusive range. For example, the following indicates that port numbers 2, 4, 5, 6, 7, and 23 will never be fenced:

```
#2,4-7,23
```

Note: For the bladed Brocade 48000 switch (where the port number is not unique), the value you should use for **Mask** is the `Index` value that is displayed by the `switchShow` command. For example, the `switchShow` output below indicates that you would use a `mask` value of `16` for port `0` in slot `2`:

```
brocade48000:admin> switchShow
Index Slot Port Address Media Speed State     Proto
===================================================
  0    1    0   010000   id    N4   Online    F-Port  10:00:00:00:c9:5f:9b:ea
  1    1    1   010100   id    N4   Online    F-Port  10:00:00:00:c9:5f:ab:d9
....
142    1   30   018e00   id    N4   Online    F-Port  50:06:0e:80:04:5c:0b:46
143    1   31   018f00   id    N4   Online    F-Port  50:06:0e:80:04:5c:0b:66
 16    2    0   011000   id    N4   Online    F-Port  10:00:00:00:c9:5f:a1:f5
 17    2    1   011100   id    N4   Online    F-Port  10:00:00:00:c9:5f:a1:72
...
```

– A hexadecimal string that represents the list of ports in the switch that will never be fenced.

Ports are numbered from zero. If a given bit has a binary value of 0, the port that corresponds to that bit is eligible for fencing operations; if 1, then the port that corresponds to that bit will always be excluded from any fencing operations. For example, Figure 10-5 shows that a mask of `FF03` for a 16-port switch indicates that only ports 2–7 are eligible for fencing (because they have binary values of 0). Similarly, it shows that a mask of `A4` for an 8-port switch allows fencing only on ports 0, 1, 3, 4, and 6 (the port numbers corresponding to binary 0) — ports 2, 5, and 7 will never be fenced (the port numbers corresponding to the nonzero value).

16-port Switch (1= never fence, 0= may fence)

Port #	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Binary	1 1 1 1	1 1 1 1	0 0 0 0	0 0 1 1
Hexadecimal	F	F	0	3

8-port Switch

Port #	7 6 5 4	3 2 1 0
Binary	1 0 1 0	0 1 0 0
Hexadecimal	A	4

Figure 10-5 Bit Mask Representation for I/O Fencing

Server-capable administration nodes automatically discover the available HBAs and, when fencing is triggered, will fence off all of the Fibre Channel HBAs when the **Fence** or **FenceReset** fail policy is selected. However, masked HBAs will not be fenced. Masking allows you to prevent the fencing of devices that are attached to the SAN but are not shared with the cluster, to ensure that they remain available regardless of CXFS status. You would want to mask HBAs used for access to tape storage, or HBAs that are only ever used to access local (nonclustered) devices.

- **Vendor**: Select the name of the switch vendor or enter the vendor name manually if not found in the list.

2. Click **OK** to complete the task.

Modify a Switch Definition with the GUI

This task lets you modify an existing switch definition.

Do the following:

1. Enter the following information:
   - **Switch Name:** Select the hostname of the switch to be modified.
   - **Username:** Enter the user name to use when sending a `telnet` message to the switch. By default, this value is `admin`.
   - **Password:** Enter the password for the specified **Username** field.
   - **Mask:** Enter a list of port numbers or a hexadecimal string that represents the list of ports in the switch that will not be fenced. For more information, see "Define a Switch with the GUI" on page 194.
2. Click **OK** to complete the task.

Note: You cannot modify the vendor name for a switch. To use a different vendor, delete the switch and redefine it.

Update Switch Port Information with the GUI

This task lets you update the mappings between the host bus adapters (HBAs) and switch ports. You should run this command if you reconfigure any switch or add ports. Click **OK** to complete the task.

Delete a Switch Definition with the GUI

This task lets you delete an existing switch definition. Do the following:

1. **Switch Name:** Select the hostname of the Fibre Channel switch to be deleted.
2. Click **OK** to complete the task.

Raise the I/O Fence for a Node with the GUI

This task lets you raise the I/O fence for a node. Raising an I/O fence isolates the node from the SAN; CXFS sends a messages via the `telnet` protocol to the switch and disables the port. After the node is isolated, it cannot corrupt data in the shared CXFS filesystem.

Do the following:

1. **Raise Fence for Node:** Select the name of the node you want to isolate. Only nodes that have been configured with a **Fence** or **FenceReset** fail policy can be selected.
2. Click **OK** to complete the task.

Lower the I/O Fence for a Node with the GUI

This task lets you lower the I/O fence for a given node by reenabling the port. Lowering an I/O fence allows the node to reconnect to the SAN and access the shared CXFS filesystem.

Do the following:

1. **Lower Fence for Node:** Select the node you want to reconnect. Only nodes that have been configured with a **Fence** or **FenceReset** fail policy can be selected.
2. Click **OK** to complete the task.

Filesystem Tasks with the GUI

The following tasks let you configure CXFS filesystems as shared XVM volumes. These shared volumes can be directly accessed by all nodes in a CXFS cluster. Each volume is identified by its device name. Each volume must have the same mount point on every node in the cluster.

Note: The **Set Up a New CXFS Filesystem** guided configuration task leads you through the steps required to set up a new CXFS filesystem. See "Set Up a New CXFS Filesystem with the GUI" on page 134.

This section discusses the following:

- "Make Filesystems with the GUI" on page 199
- "Grow a Filesystem with the GUI" on page 201
- "Define CXFS Filesystems with the GUI" on page 202
- "Modify a CXFS Filesystem with the GUI" on page 206
- "Mount CXFS Filesystems with the GUI" on page 207
- "Unmount CXFS Filesystems with the GUI" on page 208
- "Mount a Filesystem Locally" on page 208
- "Unmount a Local Filesystem" on page 209
- "Delete a CXFS Filesystem with the GUI" on page 209
- "Remove Filesystem Mount Information" on page 209
- "Relocate a Metadata Server for a CXFS Filesystem with the GUI" on page 210

Make Filesystems with the GUI

This task lets you create a filesystem on a volume that is online but not open. To create filesystems on multiple volume elements, use the **Browse** button.

Caution: Clicking OK will erase all data that exists on the target volume.

To make a filesystem, do the following:

1. Enter the following information:

   - **Domain**: Select the domain that will own the volume element to be created. Choose **Local** if the volume element or disk is defined for use only on the node to which the GUI is connected, or choose **Cluster** if it is defined for use on multiple nodes in the cluster.

   - **Volume Element**: Select the volumes on which to create the filesystem or select the volume elements whose parent volumes will be used for the filesystems. The menu lists only those volume elements that are available. (When volume elements other than volumes are created or detached, the system automatically creates a volume and a subvolume that are associated with the volume element. If you did not explicitly name an automatically generated volume, the GUI will display its children only.)

   - **Specify Sizes**: Check this box to modify the default options for the filesystem, including data region size, and log size.

     By default, the filesystem will be created with the data region size equal to the size of the data subvolume. If the volume contains a log subvolume, the log size will be set to the size of the log subvolume.

2. If you checked the **Specify Sizes** box, click **Next** to move to page 2. On page 2, enter the following information. For more information about these fields, see the `mkfs.xfs` man page.

   - **Block Size**: Select the fundamental block size of the filesystem in bytes.

   - **Directory Block Size**: Select the size of the naming (directory) area of the filesystem in bytes.

   - **Inode Size**: Enter the number of blocks to be used for inode allocation, in bytes. The inode size cannot exceed one half of the **Block Size** value.

   - **Maximum Inode Space**: Enter the maximum percentage of space in the filesystem that can be allocated to inodes. The default is 25%. (Setting the value to 0 means that the entire filesystem can become inode blocks.)

   - **Flag Unwritten Extents**: Check this box to flag unwritten extents. If unwritten extents are flagged, filesystem write performance will be negatively affected for preallocated file extents because extra filesystem transactions are required to convert extent flags for the range of the file.

You should disable this feature (by unchecking the box) if the filesystem must be used on operating system versions that do not support the flagging capability.

- **Data Region Size**: Enter the size of the data region of the filesystem as a number of 512-byte blocks. This number is usually equal to the size of the data subvolume. You should specify a size other than 0 only if the filesystem should occupy less space than the size of the data subvolume.
- **Use Log Subvolume for Log**: Check this box to specify that the log section of the filesystem should be written to the log subvolume of the XVM logical volume. If the volume does not contain a log subvolume, the log section will be a piece of the data section on the data subvolume.
- **Log Size**: Enter the size of the log section of the filesystem as a number of 512-byte blocks. You should specify a size other than 0 only if the log should occupy less space than the size of the log subvolume.

Note: XVM on SGI ProPack does not support real-time subvolumes.

3. Click **OK**.

Grow a Filesystem with the GUI

This task lets you grow a mounted filesystem.

Note: In order to grow a filesystem, you must first increase the size of the logical volume on which the filesystem is mounted. For information on modifying XVM volumes, see the *XVM Volume Manager Administrator's Guide*.

To grow a filesystem, do the following:

1. Enter the following information:
   - **Filesystem**: Select the name of the filesystem you want to grow. The list of available filesystems is determined by looking for block devices containing XFS superblocks.
   - **Specify Sizes**: Check this option to modify the default options for the filesystem, including data region size and (if already present for the filesystem) log size.

By default, the filesystem will be created with the data region size equal to the size of the data subvolume. If the volume contains a log subvolume, the log size will be set to the size of the log subvolume.

2. If you checked the **Specify Sizes** box, click **Next** to move to page 2. For more information about these fields, see the `mkfs.xfs` man page.

   - **Data Region Size**: Enter the size of the data region of the filesystem as a number of 512-byte blocks. This number is usually equal to the size of the data subvolume. You should specify a size other than 0 only if the filesystem should occupy less space than the size of the data subvolume.

   - **Log Size**: Enter the size of the log section of the filesystem as a number of 512-byte blocks. You should specify a size other than 0 only if the log should occupy less space than the size of the log subvolume. This option only appears if the filesystem has a log subvolume.

   Note: XVM on SGI ProPack does not support real-time subvolumes.

3. Click **OK**.

Define CXFS Filesystems with the GUI

This task lets you define one or more CXFS filesystems having the same ordered list of potential metadata servers and the same list of client nodes.

Note: The list of potential metadata servers for a given filesystem is ordered, but because of network latencies and other unpredictable delays, it is impossible to predict which node will become the active metadata server.

If you select multiple device names, the path you enter for the mount point will be used as a prefix to construct the actual mount point for each filesystem.

This task assumes that you have created volume headers on your disk drives, created the XVM logical volumes, and made the filesystems. "Initial Setup with the CXFS GUI" on page 131.

To define filesystems, do the following:

1. Enter the following information:

- **Device Name**: Select the device names of the XVM volumes on which the filesystems will reside.
- **Mount Point**: The directory on which the specified filesystem will be mounted. This directory name must begin with a slash (`/`). The same mount point will be used on all the nodes in the cluster. For example, if you select the device name `/dev/cxvm/cxfs1` and want to mount it at `/mount/cxfs1`, you would enter `/mount/cxfs1` for the **Mount Point** value.

  If you selected multiple device names in order to define multiple CXFS filesystems, the mount point path will be constructed using the mount point you enter as a **prefix** and the name of each device name (not including the `/dev/cxvm` portion) as the suffix. For example, if you select two volume device names (`/dev/cxvm/cxfs1` and `/dev/cxvm/cxfs2`) and enter a mount point of `/mount/`, then the CXFS filesystems will be mounted as `/mount/cxfs1` and `/mount/cxfs2`, respectively. If instead you had entered `/mount` for the mount point, the filesystems would be mounted as `/mountcxfs1` and `/mountcxfs2`.

  For more information, see the `mount` man page.
- (*Optional*) **Mount Options**: These options are passed to the `mount` command and are used to control access to the specified XVM volume. Separate multiple options with a comma. For a list of the available options, see the `fstab` man page.
- **Force Unmount**: Select the default behavior for the filesystem. This option controls what action CXFS takes if there are processes that have open files or current directories in the filesystems that is to be unmounted. If you select **On**, the processes will be killed and the unmount will occur. If you select **Off**, the processes will not be killed and the filesystem will not be unmounted. SGI recommends that you set **Force Unmount** to **On** in order to improve the stability of the CXFS cluster. This value can be overridden when you perform a manual unmount; see "Unmount CXFS Filesystems with the GUI" on page 208.
- **Metadata Servers**: A list of server-capable administration nodes that are able to act as metadata servers.

  To add a server-capable administration node to the list of servers, choose a name from the pull-down node list and click **Add**. To select all nodes listed, click **Add All**.

Note: Relocation is disabled by default. Recovery and relocation are supported only when using standby nodes. Therefore, you should only define multiple metadata servers for a given filesystem if you are using the standby node model. See "Relocation" on page 25.

To remove a node from the list of servers, click the name in the list to select it and then click **Remove**.

Note: The order of servers is significant. The first node listed is the preferred metadata server. Click a logical name to select it and then click the arrow buttons to arrange the servers in the order that they should be used.

However, it is impossible to predict which server will actually become the server during the boot-up cycle because of network latencies and other unpredictable delays. The first available node in the list will be used as the active metadata server.

- **Enable Mount on:** A choice of either all nodes in the cluster or a list of selected server-capable administration nodes that you specify on a second page. (The filesystem is always mounted on the current metadata server.)
- **If Nodes are Added to the Cluster Later:** This option permits the filesystem to be mounted on all nodes that might be added to the cluster at some later date. This option is selected by default.
- If you chose **Only Selected Nodes** above, click **Next** to move to the second page of the task.

  Selected Nodes: You can select the desired nodes from the **Node** list. You can also click **Add All** to select all nodes, which is the same as selecting **All Nodes Currently in Cluster**.

2. Click **OK**.

After defining the filesystems, you can mount them on the specified client nodes in the cluster by running the **Mount CXFS Filesystems** task.

Note: After a filesystem has been defined in CXFS, running `mkfs` on it (or using the "Make Filesystems with the GUI" on page 199 task) will cause errors to appear in the system log file. To avoid these errors, run `mkfs` before defining the filesystem in CXFS, or delete the CXFS filesystem before running `mkfs`. See "Delete a CXFS Filesystem with the GUI" on page 209.

Modify a CXFS Filesystem with the GUI

Note: You cannot modify a mounted filesystem.

To modify an existing filesystem, do the following:

1. Enter the following information:

   - **Filesystem to Modify**: Choose a filesystem from the pull-down menu. This displays information for that filesystem in the various fields.
   - **Mount Point** and **Mount Options**: Change the information displayed for the selected filesystem as needed. To erase text, backspace over the text or select the text and type over it.
   - (*Optional*) **Mount Options**: These options are passed to the `mount` command and are used to control access to the specified XVM volume. For a list of the available options, see the `fstab` man page.
   - **Metadata Servers**:
     - To delete a node from the list of servers, click its name and then click **Delete**.
     - To add a new server-capable administration node to the list of servers, select it from the pull-down list and click **Add**. To select all server-capable administration nodes, select **Add All**. The list for a given filesystem must consist of nodes running the same operating system.
     - To rearrange the priority of a server, select it by clicking its name and then click the arrow buttons as needed.

       Note: The list of potential metadata servers for a given filesystem is ordered, but because of network latencies and other unpredictable delays, it is impossible to predict which node will become the active metadata server.
   - **Enable Mount on:** A choice of either all nodes in the cluster or a list of selected nodes that you specify on a second page. (The filesystem is always mounted on the current metadata server.)

- **If Nodes are Added to the Cluster Later:**This option permits the filesystem to be mounted on all nodes that might be added to the cluster at some later date. This option is selected by default.
- If you chose **Only Selected Nodes** above, click **Next** to move to the second page of the task.

  Selected Nodes: You can select the desired nodes from the **Node** list. You can also click **Add All** to select all nodes, which is the same as selecting **All Nodes Currently in Cluster**.

2. Click **OK**.

Mount CXFS Filesystems with the GUI

To mount existing filesystems on all of their client nodes, do the following:

1. **Filesystem to Mount**: Choose the filesystem to be mounted.
2. Click **OK**.

If CXFS services are not active, mounting a filesystem will not completely succeed. The filesystem will be marked as ready to be mounted and a warning message will be displayed in the **Mount a Filesystem** task. The filesystem will not actually be mounted until you have started CXFS services. For information, see "Start CXFS Services with the GUI" on page 189.

Unmount CXFS Filesystems with the GUI

To unmount filesystems from all of their client nodes, do the following:

1. Enter the following information:

   - **Filesystem to Unmount**: Choose the filesystems to be unmounted.
   - **Force Unmount** : Click **On** to force an unmount for all selected filesystems (no matter how they have been defined) or **Default** to force an unmount for those filesystems that have the forced unmount option set in their definition.

     This option controls what action CXFS takes if there are processes that have open files or current directories in the filesystems that are to be unmounted. If forced is used (by selecting **On** or by selecting **Default** if force is the default behavior), the processes will be killed and the unmount will occur. If you select **Off**, the processes will not be killed and the filesystem will not be unmounted. The option is set to **Default** by default.

2. Click **OK**.

Mount a Filesystem Locally

This task lets you mount a filesystem only on the node to which the GUI is connected (the local node).

To mount a filesystem locally, do the following:

1. Enter the following information:

   - **Filesystem to Mount**: Select the filesystem you wish to mount. The list of available filesystems is determined by looking for block devices containing XFS superblocks.
   - **Mount Point**: Specify the directory on which the selected filesystem will be mounted.
   - (*Optional*) **Mount Options**: Specify the options that should be passed to the `mount` command. For more information about available options, see the `fstab` man page.

2. By default, the filesystem will remount every time the system starts. However, if you uncheck the box, the mount will take place only when you explicitly use this task.

3. Click **OK**.

For more information, see the `mount` man page.

Unmount a Local Filesystem

To unmount a filesystem from the local node, do the following:

1. Enter the following information:
   - **Filesystem to Unmount**: Choose the filesystem to be unmounted.
   - **Remove Mount Information**: Click the check box to remove the mount point from the `/etc/fstab` file, which will ensure that the filesystem will remain unmounted after the next reboot. This item is available only if the mount point is currently saved in `/etc/fstab`.
2. Click **OK**.

Delete a CXFS Filesystem with the GUI

You cannot delete a filesystem that is currently mounted. To unmount a filesystem, see "Unmount CXFS Filesystems with the GUI" on page 208.

To permanently delete an unmounted filesystem, do the following:

1. **Filesystem to Delete**: Choose the name of the filesystem from the pull-down list.
2. Click **OK**.

Remove Filesystem Mount Information

This task lets you delete a local filesystem's mount information in `/etc/fstab`.

Note: The filesystem will still be present on the volume.

Do the following:

1. **Filesystem Name**: Select the filesystem for which you want to remove mount information. The list of available filesystems is determined by looking for block devices containing XFS superblocks.
2. Click **OK**.

Relocate a Metadata Server for a CXFS Filesystem with the GUI

If relocation is explicitly enabled in the kernel with the `cxfs_relocation_ok` systune, you can relocate the metadata server for a filesystem to any other potential metadata server in the list (see "Relocation" on page 25). The filesystem must be mounted on the system to which the GUI is connected.

1. Enter the following information:
   - **Filesystem**: Select the desired filesystem from the list.
   - **Current Metadata Server:** The current metadata server will be displayed for you.
   - **New Metadata Server**: Select the desired node from the list.

     The selected server will assume responsibility for moderating access to the selected filesystem **after** you run the **Start CXFS Services** task; see "Start CXFS Services with the GUI" on page 189.
2. Click **OK** to complete the task.

CXFS kernel membership is not affected by relocation. However, users may experience a degradation in filesystem performance while the metadata server is relocating.

Privileges Tasks with the GUI

The privileges tasks let you grant specific users the ability to perform specific tasks and to revoke those privileges.

Note: You cannot grant or revoke tasks for users with a user ID of 0.

This section discusses the following:

- "Grant Task Access to a User or Users" on page 211
- "Revoke Task Access from a User or Users" on page 214

Grant Task Access to a User or Users

You can grant access to a specific task to one or more users at a time.

Note: Access to the task is only allowed on the node to which the GUI is connected; if you want to allow access on another node in the pool, you must connect the GUI to that node and grant access again.

Do the following:

1. Select the user or users for whom you want to grant access. You can use the following methods to select users:

   - Click to select one user at a time
   - `Shift`+click to select a block of users
   - `Ctrl`+click to toggle the selection of any one user, which allows you to select multiple users that are not contiguous
   - Click **Select All** to select all users

   Click **Next** to move to the next page.

2. Select the task or tasks to grant access to, using the above selection methods. Click **Next** to move to the next page.

3. Confirm your choices by clicking **OK**.

Note: If more tasks than you selected are shown, then the selected tasks run the same underlying privileged commands as other tasks, such that access to the tasks you specified cannot be granted without also granting access to these additional tasks.

To see which tasks a specific user can access, select **View: Users**. Select a specific user to see details about the tasks available to that user.

To see which users can access a specific task, select **View: Task Privileges**. Select a specific task to see details about the users who can access it and the privileged commands it requires.

Granting Access to a Few Tasks

Suppose you wanted to grant user `diag` permission to define, modify, and mount CXFS filesystems. You would do the following:

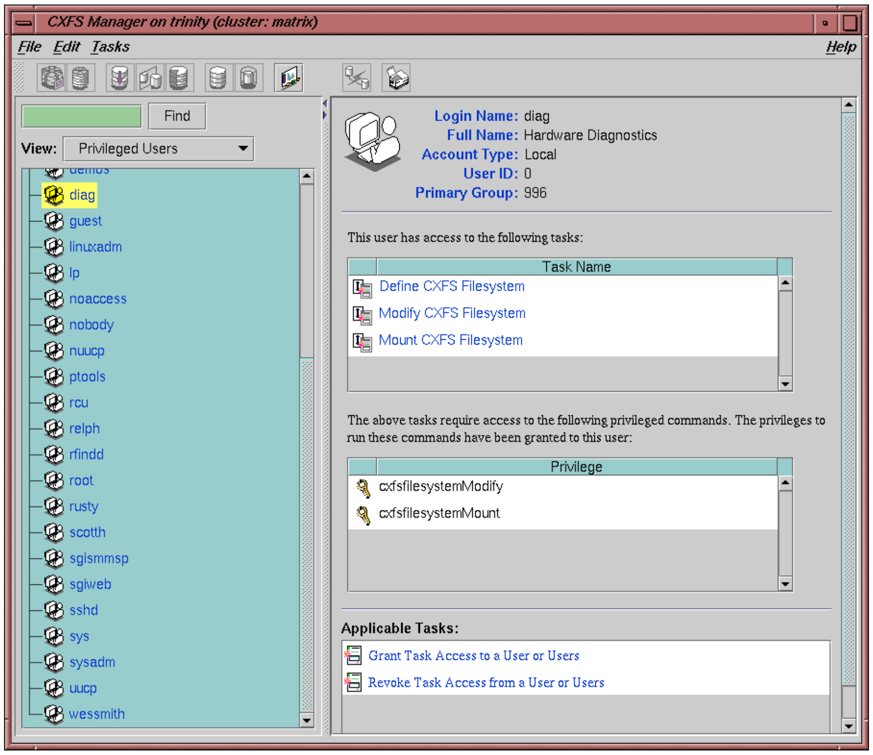
1. Select `diag` and click **Next** to move to the next page.
2. Select the tasks you want `diag` to be able to execute:
   a. `Ctrl`+click **Define CXFS Filesystem**

   b. `Ctrl`+click **Modify CXFS Filesystem**

   c. `Ctrl`+click **Mount CXFS Filesystem**

   Click **Next** to move to the next page.
3. Confirm your choices by clicking **OK**.

Figure 10-6 shows the tasks that `diag` can now execute. This screen is displayed when you select **View: Users** and click `diag` to display information in the details area of the GUI window. The privileged commands listed are the underlying commands executed by the GUI tasks.



Login Name: diag
Full Name: Hardware Diagnostics
Account Type: Local
User ID: 0
Primary Group: 996

This user has access to the following tasks:

Task Name
Define CXFS Filesystem
Modify CXFS Filesystem
Mount CXFS Filesystem

The above tasks require access to the following privileged commands. The privileges to run these commands have been granted to this user:

Privilege
cxfsfilesystemModify
cxfsfilesystemMount

Applicable Tasks:

Grant Task Access to a User or Users
Revoke Task Access from a User or Users

Figure 10-6 Task Privileges for a Specific User

Granting Access to Most Tasks

Suppose you wanted to give user `sys` access to all tasks **except** changing the cluster contents (which also implies that `sys` cannot delete the nodes in the cluster, nor the cluster itself). The easiest way to do this is to select all of the tasks and then deselect the few you want to restrict. You would do the following:

1. Select `sys` and click **Next** to move to the next page.
2. Select the tasks you want `sys` to be able to execute:
   a. Click **Select All** to highlight all tasks.
   b. Deselect the task to which you want to restrict access. `Ctrl`+click **Add/Remove Nodes in Cluster**.

Click **Next** to move to the next page.

3. Confirm your choices by clicking **OK**.

Revoke Task Access from a User or Users

You can revoke task access from one or more users at a time.

Note: Access to the task is only revoked on the node to which the GUI is connected; if a user has access to the task on multiple nodes in the pool, you must connect the GUI to those other nodes and revoke access again.

Do the following:

1. Select the user or users from whom you want to revoke task access. You can use the following methods to select users:
   - Click to select one user at a time
   - `Shift`+click to select a block of users
   - `Ctrl`+click to toggle the selection of any one user, which allows you to select multiple users that are not contiguous
   - Click **Select All** to select all users

   Click **Next** to move to the next page.
2. Select the task or tasks to revoke access to, using the above selection methods. Click **Next** to move to the next page.
3. Confirm your choices by clicking **OK**.

   Note: If more tasks than you selected are shown, then the selected tasks run the same underlying privileged commands as other tasks, such that access to the tasks you specified cannot be revoked without also revoking access to these additional tasks.

To see which tasks a specific user can access, select **View: Users**. Select a specific user to see details about the tasks available to that user.

To see which users can access a specific task, select **View: Task Privileges**. Select a specific task to see details about the users who can access it.

Chapter 11

Reference to cxfs_admin Tasks

This chapter discusses the following:

- "cxfs_admin Overview" on page 217
- "Node Tasks with cxfs_admin" on page 234
- "Cluster Tasks with cxfs_admin" on page 248
- "CXFS Filesystem Tasks with cxfs_admin" on page 253
- "Network Failover Tasks with cxfs_admin" on page 262
- "Switch Tasks with cxfs_admin" on page 262
- "Saving and Recreating the Current Configuration with cxfs_admin" on page 269

Note: The example output might not exactly match the output of your system.

For an overview of the tasks that must be performed to configure a cluster, see "Initial Setup with the cxfs_admin Command" on page 135.

You can also use the clconf_info tool to view status. See Chapter 14, "Monitoring Status" on page 341.

For help with error messages, see "cxfs_admin Errors" on page 407.

See also Appendix H, "Migration from cmgr to cxfs_admin" on page 595.

cxfs_admin Overview

To use the cxfs_admin command, you must be logged to a node that has permission to access the CXFS cluster database; to make configuration changes using cxfs_admin, you must be logged in as root. See "Setting cxfs_admin Access Permissions" on page 233.

Note: For the steps to create a cluster for the first time, see "Initial Setup with the cxfs_admin Command" on page 135.

This section discusses the following:

- "Starting `cxfs_admin`" on page 218
- "Command Syntax Overview" on page 219
- "Getting Help" on page 221
- "Making Changes Safely" on page 222
- "Basic and Advanced Mode" on page 223
- "Using Prompting Mode" on page 225
- "Command History" on page 226
- "Waiting for Commands to Complete" on page 226
- "Entering `cxfs_admin` Commands on the Command Line" on page 227
- "Using Script Files" on page 228
- "Setting `cxfs_admin` Defaults" on page 231
- "Setting `cxfs_admin` Access Permissions" on page 233
- "Exiting from `cxfs_admin`" on page 234

Starting `cxfs_admin`

To start `cxfs_admin`, enter the following:

```
/usr/cluster/bin/cxfs_admin
```

See also:

- "Entering `cxfs_admin` Commands on the Command Line" on page 227
- "Accessing the Correct Cluster at a Multiple-Cluster Site" on page 234
- "Setting `cxfs_admin` Access Permissions" on page 233

Command Syntax Overview

Some `cxfs_admin` commands affect the `cxfs_admin` operating environment itself, some display status information, and others affect objects or classes. Within `cxfs_admin`, an *object* is a specific item that is configured in the CXFS cluster and a *class* contains a group of similar objects. For example, the filesystem names `fs1` and `fs2` would both be objects within the `filesystem` class.

Within a class, all objects must have unique names. If all objects in the cluster have unique names, you can abbreviate some commands by omiting the class name. However, if two or more objects in the cluster database have the same name, you must specify the class in order to uniquely identify the object.

The basic command syntax is:

command [[class:]object] [attributes]

where *attributes* takes a number of forms depending on the context:

attribute
attribute=value
attribute=value1,value2,value3...

The actual syntax components for any given command varies, based on the needs of the command. For example, the following command requires no parameters to see a summary of the cluster:

```
cxfs_admin:mycluster> show
```

If an object name is unique within the cluster database, you can omit its class name. For example, if the name `nodeA` is unique within the database:

```
cxfs_admin:mycluster> show nodeA
```

However, if there were multiple objects named `production`, you must enter the class name:

```
cxfs_admin:mycluster> show node:production
```

Classes and objects may include the following shell-style wildcard characters:

```
*
?
[...]
```

Note: You can use the asterisk (*) alone only with the show and configcommands, which do not make changes to the database. For these commands, you can use a * character in place of the *object* to apply the command to entire cluster (if you do not specify any attributes, you can omit the * character.)

As a safety measure, however, using a wildcard in other instances requires that you also include at least one other character in the object's name; for example, if you wanted to unmount both myfileA and yourfileB, you could enter unmount *file*.

With the show and config commands,

Command names and attribute names are not case-sensitive. However, all attribute values other the node name (in create node *nodename*) and the switch name (in create switch *switchname*) are case sensitive.

You can see possible attributes by pressing the <TAB> key after entering the command or object. For example:

```
cxfs_admin:mycluster> create filesystem <TAB>
  Required attributes:
    name= : A string

  Optional attributes:
    forced_unmount= : True/false or enabled/disabled (default is "false")
    mounted=        : True/false or enabled/disabled (default is "true")
    mountpoint=     : A pathname
    options=        : Nothing, one or more strings (can be empty)
```

The required attributes are listed first followed by optional attributes. The list of attributes will differ depending upon whether you are in basic or advanced mode; see "Basic and Advanced Mode" on page 223.

Partially typing in the attribute name and pressing <TAB> will complete the attribute name if unique, or show a list of matching attribute names. To see what kind of values are required for an attribute, press <TAB> after the = sign. For example:

```
cxfs_admin:mycluster> create node os=<TAB>
  AIX       IRIX      Linux     MacOSX    Solaris   Unknown   Windows
```

Use $ to refer to the object in the last command.

For example, to delete `nodeA`, if it has a unique name within the cluster database:

```
cxfs_admin:mycluster> disable nodeA
cxfs_admin:mycluster> delete $
```

To specify multiple objects, separate them with a comma. For example:

```
cxfs_admin:mycluster> show nodeA,nodeB
```

You can abbreviate commands, objects, and attributes by entering in the first character or two followed by pressing the `<TAB>` key. If more than one match is available, `cxfs_admin` shows a list of the possible matches.

Getting Help

At any time, you can enter `help` or `?` to see help text.

To see help for a given topic:

`help` *topicname*

The list of general topics includes the following:

```
attributes
commandline
commands
cxfs
objects
overview
setup
syntax
tasks
waiting
```

There is also help for each `cxfs_admin` command. For example, to see help about the `create` command:

```
cxfs_admin:mycluster> help create
```

To see all of the available help topics, press the `<TAB>` key:

```
cxfs_admin:mycluster> help <TAB>
```

To see a list of available commands for an object, such as a class like `filesystem` or a specific instance of a class like the filesystem `myfs`, use the `ops` command:

`ops` *object*

For example:

```
cxfs_admin:mycluster> ops filesystem
Commands for "filesystem":
    config, create, ops, show
cxfs_admin:mycluster> ops myfs
Commands for "filesystem:myfs":
    config, delete, modify, mount, ops, relocate, show, unmount
```

Making Changes Safely

The `cxfs_admin` tool only allows one user to use `cxfs_admin` to make changes to the cluster database at a time. If you are the first person to invoke `cxfs_admin`, you automatically get the lock. If someone else already has the lock, you will enter in read-only mode. If you are in read-only mode, it is reflected in the `cxfs_admin` prompt.

To forcefully obtain the lock from someone else, you can use the `steal` attribute with the `lock` command. For example:

```
cxfs_admin:mycluster (read only) > lock
The administration lock is already held by root@node2 (pid=48449)
cxfs_admin:mycluster (read only) > lock steal=true
The administration lock has been stolen from root@node2 (pid=48449)
cxfs_admin:mycluster>
```

If someone holds the lock while you are using `cxfs_admin` but later drops it, there is no need to steal the lock.

If you want to manually enter read-only mode, use the `unlock` command. For example:

```
cxfs_admin:mycluster> unlock
cxfs_admin:mycluster (read only) >
```

Caution: The `cxfs_admin` lock does not prevent other users from using the CXFS GUI while `cxfs_admin` is running. You should make database changes with only one instance of the CXFS GUI or locked `cxfs_admin` commands at any one time.

Basic and Advanced Mode

The `cxfs_admin` operates in two modes:

- Basic, which only shows the common options and attributes in `show` output, `<TAB>` key completion, and prompting mode.
- Advanced, which allows `<TAB>` key completion, prompts for all possible fields, displays all attributes, and includes debugging information in output. Advanced-mode commands and attributes are not included in prompts or `<TAB>` key completion when you are in basic mode. However, you can still manually enter an advanced attribute if you know it, even in basic mode.The advanced commands and attributes are noted in their help topics.

  Note: You should only use the advanced-mode commands and attributes at the advice of SGI support. Using the advanced mode commands or changing advanced mode attributes may induce unexpected behavior.

You can enter advanced mode by using `cxfs_admin -a` on the command line or by entering the following `cxfs_admin` command:

```
cxfs_admin:mycluster> set mode=advanced
```

To return to basic mode:

```
cxfs_admin:mycluster> set mode=basic
```

For example, the following output shows only the basic-mode information:

```
cxfs_admin:cc_test2> set mode=basic
Mode is set to basic
cxfs_admin:cc_test2> show cc-xe
node:cc-xe:
    cellid=0
    enabled=true
    os=Linux
    private_net:
        10.11.0.239, 128.162.242.4
    status:
        connected=true
        fencing=Stable
        license:
            have_license=unknown
        summary=Stable
        version=5.0.0.1_ALPHA
        wwns:
            100000062b0f5284, 100000062b0f5285
    type=server_admin
```

Note: The `have_license` field reports licenses for client-only nodes that have been obtained from the server. It does not display information about the server licenses. For more information about licensing, see "Show License Information with `cxfs_admin`" on page 252.

For example, the following output shows all information that is available in advanced mode for `node2`:

```
cxfs_admin:cc_test2> set mode=advanced
Mode is set to advanced
cxfs_admin:cc_test2> show cc-xe
node:cc-xe:
    cellid=0
    clustername=cc_test2
    enabled=true
    failpolicy=Fence,Shutdown
    hostname=cc-xe.americas.sgi.com
    nodeid=10
    os=Linux
    private_net:
        10.11.0.239, 128.162.242.4
    status:
        build=13:25:11 Jan 31 2008
        connected=true
        fencing=Stable
        license:
            have_license=unknown
        member=true
        stable=true
        summary=Stable
        version=5.0.0.1_ALPHA
        wwns:
            100000062b0f5284, 100000062b0f5285
    type=server_admin
```

Note: If a client is not connected to the cluster, the `build` and `version` fields will not display because the node cannot respond to for requests for this information.

Using Prompting Mode

Some `cxfs_admin` commands will prompt you for required attributes if you press `ENTER` after the command name. To see information about the legal values for an attribute, press `<TAB>` after a question.

For example:

```
cxfs_admin:mycluster> create
What do you want to create? The following can be used:
    cluster, failover_net, filesystem, node, switch
create what? node
Specify the attributes for create node:
 name? mynode
 type? client_only
 os?
  AIX       IRIX      Linux     MacOSX    Solaris   Unknown   Windows
 os?
...
```

In basic mode, you are only prompted for required parameters. To be prompted for all possible parameters, use advanced mode. See "Basic and Advanced Mode" on page 223.

Depending upon the context, `cxfs_admin` prompts will vary based upon your answers to previous prompts. For example, if you specify that a node's `os` value is `MacOSX`, `cxfs_admin` will not prompt you for the `type` because Mac OS X nodes are required to be client-only nodes.

To exit from prompt mode, sent an interrupt signal (typically, press `Ctrl-C`).

Command History

The `history` command displays a list of commands that have been used in `cxfs_admin` since it was started:

- Display all of the commands (up to the previous 1000 commands):

  ```
  history
  ```

- Limit the commands to the last specified number of items:

  `history num=`*number_of_items*

  For example, to display only the last 10 commands:

  ```
  cxfs_admin:mycluster> history num=10
  ```

- Clear the history:

  ```
  history clear
  ```

- Send the history to a file (you must enter the full pathname of the file):

  `history output=`*full_pathname*

  For example, to send the history output to the file `/tmp/myhistory`:

  ```
  cxfs_admin:mycluster> history output=/tmp/myhistory
  ```

Waiting for Commands to Complete

Some commands in `cxfs_admin` take a noticeable period of time to complete. `cxfs_admin` displays informational updates as a command progresses or a period character if nothing has changed within 2 seconds.

After 1 minute without change, a command will terminate. This may happen when there is a problem in creating or modifying a node or filesystem. The update message shows the problem status.

To interrupt a command, send an interrupt signal (usually `Ctrl-C`).

Entering `cxfs_admin` Commands on the Command Line

You can enter `cxfs_admin` commands directly from the `cxfs_admin` command line by using the following format:

```
admin# /usr/cluster/bin/cxfs_admin -c "cxfs_admin_commands"
```

For example, to display information about the cluster (line breaks shown for readability):

```
admin# /usr/cluster/bin/cxfs_admin -c "show cluster"
Connecting to the local CXFS server...
cxfs:cluster:
    cc_test2:
        access:
            admin=server
            monitor=
        failover_net:
            10.11.0.0, 128.162.242.0
        filesystem:
            concatfs, mirrorfs, stripefs
        node:
            cc-xe, cc-xe2, aiden, brenna, cc-mac1, cc-vista, cc-win2003, cc-win64, cc-winxp,
            cxfsibm2, cxfssun4, gaeth, liam, lorcan, minnesota, nasca, padraig
        status:
            filesystems:
                summary=concatfs: cxfsibm2 trying to mount
                        stripefs: cxfsibm2 trying to mount
            licenses:
                cxfs_client
            nodes:
                summary=cc-vista: Inactive
                        cc-win64: Inactive
                        cxfsibm2: Mounted 0 of 2 filesystems
            summary=node(s) not stable, filesystem(s) not stable
        switch:
            fcswitch12, fcswitch13
        tiebreaker=minnesota
```

Using Script Files

You can execute a series of cxfs_admin commands by using the -f option and specifying an input file:

```
admin# /usr/cluster/bin/cxfs_admin -f command_file
```

For example, suppose the file /tmp/showme contains the following:

```
cxfs6# more /tmp/showme
show cluster
show filesystem
```

You can execute the following command, which will yield the indicated output: (line breaks shown for readability):

```
# /usr/cluster/bin/cxfs_admin -f /tmp/showme
Connecting to the local CXFS server...
cxfs:cluster:
    cc_test2:
        access:
            admin=server
            monitor=
        failover_net:
            10.11.0.0, 128.162.242.0
        filesystem:
            concatfs, mirrorfs, stripefs
        node:
            cc-xe, cc-xe2, aiden, brenna, cc-mac1, cc-vista, cc-win2003, cc-win64, cc-winxp, cxfsibm2, c
        status:
            filesystems:
                summary=concatfs: cxfsibm2 trying to mount
                        stripefs: cxfsibm2 trying to mount
            licenses:
                cxfs_client
            nodes:
                summary=cc-vista: Inactive
                        cc-win64: Inactive
                        cxfsibm2: Mounted 0 of 2 filesystems
            summary=node(s) not stable, filesystem(s) not stable
        switch:
            fcswitch12, fcswitch13
        tiebreaker=minnesota
```

```
filesystem:
    concatfs:
        forced_unmount=true
        mount=true
        mountpoint=/mnt/concatfs
        nodes:
            aiden, brenna, cc-mac1, cc-vista, cc-win2003, cc-win64, cc-winxp, cc-xe, cc-xe2, cxfsibm2, c
        options=rw
        servers:
            cc-xe, cc-xe2
        status:
            free=918GB
            nodes:
                aiden=mounted
                brenna=mounted
                cc-mac1=mounted
                cc-vista=inactive
                cc-win2003=mounted
                cc-win64=inactive
                cc-winxp=mounted
                cc-xe=mounted
                cc-xe2=mounted
                cxfsibm2=trying to mount
                cxfssun4=mounted
                gaeth=mounted
                liam=mounted
                lorcan=mounted
                minnesota=mounted
                nasca=mounted
                padraig=mounted
            server=cc-xe2
            size=930GB
            summary=cxfsibm2 trying to mount
            utilization=1%
    mirrorfs:
        forced_unmount=true
        mount=false
        mountpoint=/mnt/mirrorfs
        nodes:
            aiden, brenna, cc-mac1, cc-vista, cc-win2003, cc-win64, cc-winxp,
            cc-xe, cc-xe2, cxfsibm2, cxfssun4, gaeth, liam, lorcan, minnesota,
            nasca, padraig
        options=rw
        servers:
            cc-xe, cc-xe2
        status:
            nodes:
                aiden=unmounted
                brenna=unmounted
                cc-mac1=unmounted
                cc-vista=inactive
                cc-win2003=unmounted
                cc-win64=inactive
                cc-winxp=unmounted
                cc-xe=unmounted
                cc-xe2=unmounted
                cxfsibm2=unmounted
                cxfssun4=unmounted
                gaeth=unmounted
                liam=unmounted
                lorcan=unmounted
                minnesota=unmounted
                nasca=unmounted
                padraig=unmounted
            summary=Unmounted
    stripefs:
        forced_unmount=false
        mount=true
        mountpoint=/mnt/stripefs
        nodes:
            aiden, brenna, cc-mac1, cc-vista, cc-win2003, cc-win64, cc-winxp,
            cc-xe, cc-xe2, cxfsibm2, cxfssun4, gaeth, liam, lorcan, minnesota,
            nasca, padraig
        options=rw,dmi
        servers:
            cc-xe, cc-xe2
        status:
            free=3.37TB
            nodes:
                aiden=mounted
                brenna=mounted
                cc-mac1=mounted
                cc-vista=inactive
                cc-win2003=mounted
                cc-win64=inactive
                cc-winxp=mounted
                cc-xe=mounted
                cc-xe2=mounted
                cxfsibm2=trying to mount
                cxfssun4=mounted
                gaeth=mounted
                liam=mounted
                lorcan=mounted
                minnesota=mounted
                nasca=mounted
                padraig=mounted
            server=cc-xe2
            size=3.63TB
            summary=cxfsibm2 trying to mount
            utilization=7%
```

Setting `cxfs_admin` Defaults

You can use one of the following methods to set the defaults for the way `cxfs_admin` behaves and the editor to use within `cxfs_admin`, in the following order of precedence:

1. Use the `set` command within `cxfs_admin`:

   ```
   set
     [editor=emacs|vi]           (emacs)
     [line_wrap=true|false]      (true)
     [mode=basic|advanced]       (basic)
     [stop_on_error=true|false]  (true)
   ```

   For example, to change to `vi`:

   ```
   cxfs_admin:mycluster> set editor=vi
   ```

   Usage notes:

   - `editor` specifies the editor style (`emacs` or `vi`). The default is `emacs`.

- line_wrap specifies the ability to wrap a line at the edge of the current window (true) or no line wrap (false). The default is true.
- mode determines whether all values (advanced) or only those values that are required (basic). The default is basic. See "Basic and Advanced Mode" on page 223.
- stop_on_error will abort a command upon encountering an error (true) or keep going (false). The default is true.

2. Set the following environment variables:

Environment Variable	**Values**
CXFS_ADMIN_CLUSTER_NAME	*clustername*. Setting this value lets you bypass using the -i option if you have multiple clusters using the same public network as the backup CXFS metadata network. There is no default.
CXFS_ADMIN_EDITOR	emacs (default) or vi
CXFS_ADMIN_LINE_WRAP	true (default) or false
CXFS_ADMIN_MODE	basic (default) or advanced
CXFS_ADMIN_STOP_ON_ERROR	true (default) or false

3. Use the .cxfs_admin file in your home directory (as defined by the $HOME environment variable) to set the following:

```
mode=basic|advanced
cluster_name=clustername
editor=emacs|vi
stop_on_error=true|false
line_wrap=true|false
```

Lines within the .cxfs_admin file that begin with the # character or a space are ignored, as are lines that do not contain the = character.

For example, to use the mycluster cluster in advanced mode and the vi editor:

```
# My settings for cxfs_admin:
cluster=mycluster
mode=advanced
editor=vi
```

Setting `cxfs_admin` Access Permissions

The `access` command allows you to specify hosts that have permission to modify the cluster configuration and hosts that have permission to monitor the cluster state:

```
access
  allow=hostname_or_IPaddress_list
     permission=admin|monitor              (monitor, only available with allow)
  deny=server_name
```

By default, all server-capable administration nodes in the cluster are granted `admin` access (without using the `access` command).

For example, to grant `remotehostA` and `remotehostB` permission to modify the cluster configuration:

```
cxfs_admin:mycluster> access allow=remotehostA,remotehostB permission=admin
```

To grant read-only rights in order to monitor to the cluster configuration and status (`monitor` is the default access level):

```
cxfs_admin:mycluster> access allow=remotehostA
```

To revoke all access to the cluster database for a host that was previously granted some level of access, use the following command:

```
cxfs_admin:mycluster> access deny=remotehostA,remotehostB
```

To view the current access rights, use the following command:

```
show access
```

For example:

```
cxfs_admin:mycluster> show access
access:
    admin=server
    monitor=cluster
```

Usage notes:

- `allow` specifies the hosts to be granted the specified permission. These hosts must be on the same private network as the cluster nodes. To specify multiple hosts, use a comma-separated list. There are three reserved hostnames:
  - `cluster` denotes any node defined in the cluster

  - `server` denotes any server-capable administration node, even one that is disabled from CXFS membership (see "Disable a Node with `cxfs_admin`" on page 244)
  - `any` denotes any system that is on the private network

- `permission` specifies read/write access (`admin`) or read-only access (`monitor`). The default is `monitor`.
- `deny` specifies the hosts to be denied all access to the cluster database (for hosts that were previously granted some level of access). To specify multiple hosts, use a comma-separated list. The same reserved hostnames as `allow` apply.

Accessing the Correct Cluster at a Multiple-Cluster Site

If you have multiple clusters using the same public network as the backup CXFS metadata network, use the `-i` option to identify the cluster name. For example:

```
admin# /usr/cluster/bin/cxfs_admin -i mycluster
```

Exiting from cxfs_admin

To exit from prompt mode, sent an interrupt signal (typically, press `Ctrl-C`).

To exit out of the `cxfs_admin` session, enter `exit` or `quit` at the `cxfs_admin` command line:

```
cxfs_admin:mycluster> exit
```

Node Tasks with cxfs_admin

This section discusses the following:

- "Create or Modify a Node with `cxfs_admin`" on page 235
- "Delete a Node with `cxfs_admin`" on page 244
- "Enable a Node with `cxfs_admin`" on page 244
- "Disable a Node with `cxfs_admin`" on page 244
- "Show Node Information with `cxfs_admin`" on page 245

- "Revoke and Restore CXFS Kernel Membership for the Local Node" on page 248
- "Revoke and Restore CXFS Kernel Membership for the Local Node" on page 248

Note: The entire cluster status information is sent to each server-capable administration node each time a change is made to the cluster database; therefore, the more server-capable administration nodes in a configuration, the longer it will take.

Create or Modify a Node with `cxfs_admin`

To define a node, use the following command and attributes (line breaks shown here for readability, defaults in parentheses):

```
create node
  name=nodename
  type=client_only|server_admin          (client_only)
  os=AIX|IRIX|Linux|MacOSX|Solaris|Windows|Unknown    (Linux)
  private_net private_network_IPaddress_list|hostname_list
  Advanced-mode:
  enabled=true|false                                  (true)
  failpolicy=FenceReset,Fence,Reset,Shutdown          (Fence,Shutdown)
  hostname=logical_hostname                  (fully_qualified_domain_name_of_nodename)
  nodeid=nodeID                                      (assigned by cxfs_admin)
  partition_id=partition_number
  reset_method=nmi|powerCycle|reset                    (powerCycle)
  reset_port=l1|l2|bmc|msc|mmsc
  reset_password=password
  reset_status=enabled|disabled                 (enabled)
  reset_node=node_sending_reset_command
  reset_comms=tty|network|ipmi
  reset_device=port|IP_address_or_hostname_of_device
```

When you create a client-only node, it will by default automatically be enabled and join the cluster. When adding the first server-capable administration node, you must restart it or restart CXFS services and cluster services on the node:

```
admin# service /cxfs stop
admin# service /cxfs_cluster stop
```

```
admin# service /cxfs_cluster start
admin# service /cxfs start
```

To use prompting mode, press <ENTER>. To obtain information about legal values, press <TAB>.

For example, to create a client-only node, you could do the following, pressing the <TAB> key to see the list of operating system values:

```
cxfs_admin:mycluster> create node
Specify the attributes for create node:
 name? newnode
 type? client_only
 os? <TAB>
  AIX       IRIX      Linux     MacOSX    Solaris   Unknown   Windows
 os? irix
 private_net? 192.168.0.178
 Node "newnode" has been created, waiting for it to join the cluster...
Waiting for node newnode, current status: Inactive
Waiting for node newnode, current status: Establishing membership
Waiting for node newnode, current status: Probing XVM volumes
Operation completed successfully
```

Note: When you add a Linux node, cxfs_admin must prompt you for the type value because that node could be of type server-capable or client-only.

To create a server-capable administration node using the defaults, you must delete the client_only default for type and enter server_admin. For example:

```
cxfs_admin:mycluster> create node
Specify the attributes for create node:
 name? newnode
 type? server_admin
 private_net? 192.168.0.178
 Node "newnode" has been created, waiting for it to join the cluster...
Please restart all cxfs and cluster services on the server "newnode" to make it
join the cluster.
```

To create a server-capable administration node in advanced mode, which can prompt you to set additional values, such as for `reset_method` and `failpolicy`:

```
cxfs_admin:mycluster> set mode=advanced
cxfs_admin:mycluster> create node
Specify the attributes for create node:
 name? newnode
 type? server_admin
 private_net? 192.168.0.178
 enabled? true
 failpolicy? Reset,Shutdown
 hostname? newnode.example.com
 nodeid? 1
 partition_id?
 reset_method? reset
 reset_port? l2
 reset_password?
 reset_status? enabled
 reset_node? node2
 reset_comms? network
 reset_device? newnode-l2.mycompany.com
Node "newnode" has been created, waiting for it to join the cluster...
Please restart all cxfs and cluster services on the server "newnode" to make it
join the cluster.
```

To modify the `failpolicy` to eliminate `Shutdown`:

```
cxfs_admin:mycluster> modify newnode failpolicy=Reset,Fence
```

Basic-mode usage notes:

- `name` is a simple hostname (such as `lilly`) or a fully qualified domain name (such as `lilly.example.com`) or an entirely different name (such as `node1`). It cannot begin with a number or an underscore (_), or include any whitespace characters, and can be at most 255 characters.
- `type` specifies the function of the node. Enter one of the following:
  - `client_only` is a node that shares CXFS filesystems but will never be a CXFS metadata server. Most nodes should be client-only nodes. `AIX`, `MacOSX`, `Solaris`, `Windows` and `Unknown` nodes are automatically specified as `client-only` and you will not be prompted for this value for these operating systems.

  - server_admin is an SGI ProPack (type linux) node that is a potential CXFS metadata server. (You will use the create filesystem command to define the specific filesystem for which this node can be a metadata server.)

- os is one of the following for client-only nodes:

  AIX
  IRIX
  Linux (SGI ProPack or Linux third-party)
  MacOSX
  Solaris
  Unknown
  Windows

- private_net is the IP address or hostname of the private network. (The hostname must be resolved in the /etc/hosts file.) SGI requires that this network be private; see "Private Network" on page 22.

  There can be up to 8 network interfaces. The order in which you specify the interfaces determines their priority; the first interface will be priority 1, the second priority 2, and so on. There is no default.

  To inform the servers of the failover networks, you must create a failover_net network. See "Network Failover Tasks with cxfs_admin" on page 262.

  For more information about using the hostname, see "Hostname Resolution and Network Configuration Rules" on page 103.

Advanced-mode usage notes:

- enabled determines if a node will be able to obtain CXFS membership (true) or not (false). By default, the new node is enabled (true). To enable a command created with enabled=false, use the enable command. See "Enable a Node with cxfs_admin" on page 244.

- failpolicy determines what happens to a failed node. You can specify up to three methods. The second method will be completed only if the first method fails; the third method will be completed only if both the first and second options fail. Separate options by commas (not whitespace). The option choices are as follows:

  - Fence disables access to the SAN from the problem node. Fencing provides faster recovery of the CXFS kernel membership than reset.

- `FenceReset` performs a fence and then, if the node is successfully fenced, **also** performs an asynchronous reset of the node via a system controller; recovery begins without waiting for reset acknowledgement.

  Note: SGI recommends that a server-capable administration node include `Reset` in its `failpolicy` (unless it is the only server-capable administration node in the cluster). See "Data Integrity Protection" on page 24.

  The `FenceReset` and `Fence` policies are mutually exclusive.

- `Reset` performs a system reset via a system controller. This action requires a `reset_method` value; see "Hardware and Software Requirements for Server-Capable Administration Nodes" on page 34.

- `Shutdown` tells the other nodes in the cluster to wait for a period of time (long enough for the node to shut itself down) before reforming the CXFS kernel membership. (However, there is no notification that the node's shutdown has actually taken place.)

  Caution: Because there is no notification that a shutdown has occurred, if you have a cluster with no tiebreaker, you must not use the `shutdown` setting for any server-capable administration node in order to avoid split clusters being formed. See "Shutdown" on page 55.

  You should not use the `Shutdown` fail policy on client nodes if you choose `dynamic` CXFS kernel heartbeat monitoring for the cluster.

Note: If the failure hierarchy contains `reset` or `fencereset`, the reset might be performed before the system kernel core-dump can complete, resulting in an incomplete core-dump.

For a list of valid `failpolicy` sets, see "Data Integrity Protection" on page 24.

For example, to perform a reset only if a fencing action fails, specify the following:

```
failpolicy=Fence,Reset
```

Note: If you do not specify `Shutdown` and all of the other methods fail, the node attempting to deliver the CXFS kernel membership will stall delivering the membership until either the failed node attempts to re-enter the cluster or the system administrator intervenes using `cms_intervene`. Objects held by the failed node stall until membership finally transitions and initiates recovery.

To perform a fence and an asynchronous reset, specify the following:

```
failpolicy=FenceReset
```

- `hostname` is by the fully qualified hostname. Use the `ping` to display the fully qualified hostname. Do not enter an IP address. The default for `hostname` is the fully qualified domain name for the value of `name`.
- `nodeid` is an integer in the range 1 through 32767 that is unique among the nodes in the cluster. If you change this value after a node has been defined, you must reboot the affected node. You do not normally need to specify this attribute because `cxfs_admin` will calculate an ID for you.
- `partition_id` uniquely defines a partition in a partitioned Origin 3000 system, Altix 3000 series system, or Altix 4700 system. For a non-partitioned system, this attribute is not required (the default unassigned).

Note: For an Origin 3000 series system, use the `mkpart` command to determine this value:

– The `-n` option lists the partition ID (which is 0 if the system is not partitioned).
– The `-l` option lists the bricks in the various partitions (use *rack#.slot#* format in `cxfs_admin`)

For example (output truncated here for readability):

```
irix# mkpart -n
Partition id = 1
irix# mkpart -l
partition: 3 = brick: 003c10 003c13 003c16 003c21 003c24 003c29 ...
partition: 1 = brick: 001c10 001c13 001c16 001c21 001c24 001c29 ...
```

To unset the partition ID, use a value of `0`.

For an Altix 3000, you can find the partition ID by reading the `proc` file. For example:

```
altix# cat /proc/sgi_sn/partition_id
0
```

The `0` indicates that the system is not partitioned. If the system is partitioned, the number of partitions (such as `1`, `2`, etc.) is displayed.

- `reset_method` can be one of the following:
  - `powerCycle` shuts off power to the node and then restarts it
  - `reset` simulates the pressing of the reset button on the front of the machine
  - `nmi` (nonmaskable interrupt) performs a core-dump of the operating system kernel, which may be useful when debugging a faulty machine

  Note: `nmi` is not available on systems containing a baseboard management controller (BMC).

  The default is `powerCycle`.

- `reset_port` is the system controller port type based on the node hardware, as show in Table 11-1 on page 243.
- `reset_password` is the password for the node's system controller port (not the node's `root` password or PROM password). On some nodes, the system administrator may not have set this password. If you wish to set or change the system controller password, consult the hardware manual for your node.
- `reset_status` specifies if the system reset capability is turned on (`enabled`) or turned off (`disabled`). Using `disabled` allows you to provide information about the system controller but temporarily disable reset (meaning that CXFS cannot reset the node). The default for nodes with system controllers is `enabled`, for nodes without system controllers, the default is `disabled`; see "Hardware and Software Requirements for Server-Capable Administration Nodes" on page 34.
- `reset_node` specifies the name of the node that is to send the reset command. It can be the logical name, hostname, or fully qualified domain name. If you use `reset_comms=tty`, serial cables must physically connect the node being defined and the owner node through the system controller port. The node must be defined in the cluster database.

- reset_comms is tty for TTY serial devices, network for network reset to systems with L2 system controllers (you can use network reset even if you have hardware reset capability.), or ipmi for intelligent platform management interface (IPMI) network reset to systems with BMC system controllers.
- reset_device is one of the following:
  - For systems with serial ports (reset_comms=tty), this is the name of the terminal port (TTY) on the owner node (the node issuing the reset). A serial cable connects the terminal port on the owner node to the system controller of the node being reset. /dev/ttyd2 is the most commonly used port, except on Origin 300 and Origin 350 systems (where /dev/ttyd4 is commonly used) and Altix 350 systems (where /dev/ttyIOC0 is commonly used).

    Note: Check the owner node's specific hardware configuration to verify which tty device to use.

  - For systems with network-attached L2 system controllers (reset_comms=network), this is the IP address or hostname of the L2 controller on the node being reset. For example, reset_device=nodename-l2.mycompany.com.
  - For systems with network-attached BMC system controllers (reset_comms=ipmi), this is the IP address or hostname of the BMC controller on the node being reset. For example, reset_device=nodename-bmc.mycompany.com.

For example:

For an Origin 3000 series system:

```
reset_comms=tty reset_device=/dev/ttyd2
```

For an SGI Altix 3000 Bx2 system:

```
reset_comms=network reset_device=nodename-l2.mycompany.com
```

For an Altix 350 system without an L2:

```
reset_comms=tty reset_device=/dev/ttyIOC0
```

For an Altix XE system with a BMC:

```
reset_comms=ipmi reset_device=nodename-bmc.mycompany.com
```

Table 11-1 System Controller Types

`bmc`	`l1`	`l2`	`mmsc`	`msc`
Any Altix XE	Origin/Onyx 300/350	Any Altix with an L2	Rackmount SGI 2400/2800	Origin 200
	Origin/Onyx 3200C	Prism	Onyx2	Onyx2 Deskside
		Origin/Onyx 3000 series		SGI 2100/2200 deskside systems
		Origin 300/350		

The following is an example of a node created for BMC reset, where the default for `reset_password` is `admin` as explained in"BMC System Controller" on page 451 (line breaks added here for readability):

```
cxfs_admin:mycluster> create node name=node1 type=server_admin private_net=192.168.0.168
enabled=true hostname=node1.example.com failpolicy=Fence,Reset,Shutdown
nodeid=1 reset_method=powerCycle reset_port=bmc reset_status=enable
reset_node=node2 reset_comms=ipmi reset_device=node1-bmc.mycompany.com
```

Delete a Node with `cxfs_admin`

To delete a node from the cluster and the cluster database, use the following command:

```
delete [node:]nodename
```

If the node is enabled (which is the default), you must disable it before you delete it. For example, if `mynode` is a unique name in the cluster database:

```
cxfs_admin:mycluster> disable mynode
cxfs_admin:mycluster> delete mynode
```

Note: If you delete an active metadata server, `cxfs_admin` will enter read-only mode. You can use the `lock` or `lock steal=true` to reenter lock mode. For more information, see "Making Changes Safely" on page 222.

Enable a Node with `cxfs_admin`

To allow a disabled node to join the cluster, enter the following:

```
enable [node:]nodename
```

For example, if `node1` is a unique name in the cluster database:

```
cxfs_admin:mycluster> enable node1
```

Disable a Node with `cxfs_admin`

To prevent a node from joining the cluster, enter the following:

```
disable [node:]nodename
```

For example, if `node1` is a unique name in the cluster database:

```
cxfs_admin:mycluster> disable node1
```

Note: This procedure is only recommended as needed for a CXFS server-capable administration node because it updates the cluster database and is therefore intrusive to other nodes. When shutting down a CXFS client–only node, do not disable it. Rather, let the CXFS services stop by themselves when the client-only node is shut down.

After you have disabled a node, the node is no longer an active member of the cluster.

Caution: If you disable a node, it will be marked as `Disabled` and it will therefore not rejoin the cluster after a reboot. To allow a node to rejoin the cluster, you must enable the node. See "Enable a Node with `cxfs_admin`" on page 244.

Show Node Information with `cxfs_admin`

You can display a node's parameters with the following command:

`show [node:]`*nodename*

For example, if `node1` is a unique name in the cluster database:

```
cxfs_admin:mycluster> show node1
node:node1:
    cellid=1
    enabled=true
    os=IRIX
    private_net:
        192.168.0.204
    status:
        client=stable
        connected=true
        fencing=Stable
        filesystems=up
        license:
            cpu_count=1
            have_license=true
            oem=none
            os=IRIX64
            version=4.0.0.2
        membership=up
        summary=Stable
        version=4.0.0.2
        wwns:
            210000e08b081f23
        xvm=up
    type=client_only
```

Note: The `have_license` field reports licenses for client-only nodes that have been obtained from the server. It does not display information about the server licenses. For more information about licensing, see "Show License Information with `cxfs_admin`" on page 252.

You can see a list of all of the nodes that have been defined with the following command:

```
show node
```

For example (output truncated):

```
cxfs_admin:cc_test2> show node
node:
    cc-xe:
        cellid=0
        enabled=true
        os=Linux
        private_net:
            10.11.0.239, 128.162.242.4
        status:
            connected=true
            fencing=Stable
            license:
                have_license=unknown
            summary=Stable
            version=5.0.0.1_ALPHA
            wwns:
                100000062b0f5284, 100000062b0f5285
        type=server_admin
    cc-xe2:
        cellid=1
        enabled=true
        os=Linux
        private_net:
            10.11.0.242, 128.162.242.6
        status:
            connected=true
            fencing=Stable
            license:
                have_license=unknown
            summary=Stable
            version=5.0.0.1_ALPHA
            wwns:
                100000062b0f568c, 100000062b0f568d
        type=server_admin
    aiden:
        cellid=2
        enabled=true
        os=IRIX
        private_net:
            10.11.0.241, 128.162.242.12
        status:
            client=stable
            connected=true
            fencing=Stable
            filesystems=up
            license:
                cpu_count=8
                have_license=true
                oem=none
                os=IRIX64
            membership=up
            summary=Stable
            version=5.0.0.1_ALPHA
            wwns:
                210000e08b01479d, 210000e08b041a3a
            xvm=up
        type=client_only
...
```

See also "Displaying the License Keys with `cxfs_admin`" on page 99.

Revoke and Restore CXFS Kernel Membership for the Local Node

Note: These commands are only effective for the local node (the node on which `cxfs_admin` is running), which must be a server-capable administration node for these specific commands.

You should revoke CXFS kernel membership of the local node only in the case of error, such as when you need to perform a forced CXFS shutdown (see "Shutdown of the Database and CXFS" on page 296).

To revoke CXFS kernel membership for the local node, use the `stop_cxfs` command. For example:

```
cxfs_admin:mycluster> stop_cxfs
```

The result of the `stop_cxfs` command is considered as a node failure by the rest of the cluster. The cluster may then fail due to a loss of CXFS kernel membership quorum or it may reset the failed node. To avoid the reset, modify the local node's definition to disable the system controller status.

To permit the local node to reapply for membership, use the `start_cxfs` command. For example:

```
cxfs_admin:mycluster> start_cxfs
```

Cluster Tasks with `cxfs_admin`

This section discusses the following:

- "Create or Modify a Cluster with `cxfs_admin`" on page 249
- "Create a Tiebreaker with `cxfs_admin`" on page 250
- "Delete a Cluster with `cxfs_admin`" on page 251
- "Display a Cluster with `cxfs_admin`" on page 252
- "Show License Information with `cxfs_admin`" on page 252

Create or Modify a Cluster with `cxfs_admin`

To create the cluster, use the following command (line breaks shown here for readability, defaults in parentheses):

```
create cluster name=clustername
Advanced-mode:
  heartbeat_monitor=dynamic|static      (static)
  id=clusterID
```

For example:

```
cxfs_admin:> create cluster name=mycluster
```

You can use the `modify` command to add a tiebreaker node or change the CXFS kernel heartbeat monitor type. (You cannot change the cluster's name or ID.)

```
modify clustername
  tiebreaker=client_only_nodename
Advanced-mode:
  heartbeat_monitor=dynamic|static
```

For example, if `mycluster` is a unique name in the cluster database, to make the client-only node `clientA` the CXFS tiebreaker:

```
cxfs_admin:mycluster> modify mycluster tiebreaker=clientA
```

Basic-mode usage notes:

- *clustername* is the logical name of the cluster. It cannot begin with a number or an underscore (_), or include any whitespace characters, and can be at most 255 characters.

  Note: In basic mode, you are not prompted for a device name. Instead, `cxfs_admin` uses the value for `name` and prepends `/dev/cxvm/` to it.

- `tiebreaker` specifies the CXFS tiebreaker. See "Create a Tiebreaker with `cxfs_admin`" on page 250.

Advanced-mode usage notes:

- `heartbeat_monitor` specifies how CXFS kernel membership is monitored. All nodes send CXFS kernel heartbeat messages once per second. If a node does not receive a heartbeat within a defined period, that node loses membership and is denied access to the cluster's filesystems. The defined period is one of the following:

- `static`: Monitors constantly at 1-second intervals and declares a timeout after 5 consecutive missed seconds (default).
- `dynamic`: Starts monitoring only when the node is processing a message from another node (such as for token recall or XVM multicast) or when the client monitors the server because it has a message pending (for example, a token acquire or metadata operation). Once monitoring initiates, it monitors at 1-second intervals and declares a timeout after 5 consecutive missed seconds, just like static monitoring. Dynamic heartbeat monitoring is appropriate for clusters that have clients with heavy workloads; using it avoids inappropriate loss of membership. However, it may take longer to recover a client's tokens and other state information when there is an actual problem.

Note: You should not use the `Shutdown` fail policy on client nodes if you choose `dynamic` heartbeat monitoring for the cluster.

- `id` is a unique number within your network in the range 1 through 255. The cluster ID is used by the operating system kernel to make sure that it does not accept cluster information from any other cluster that may be on the network. The kernel does not use the database for communication, so it requires the cluster ID in order to verify cluster communications. This information in the kernel cannot be changed after it has been initialized; therefore, you cannot change a cluster ID after the cluster has been defined. Clusters must have unique IDs.

Create a Tiebreaker with `cxfs_admin`

The CXFS tiebreaker node determines whether a CXFS kernel membership quorum is maintained when exactly half of the server-capable administration nodes can communicate with each other. There is no default CXFS tiebreaker.

Caution: SGI recommends that you use client-only nodes as tiebreakers to ensure that the cluster remains operational. `cxfs_admin` will only let you specify a server-capable administration node as a tiebreaker if the cluster contains four or more server-capable administration nodes, and an even number of server-capable administration nodes.

The reset capability or I/O fencing with switches is **mandatory** to ensure data integrity for all nodes. Clusters should have an odd number of server-capable administration nodes. If you have an even number of server-capable administration nodes, define a CXFS tiebreaker node. (See "CXFS Recovery Issues in a Cluster with Only Two Server-Capable Administration Nodes " on page 443.)

To set the CXFS tiebreaker node, use the `modify` command as follows:

`modify [cluster:]`*clustername* `tiebreaker=`*client_nodename*

For example:

```
cxfs_admin:mycluster> modify mycluster tiebreaker=myclient
```

To unset the CXFS tiebreaker node, do not supply a value for `tiebreaker`. For example:

```
cxfs_admin:mycluster> modify mycluster tiebreaker=
```

Delete a Cluster with `cxfs_admin`

To delete a cluster, use the following command:

`delete [cluster:]`*clustername*

For example, if `mycluster` is a unique name in the cluster database:

```
cxfs_admin:mycluster> delete mycluster
```

However, you cannot delete an active cluster; you must first unmount and delete the filesystems, disable and delete the nodes, and so on.

Display a Cluster with `cxfs_admin`

To display the cluster, use the following command:

```
show cluster
```

For example:

```
cxfs_admin:cc_test2> show cluster
cxfs:cluster:
    cc_test2:
        access:
            admin=server
            monitor=
        failover_net:
            10.11.0.0, 128.162.242.0
        filesystem:
            concatfs, mirrorfs, stripefs
        node:
            cc-xe, cc-xe2, aiden, brenna, cc-mac1, cc-vista, cc-win2003,
                    cc-win64, cc-winxp, cxfsibm2, cxfssun4, gaeth, liam,
                    lorcan, minnesota, nasca, padraig
        status:
            filesystems:
                summary=concatfs: cxfsibm2 trying to mount
                        stripefs: cxfsibm2 trying to mount
            licenses:
                cxfs_client
            nodes:
                summary=cc-vista: Inactive
                        cc-win64: Inactive
                        cxfsibm2: Mounted 0 of 2 filesystems
            summary=node(s) not stable, filesystem(s) not stable
        switch:
            fcswitch12, fcswitch13
        tiebreaker=minnesota
```

Show License Information with `cxfs_admin`

To show the CXFS licenses available for the cluster, use the following command:

```
show licenses
```

For example:

```
cxfs_admin:mycluster> show licenses
status:licenses:
    cxfs_client:
        enterprise:
            allocated=17
            valid=22
        workstation:
            allocated=4
            valid=15
```

CXFS Filesystem Tasks with `cxfs_admin`

The `filesystem` class represents the clustered XVM volumes that can be mounted by CXFS nodes. Before you can create a filesystem definition, you must create the clustered XVM volume and make the filesystem with `mkfs`.

By default, the filesystem:

- Uses the XVM device of the same name
- Enables all nodes to mount the filesystem
- Mounts the filesystem in `/mnt/`
- Is not managed by GRIOv2

To override these defaults, use the optional attributes listed below.

This section discusses the following:

- "Create or Modify a CXFS Filesystem with `cxfs_admin`" on page 254
- "Mount a CXFS Filesystem with `cxfs_admin`" on page 259
- "Unmount a CXFS Filesystem with `cxfs_admin`" on page 259
- "Relocate the Metadata Server for a Filesystem with `cxfs_admin`" on page 260
- "Delete a CXFS Filesystem with `cxfs_admin`" on page 260
- "Show a CXFS Filesystem" on page 260

Create or Modify a CXFS Filesystem with cxfs_admin

Use the following commands to define a filesystem and the nodes on which it may be mounted (line breaks shown here for readability, defaults in parentheses):

```
create filesystem name=filesystemname
  [options=mount_options]
  [forced_unmount=true|false]                    (false)
  [mountpoint=mountpoint]                   (/mnt/filesystemname)
  [mounted=true|false]                        (true)
Advanced-mode:
  [device=devicename]                       (filesystemname)
  [servers=server_list]                     (all servers are potential MDS)
  [nodes=nodes_that_can_mount]             (all nodes can mount)
  [mount_new_nodes=true|false]                (true)
  [grio_managed=true|false]            (false)
  [grio_qual_bandwidth=qualified_bandwidth]
```

Note: Relocation is disabled by default. Recovery and relocation are supported only when using standby nodes. Therefore, you should only define multiple metadata servers for a given filesystem if you are using the standby node model. See "Relocation" on page 25.

Basic-mode usage notes:

- name specifies the name of the filesystem:

  Note: You must create the CXFS filesystem with xvm before you set it up using cxfs_admin.

  – If you also specify a value for device, then name can be any string that does not begin with a number or an underscore (_), or include any whitespace characters, and can be at most 255 characters.For example, if the full XVM volume name is /dev/cxvm/concat1:

```
cxfs_admin:mycluster> create filesystem name=filesys1 device=concat1
```

  - If you do not specify a value for `device`, then `name` must be the name of the XVM volume following `/dev/cxvm`. For example:

```
cxfs_admin:mycluster> create filesystem name=concat1
```

Note: Within the GUI, the default is to use the last portion of the device name; for example, for a device name of `/dev/cxvm/d76lun0s0`, the GUI will automatically supply a logical filesystem name of `d76lun0s0`. The GUI will accept other logical names defined with `cxfs_admin` but the GUI will not allow you to modify a logical name; you must use `cxfs_admin` to modify the logical name.

- `options` specifies the mount options that are passed to the `mount` operating system command. These mount options control access to the specified filesystem. For a list of supported mount options, see the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*. By default, this is unassigned.

  Specify multiple mount options as a comma-separated list. For example, the following specifies that the `myfs` filesystem uses `inode64` allocation and does not update the access time stamps for files and directories:

```
cxfs_admin:mycluster> create filesystem name=myfs options=inode64,noatime
```

Note: No validation is done on the mount options in `cxfs_admin`, so an invalid option may prevent the filesystem mounting on all nodes.

- `forced_unmount` controls the action that CXFS takes if there are processes that have open files or directories in the filesystem to be unmounted:
  - If set to `true`, the processes will be killed and the unmount will occur
  - If set to `false`, the processes will not be killed and the filesystem will unmount only after all references to the filesystem have been closed (default)
- `mounted` specifies whether a new filesystem is mounted on all nodes in the cluster (`true`) or not mounted on any nodes (`false`). By default, the new filesystem is mounted on all nodes (`true`).
- `mountpoint` specifies a mount point for the filesystem. The mount point is a directory to which the XVM volume is attached. This directory name must begin with a slash (`/`). The default is `/mnt/`*filesystemname*.

For example, to create a filesystem named myfs and use the default mount point of /mnt/myfs:

```
cxfs_admin:mycluster> create filesystem name=myfs
```

To create the myfs filesystem but use a mount point of /tmp/myfs:

```
cxfs_admin:mycluster> create filesystem name=myfs mountpoint=/tmp/myfs
```

Advanced-mode usage notes:

- device is the device name for an XVM volume. The default is the filesystem name specified by name.

  Note: Specify only the XVM volume name itself. Do not include /dev/cxvm/.

  For example, to create a device name of mydev for the myfs filesystem:

```
cxfs_admin:mycluster> create filesystem name=myfs device=mydev
```

- servers specifies the potential metadata servers that can serve the filesystem to the cluster. To specify multiple server capable administration nodes, use a comma-separated list of node names. The default is all server-capable administration nodes in the cluster.

  Note: The list of potential metadata servers for a given filesystem is ordered, but because of network latencies and other unpredictable delays, it is impossible to predict which node will become the active metadata server.

  For example, to specify that either node2 or node3 could be the metadata server, with node2 being the primary server, for the myfs filesystem:

```
cxfs_admin:mycluster> create filesystem name=myfs servers=node2,node3
```

- nodes specifies the only nodes that can mount the filesystem as a specified comma-separated list. If you do not specify nodes on the create command, all nodes can mount the filesystem. If you restrict the nodes on the create command line, you can later mount all nodes by specifying all of them with the nodes attribute.

  For example, to restrict mounting the myfs filesystem to nodes node1 and node2:

  ```
  create myfs nodes=node1,node2
  ```

To add node3:

```
modify myfs nodes=node1,node2,node3
```

- mount_new_nodes specifies whether a newly created node will automatically mount the filesystem when it gets membership (true) or will not mount the filesystem (false). By default, new nodes mount all defined filesystems.

  For example, to create filesystem myfs that is not automatically mounted by new nodes, use the following command:

```
cxfs_admin:mycluster> create filesystem name=myfs  mount_new_nodes=false
```

  To later mount the filesystem on node3 after it has been created, use the following command:

```
cxfs_admin:mycluster> mount myfs nodes=node3
```

- grio_managed specifies whether a filesystem is managed by GRIOv2 (true) or not (false). The default is false. Setting grio_managed to false disables GRIO management for the specified filesystem, but it does not reset the grio_qual_bandwidthvalue. In this case, grio_qual_bandwidth is left unmodified in the cluster database and ignored.
- grio_qual_bandwidth specifies a filesystem's qualified bandwidth in bytes (B suffix), kilobytes (KB), megabytes (MB), or gigabytes (GB), where the units are multiples of 1024. The default is MB for 4000 or less, B for 4001 or greater. If the filesystem is GRIO-managed, you must specify a qualified bandwidth with this attribute. You can modify the qualified bandwidth for a mounted filesystem without taking it offline.

  For example, the following commands all create the myfs filesystem with a GRIOv2 qualified bandwidth of 1.2 GB/s (assuming that grio_managed=true has already been set):

```
cxfs_admin:mycluster> create filesystem name=myfs grio_qual_bandwidth=1288500000
cxfs_admin:mycluster> create filesystem name=myfs grio_qual_bandwidth=1258300KB
cxfs_admin:mycluster> create filesystem name=myfs grio_qual_bandwidth=1288.8MB
cxfs_admin:mycluster> create filesystem name=myfs grio_qual_bandwidth=1.2GB
```

For example, using prompting in basic mode:

```
cxfs_admin:mycluster> create filesystem
 name? myfs
 options? rw
 forced_unmount? false
 mountpoint? /mnt/myfs
 mounted? true
Filesystem "myfs" has been created, waiting for it to be mounted on all
assigned nodes...
Waiting for filesystem myfs, current status: A server is trying to mount
Waiting for filesystem myfs, current status: node1 trying to mount, node2
trying to mount
Waiting for filesystem myfs, current status: node1 trying to mount
Operation completed successfully
```

For example, using prompting in advanced mode:

```
cxfs_admin:mycluster> create filesystem
Specify the attributes for create filesystem:
 name? myfs
 options? rw
 forced_unmount? false
 mountpoint? /mnt/myfs
 device? myfs
 servers? node1,node2,node3
 nodes? node1,node2,node3,node4
 mounted? true
 mount_new_nodes? true
 grio_managed? false
Filesystem "myfs" has been created, waiting for it to be mounted on all
assigned nodes...
Waiting for filesystem myfs, current status: A server is trying to mount
Waiting for filesystem myfs, current status: node1 trying to mount, node2
trying to mount,node3 trying to mount, node4 trying to mount
Waiting for filesystem myfs, current status: node1 trying to mount
Operation completed successfully
```

Note: After a filesystem has been defined in CXFS, running `mkfs` on it will cause errors to appear in the system log file. To avoid these errors, run `mkfs` before defining the filesystem in CXFS, or delete the CXFS filesystem before running `mkfs`. See "Delete a CXFS Filesystem with `cxfs_admin`" on page 260.

Mount a CXFS Filesystem with `cxfs_admin`

The mount command operates on the set of nodes that were specified in the `nodes=`*nodelist* attribute when the filesystem was created. By default, this is all nodes in the cluster.

To mount the filesystem on all enabled nodes in the cluster:

`mount` *filesystem*

To mount the filesystem on specific enabled nodes:

`mount` *filesystem* `nodes=`*nodelist*

For example, to mount the filesystem `myfs` on only nodes `node2` and `node3`:

```
cxfs_admin:mycluster> mount myfs nodes=node2,node3
```

Note: If any nodes that are set to mount the filesystem are enabled and attached are not in membership, mounting a filesystem will not completely succeed. The filesystem will be marked as ready to be mounted when the nodes achieve membership.

Unmount a CXFS Filesystem with `cxfs_admin`

To unmount a filesystem from all nodes in the cluster:

`unmount` *filesystem*

To unmount the filesystem from a specific comma-separated list of nodes:

`unmount` *filesystem* `nodes=`*nodelist*

For example, to unmount filesystem `myfs` from nodes `node1` and `node3`:

```
cxfs_admin:mycluster> unmount myfs nodes=node1,node3
```

Note: If any nodes are not in membership, the filesystem will be marked as not to be mounted when the nodes achieve membership.

Relocate the Metadata Server for a Filesystem with cxfs_admin

The `relocate` command forcefully moves a filesystem’s metadata server to another node in the cluster that has already been defined as a potential metadata server for that filesystem. This action is typically used to free a server so it can be brought down for maintenance or upgrades. Relocation must also be explicitly enabled in the kernel with the `cxfs_relocation_ok` system tunable parameter (see "Relocation" on page 25).

If relocation is explicitly enabled in the kernel, you can relocate a metadata server to another node by using the following command:

```
relocate filesystem server=new_metadata_server
```

For example:

```
cxfs_admin:mycluster> relocate myfs server=node2
```

CXFS kernel membership is not affected by relocation. However, users may experience a degradation in filesystem performance while the metadata server is relocating.

Delete a CXFS Filesystem with cxfs_admin

Use the following command to delete a filesystem:

```
delete [filesystem:]filesystem
```

You cannot delete a mounted filesystem; you must first unmount it. For example, if `myfs` is a unique name in the cluster database:

```
cxfs_admin:mycluster> unmount myfs
cxfs_admin:mycluster> delete myfs
```

Show a CXFS Filesystem

To show information about all filesystems:

```
show filesystem
```

To show information about a specific filesystem:

```
show [filesystem:]filesystemname
```

For example:

```
cxfs_admin:cc_test2> **show stripefs**
filesystem:stripefs:
    forced_unmount=false
    mount=true
    mountpoint=/mnt/stripefs
    nodes:
        aiden, brenna, cc-mac1, cc-vista, cc-win2003, cc-win64, cc-winxp,
                cc-xe, cc-xe2, cxfsibm2, cxfssun4, gaeth, liam, lorcan,
                minnesota, nasca, padraig
    options=rw,dmi
    servers:
        cc-xe, cc-xe2
    status:
        free=3.37TB
        nodes:
            aiden=mounted
            brenna=mounted
            cc-mac1=mounted
            cc-vista=inactive
            cc-win2003=mounted
            cc-win64=inactive
            cc-winxp=mounted
            cc-xe=mounted
            cc-xe2=mounted
            cxfsibm2=trying to mount
            cxfssun4=mounted
            gaeth=mounted
            liam=mounted
            lorcan=mounted
            minnesota=mounted
            nasca=mounted
            padraig=mounted
        server=cc-xe2
        size=3.63TB
        summary=cxfsibm2 trying to mount
        utilization=7%
```

Network Failover Tasks with cxfs_admin

To allow the cluster to continue operation if the primary private network fails, you can set up private network failover.

To inform the servers of the failover networks, you must create a failover_net network.

Caution: Do not create a failover_net network while CXFS services are active; doing so can lead to cluster malfunction.

Each node in the cluster must have all private_net values specified to match the subsets defined by the failover networks in the same order as all other nodes in the cluster.

Command syntax:

```
create failover_net network=IPaddress
mask=IPmask
```

To create two private networks, one on the 192.168.0.*x* and the other on the 10.0.0.*x* subnets, use the following command:

```
cxfs_admin:mycluster> create failover_net network=192.168.0.0 mask=255.255.255.0
cxfs_admin:mycluster > create failover_net network=10.0.0.0 mask=255.255.255.0
```

To create a node with failover network support:

```
cxfs_admin:mycluster> create node name=mynode private_net=192.168.0.2,10.0.0.2
```

Switch Tasks with cxfs_admin

This section discusses the following:

- "Create a Switch with cxfs_admin" on page 263
- "Delete a Switch Definition with cxfs_admin" on page 265
- "Show Switches with cxfs_admin" on page 265

For general information, see "I/O Fencing" on page 51.

Note: Nodes without system controllers require I/O fencing to protect data integrity. A switch is mandatory to support I/O fencing; therefore, multiOS CXFS clusters require a switch. See the release notes for supported switches.

To raise or lower a fence, or update switch port information, use the `hafence` command.

Create a Switch with `cxfs_admin`

To define a new switch, use the following command:

```
create switch name=switch_hostname
  [user=username]                                  (admin)
  [password=username_password]                     (password)
  [vendor=brocade|qlogic|site-specific_vendor]     (brocade)
Advanced-mode:
  [mask=ports_that_will_not_be_fenced]
```

Note: You must define all of the switches within your fabric to which a CXFS client or server is connected.

Basic-mode usage notes:

- `name` specifies the hostname of the Fibre Channel switch; this is used to determine the IP address of the switch.
- `password` specifies the password for the specified *username*. The default is `password`.
- `user` specifies the user name to use when sending a `telnet` message to the switch. The default is `admin`.
- `vendor` specifies the vendor of the Fibre Channel switch. It can be one of the following values:

  `brocade` (default)
  `qlogic`
  site-specific-value

For example, if `myswitch` is a QLogic switch:

```
cxfs_admin:mycluster> create switch name=myswitch vendor=qlogic
```

Advanced-mode usage notes:

- `mask` specifies the ports on the switch that will never be fenced. By default, no ports are masked (and therefore all ports are available for fencing). The value for `mask` is a series of comma-separated port ranges. For example, the following states that ports 0, 4 and 12 to 15 for `myswitch` will never be fenced by CXFS:

```
cxfs_admin:mycluster> create switch name=myswitch mask=0,4,12-15
```

Note: For the bladed Brocade 48000 switch (where the port number is not unique), the value you should use for `mask` is the `Index` value that is displayed by the `switchShow` command. For example, the `switchShow` output below indicates that you would use a `mask` value of `16` for port `0` in slot `2`:

```
brocade48000:admin> switchShow
Index Slot Port Address Media Speed State     Proto
===================================================
  0    1    0   010000   id    N4   Online    F-Port  10:00:00:00:c9:5f:9b:ea
  1    1    1   010100   id    N4   Online    F-Port  10:00:00:00:c9:5f:ab:d9
....
142    1   30   018e00   id    N4   Online    F-Port  50:06:0e:80:04:5c:0b:46
143    1   31   018f00   id    N4   Online    F-Port  50:06:0e:80:04:5c:0b:66
 16    2    0   011000   id    N4   Online    F-Port  10:00:00:00:c9:5f:a1:f5
 17    2    1   011100   id    N4   Online    F-Port  10:00:00:00:c9:5f:a1:72
...
```

Server-capable administration nodes automatically discover the available HBAs and, when fencing is triggered, fence off all of the Fibre Channel HBAs when the `Fence` or `FenceReset` fail policy is selected. However, masked HBAs will not be fenced. Masking allows you to prevent the fencing of devices that are attached to the SAN but are not shared with the cluster, to ensure that they remain available regardless of CXFS status. You would want to mask HBAs used for access to tape storage, or HBAs that are only ever used to access local (nonclustered) devices.

Delete a Switch Definition with `cxfs_admin`

To delete a switch, use the following command:

`delete [switch:]`*switch_hostname*

For example, if `myswitch` is a unique name in the cluster database:

```
cxfs_admin:mycluster> delete myswitch
```

Show Switches with `cxfs_admin`

To display all of the switches in the system, use the following command:

`show switch [output=`*full_pathname*`]`

For example, in basic mode:

```
cxfs_admin:cc_test2> show switch
switch:
    fcswitch12:
        hostname=fcswitch12
        num_ports=32
        port:
            0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
                    18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31
        vendor=brocade
    fcswitch13:
        hostname=fcswitch13
        num_ports=32
        port:
            0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
                    18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31
        vendor=brocade
```

To send the output to the `/tmp/switchinfo` file:

```
cxfs_admin:mycluster> show switch output=/tmp/switchinfo
```

To display a specific switch:

`show [switch:]`*switchname* `[output=`*full_pathname*`]`

To display mask values, use advanced mode (see "Basic and Advanced Mode" on page 223.) For example, if myswitch is a unique name in the cluster database:

```
cxfs_admin:cc_test2> show fcswitch12
switch:fcswitch12:
    hostname=fcswitch12
    mask=
    num_ports=32
    port:
        0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
                19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31
    switchid=Switch1
    vendor=brocade
```

To display the switches and ports each host is connected to:

```
show wwns
```

For example:

```
cxfs_admin:cc_test2> show wwns
node:cc-xe:status:wwns:
    100000062b0f5284:
        order=1
        switch=fcswitch12
        switch_port=11
    100000062b0f5285:
        order=0
        switch=fcswitch13
        switch_port=11
node:cc-xe2:status:wwns:
    100000062b0f568c:
        order=1
        switch=fcswitch13
        switch_port=20
    100000062b0f568d:
        order=0
        switch=fcswitch12
        switch_port=20
node:aiden:status:wwns:
    210000e08b01479d:
        order=1
        switch=fcswitch12
        switch_port=19
    210000e08b041a3a:
        order=0
        switch=fcswitch13
        switch_port=17
node:brenna:status:wwns:
    210000e08b128dcc:
        order=0
        switch=fcswitch13
        switch_port=15
    210100e08b328dcc:
        order=1
        switch=fcswitch12
        switch_port=15
node:cc-mac1:status:wwns:
    100000062b105528:
        order=3
    100000062b105529:
        order=2
        switch=fcswitch13
        switch_port=14
node:cc-win2003:status:wwns:
    210000e08b12767a:
        order=0
        switch=fcswitch13
        switch_port=3
    210100e08b32767a:
        order=1
        switch=fcswitch12
        switch_port=3
node:cc-winxp:status:wwns:
    210000e08b12e0c9:
        order=0
        switch=fcswitch13
        switch_port=2
    210100e08b32e0c9:
        order=1
        switch=fcswitch12
        switch_port=2
node:cxfsibm2:status:wwns:
    10000000c9323a1b:
```

```
                order=0
                switch=fcswitch12
                switch_port=16
        node:cxfssun4:status:wwns:
            100000062b1169b5:
                order=0
                switch=fcswitch13
                switch_port=19
        node:gaeth:status:wwns:
            100000062b0e1ca8:
                order=1
                switch=fcswitch12
                switch_port=27
            100000062b0e1ca9:
                order=0
                switch=fcswitch13
                switch_port=27
        node:liam:status:wwns:
            210000e08b1cad7e:
                order=0
                switch=fcswitch12
                switch_port=25
            210100e08b3cad7e:
                order=1
                switch=fcswitch13
                switch_port=25
        node:lorcan:status:wwns:
            100000062b0e41d8:
                order=1
                switch=fcswitch12
                switch_port=13
            100000062b0e41d9:
                order=0
                switch=fcswitch13
                switch_port=13
        node:minnesota:status:wwns:
            210000e08b087a3f:
                order=0
                switch=fcswitch12
                switch_port=0
            210100e08b287a3f:
```

```
        order=1
        switch=fcswitch13
        switch_port=0
node:nasca:status:wwns:
    210000e08b0e879e:
        order=0
        switch=fcswitch13
        switch_port=18
    210100e08b2e879e:
        order=1
        switch=fcswitch12
        switch_port=18
node:padraig:status:wwns:
    100000062b0ec480:
        order=1
        switch=fcswitch12
        switch_port=24
    100000062b0ec481:
        order=0
        switch=fcswitch13
        switch_port=24
```

To show full status details for each port on the switch, use one of the following commands:

```
show [switch:]switchname all
show switchname:port
```

For example, for the switch named `fcswitch12`:

```
cxfs_admin:mycluster> show fcswitch12:port
```

Saving and Recreating the Current Configuration with `cxfs_admin`

The `config` command displays a series of commands that represent the current configuration of the objects specified. You can use this output to recreate the configuration of the entire cluster or a subset of it.

By default, `config` displays information at the `cxfs_admin` prompt. To write the configuration output to a file, use the `output` attribute and specify the full pathname of the file to contain the information:

`config node output=`*full_pathname*

You can use the generated file with the `-f` command line option to recreate the configuration at a later time.

Note: You must modify the output so that the first node created is the server-capable administration node from which you are going to execute the `cxfs_admin` command to create the cluster. (By default, the `cxfs_admin config` command output lists nodes in alphabetical order by node name without regard to node type.)

For a more readable configuration output (without the related commands), use the `show` command rather than the `config` command.

For example, to display all node configuration commands:

```
config node
```

For example (blank lines and line breaks added here for readability):

```
cxfs_admin:mycluster> config node

create node name=node1 os=Linux type=server_admin private_net=192.168.0.168
enabled=true hostname=node1.example.com failpolicy=Fence,Reset,Shutdown
nodeid=1 reset_method=powerCycle reset_port=bmc reset_status=disabled reset_node=node2
reset_comms=ipmi reset_device=node1-bmc.mycompany.com

create node name=node2 os=Linux type=server_admin private_net=192.168.0.185
enabled=true hostname=node2.example.com failpolicy=Fence,Shutdown nodeid=2

create node name=node3 os=IRIX type=client_only private_net=192.168.0.204
enabled=true hostname=node3.example.com failpolicy=Fence,Shutdown nodeid=3

create node name=node4 os=Linux type=server_admin private_net=128.162.232.79
enabled=true hostname=node4.example.com failpolicy=Fence,Shutdown nodeid=4
```

To display the configuration commands for a specific node:

`config [node:]`*nodename*

For example, if the name `node3` is unique in the database:

```
cxfs_admin:mycluster> config node3
create node name=node3 os=IRIX type=client_only private_net=192.168.0.204
enabled=true hostname=node3.example.com failpolicy=Fence,Shutdown nodeid=3
```

To dump the entire cluster configuration to the `/tmp/config.txt` file (where `*` denotes all objects):

```
cxfs_admin:mycluster> config * output=/tmp/config.txt
```

Note: You must give the absolute pathname for the `output` file.

Following is an example of a file `/tmp/buildcluster` that creates a cluster named `mycluster` with two server-capable administration nodes (`mds1` and `mds2`) and a client of each OS type:

Caution: Line breaks and indentations added in this guide for readability. Each `cxfs_admin` command must actually be on one line.

```
create cluster name=mycluster id=1 heartbeat_monitor=static
create node name=mds1 os=Linux type=server_admin
  private_net=10.11.0.239,128.162.242.4 enabled=true hostname=mds1
  failpolicy=Reset nodeid=2 reset_method=powerCycle reset_port=bmc
  reset_status=enabled reset_node=mds2 reset_comms=ipmi
  reset_device=bmc-mds1
create node name=mds2 os=Linux type=server_admin
  private_net=10.11.0.242,128.162.242.6 enabled=true hostname=mds2
  failpolicy=Reset nodeid=1 reset_method=powerCycle reset_port=bmc
  reset_status=enabled reset_node=mds1 reset_comms=ipmi
  reset_device=bmc-mds2
create node name=mac-client os=MacOSX type=client_only
  private_net=10.11.0.150,128.162.242.193 enabled=true hostname=mac-client
  failpolicy=Fence,Shutdown nodeid=9
create node name=windows-client os=Windows type=client_only
  private_net=10.11.0.166,128.162.242.241 enabled=true hostname=windows-client
  failpolicy=Fence,Shutdown nodeid=6
create node name=aix-client os=AIX type=client_only
  private_net=10.11.0.48,128.162.242.197 enabled=true hostname=aix-client
  failpolicy=Fence,Shutdown nodeid=12
```

```
create node name=solaris-client os=Solaris type=client_only
  private_net=10.11.0.60,128.162.242.196 enabled=true hostname=solaris-client
  failpolicy=Fence,Shutdown nodeid=13
create node name=linux-client os=Linux type=client_only
  private_net=10.11.0.253,128.162.242.226 enabled=true hostname=linux-client
  failpolicy=Fence,Shutdown nodeid=4
create node name=irix-client os=IRIX type=client_only
  private_net=10.11.0.52,128.162.242.200 enabled=true hostname=irix-client
  failpolicy=Fence,Shutdown nodeid=8
modify cluster:mycluster tiebreaker=linux-client
create failover_net network=10.11.0.0 mask=255.255.255.0
create failover_net network=128.162.242.0 mask=255.255.255.0
create switch name=fcswitch12  mask= vendor=brocade
create filesystem name=concatfs device=concatfs mountpoint=/mnt/concatfs
  options=rw,dmi servers=mds1,mds2
  nodes=mac-client,windows-client,mds1,mds2,aix-client,solaris-client,linux-client,irix-client
  forced_unmount=false mounted=true mount_new_nodes=true
create filesystem name=mirrorfs device=mirrorfs mountpoint=/mnt/mirrorfs
  options=rw,dmi servers=mds1,mds2
  nodes=mac-client,windows-client,mds1,mds2,aix-client,solaris-client,linux-client,irix-client
  forced_unmount=false mounted=true mount_new_nodes=true
create filesystem name=stripefs device=stripefs mountpoint=/mnt/stripefs
  options=rw,dmi servers=mds1,mds2
  nodes=mac-client,windows-client,mds1,mds2,aix-client,solaris-client,linux-client,irix-client
  forced_unmount=false mounted=true mount_new_nodes=true
```

To use this file to recreate the cluster, first clear the existing cluster configuration and then send the file to `cxfs_admin`:

- Clear the cluster database. See "Clearing the Cluster Database" on page 409.
- Recreate the cluster:

```
admin# /usr/cluster/bin/cxfs_admin -s -f /tmp/buildcluster
Connecting to the local CXFS server...
Node "mds1" has been created, waiting for it to join the cluster...
Please restart all cxfs and cluster services on the server "mds1" to make it join the cluster.
Node "mds2" has been created, waiting for it to join the cluster...
Please restart all cxfs and cluster services on the server "mds2" to make it join the cluster.
Node "mac-client" has been created, waiting for it to join the cluster...
Node "windows-client" has been created, waiting for it to join the cluster...
Node "aix-client" has been created, waiting for it to join the cluster...
Node "solaris-client" has been created, waiting for it to join the cluster...
Node "linux-client" has been created, waiting for it to join the cluster...
Node "irix-client" has been created, waiting for it to join the cluster...
Filesystem "concatfs" has been created, waiting for it to be mounted on all assigned nodes...
Filesystem "mirrorfs" has been created, waiting for it to be mounted on all assigned nodes...
Filesystem "stripefs" has been created, waiting for it to be mounted on all assigned nodes...
Waiting for configuration update to complete
Operation completed successfully
```

Chapter 12

Administration and Maintenance

This chapter discusses the following topics:

- "XVM Volume Mapping to Storage Targets" on page 314
- "XVM Failover V2" on page 314
- "GPT Labels" on page 322
- "Generation of Streaming Workload for Video Streams" on page 323
- "Frame Files Defragmentation and Analysis" on page 324

See also Chapter 13, "Cluster Database Management" on page 331.

Administrative Tasks

You will use one of the following commands for CXFS administration:

- CXFS GUI connected to any server-capable administration node. See Chapter 10, "Reference to GUI Tasks" on page 147.
- `cxfs_admin`: for most commands, from any node in the cluster network that has administrative permission (see "Setting `cxfs_admin` Access Permissions" on page 233). See Chapter 11, "Reference to `cxfs_admin` Tasks" on page 217.

  Note: A few `cxfs_admin` commands, such as `stop_cxfs`, must be run from a server-capable administration node.

Precedence of Configuration Options

Figure 12-1 shows the order in which CXFS programs take their configuration options.

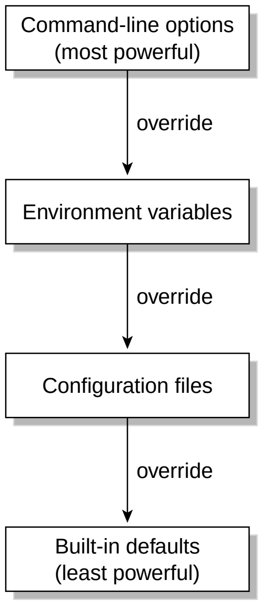


Figure 12-1 Precedence of Configuration Options

CXFS Release Versions and Rolling Upgrades

SGI lets you upgrade of a subset of nodes from *X.anything* to *X.anything* within the same major-release thread (*X*). This policy lets you to keep your cluster running and filesystems available during the temporary upgrade process.

To identify compatible CXFS releases, see the *CXFS MultiOS Software Compatibility Matrix* that is posted on Supportfolio:

https://support.sgi.com/content_request/139840/index.html

After the upgrade process is complete, all nodes should be running the same major-level release *X.Y* (such as 4.0), or any minor-level release with the same major level *X.Y.anything* (such as 4.0.3).

Caution: You must upgrade all server-capable administration nodes before upgrading any client-only nodes (server-capable administration nodes must run the same or later release as client-only nodes.) Operating a cluster with clients running a mixture of older and newer CXFS versions will result in a performance loss. Relocation to a server-capable administration node that is running an older CXFS version is not supported.

Although clients that are not upgraded might continue to function in the CXFS cluster without problems, new CXFS functionality may not be enabled until all clients are upgraded; SGI does not provide support for any CXFS problems encountered on the clients that are not upgraded.

Using the 3.4.2 release as an example, a production cluster could contain server-capable administration nodes running 3.4.2 and client-only nodes running 3.4, 3.4.1, and 3.4.2; it could contain client-only nodes running 3.4.3 only because there was no server platforms included in 3.4.3. It should not contain any nodes running 3.3.

Upgrading Licenses from 4.*X* to 5.0

CXFS 5.0 requires 5.0 licenses. To upgrade from a 4.*X* release to 5.0, do the following:

1. Obtain and install CXFS 5.0 licenses from SGI. These are available to all customers with current support contracts; contact your SGI support person. For more information, see Chapter 5, "CXFS License Keys" on page 89.

2. Delete all old 4.*X* server-side and all client-side licenses from all nodes in the cluster.

   Caution: Failure to delete the old license from a node may result in failed cluster membership.

3. Follow the steps in "General Upgrade Procedure" on page 279.

General Upgrade Procedure

To upgrade a CXFS cluster, do the following:

1. Ensure all server-capable administration nodes are running the same previous software release.

2. Upgrade the *standby node* (say `admin2`), which is a server-capable administration node that is configured as a potential metadata server for a given filesystem. See the release notes and Chapter 7, "Server-Capable Administration Node Installation" on page 109.

3. For the first server-capable administration node that is an active metadata server (say `admin1`), move all CXFS filesystems running on it to the standby node (making the standby node now the active metadata server for those filesystems). Do the following:

   ```
   admin1# /sbin/chkconfig grio2 off (if using GRIO)
   admin1# /sbin/chkconfig cxfs off
   admin1# /sbin/chkconfig cxfs_cluster off
   admin1# reboot
   ```

   Note: When performing upgrades, you should not make any other configuration changes to the cluster (such as adding new nodes or filesystems) until the upgrade of all nodes is complete and the cluster is running normally.

4. Upgrade the server-capable administration node (`admin1`).

5. Return the upgraded server-capable administration node (`admin1`) to the cluster. Do the following:

   ```
   admin1# /sbin/chkconfig cxfs_cluster on
   admin1# /sbin/chkconfig cxfs on
   admin1# /sbin/chkconfig grio2 on (if using GRIO)
   admin1# reboot
   ```

   Note: Skip steps 6, 7, and 8 if your cluster has only two server-capable administration nodes.

6. For the next server-capable administration node that is an active metadata server (say `admin3`), move all CXFS filesystems running on it to the standby node

(making the standby node now the active metadata server for those filesystems). Do the following to force recovery:

```
admin3# /sbin/chkconfig grio2 off (if using GRIO)
admin3# /sbin/chkconfig cxfs off
admin3# /sbin/chkconfig cxfs_cluster off
admin3# reboot
```

7. Upgrade the server-capable administration node (`admin3`).
8. Return the upgraded server-capable administration node (`admin3`) to the cluster. Do the following:

   ```
   admin3# /sbin/chkconfig cxfs_cluster on
   admin3# /sbin/chkconfig cxfs on
   admin3# /sbin/chkconfig grio2 on (if using GRIO)
   admin3# reboot
   ```

   If your cluster has additional server-capable administration nodes, repeat steps 6 through 8 for each remaining server-capable administration node.
9. Return the first CXFS filesystem to the server-capable administration node that you want to be its metadata server (making it the active metadata server, say `admin1`).
10. Return the next CXFS filesystem to the server-capable administration node that you want to be its metadata server (make it the active metadata server, say `admin3`). Repeat this step as needed for each CXFS filesystem.
11. Upgrade the client-only nodes. See the release notes for each platform and the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

Example Upgrade Process

The following figures show an example upgrade procedure for a cluster with three server-capable administration nodes and two filesystems (`/fs1` and `/fs2`), in which all nodes are running CXFS 4.0 at the beginning and Node2 is the standby node.

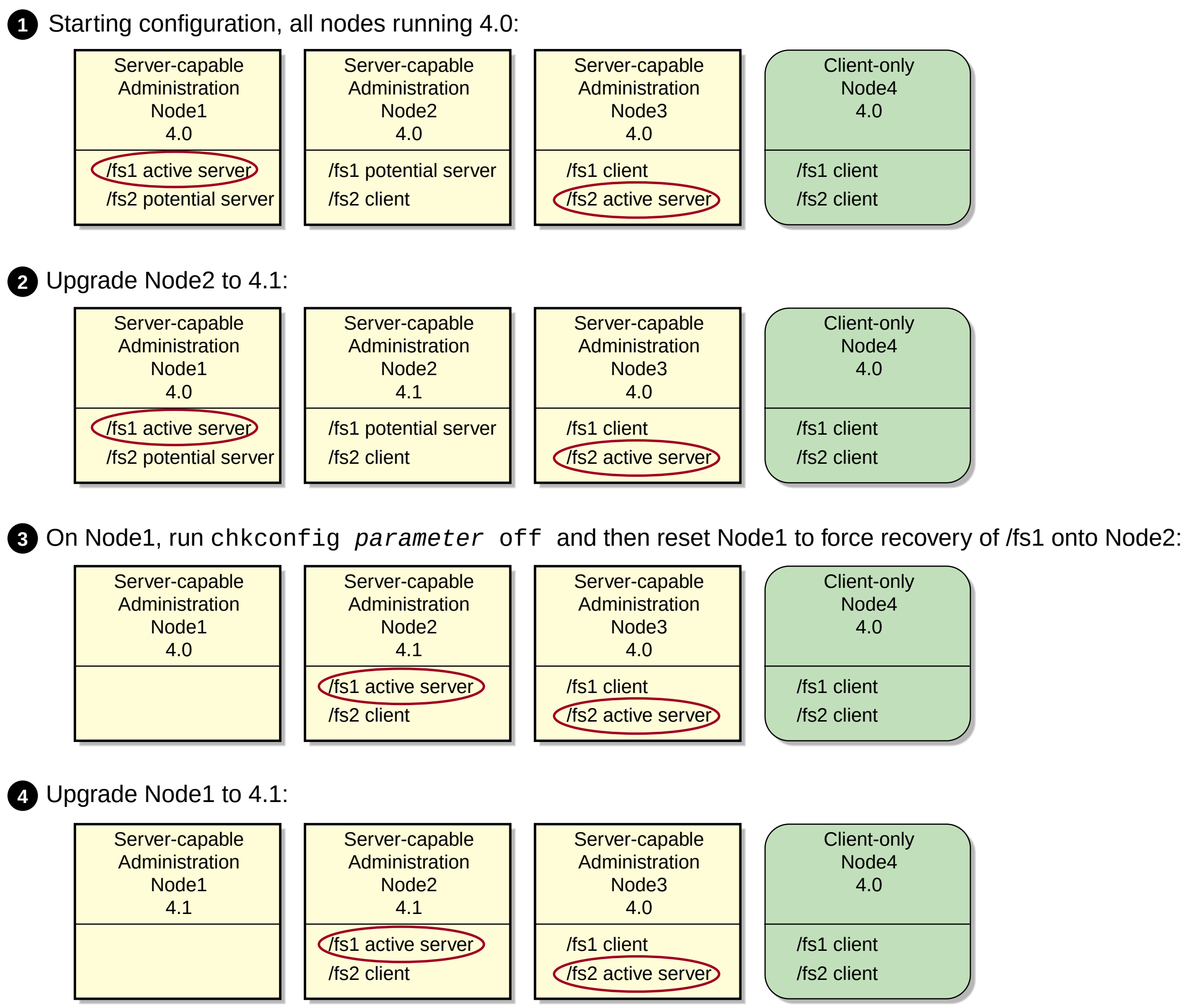
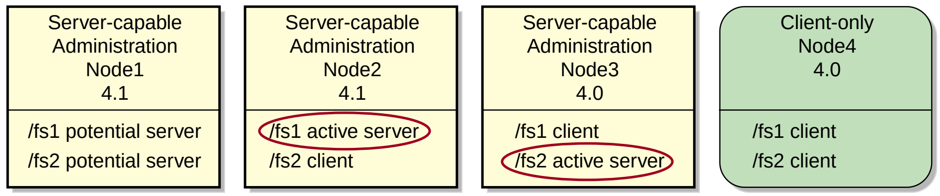

1 Starting configuration, all nodes running 4.0:
Server-capable Administration Node1 4.0
/fs1 active server
/fs2 potential server
Server-capable Administration Node2 4.0
/fs1 potential server
/fs2 client
Server-capable Administration Node3 4.0
/fs1 client
/fs2 active server
Client-only Node4 4.0
/fs1 client
/fs2 client
2 Upgrade Node2 to 4.1:
Server-capable Administration Node1 4.0
/fs1 active server
/fs2 potential server
Server-capable Administration Node2 4.1
/fs1 potential server
/fs2 client
Server-capable Administration Node3 4.0
/fs1 client
/fs2 active server
Client-only Node4 4.0
/fs1 client
/fs2 client
3 On Node1, run chkconfig parameter off and then reset Node1 to force recovery of /fs1 onto Node2:
Server-capable Administration Node1 4.0
Server-capable Administration Node2 4.1
/fs1 active server
/fs2 client
Server-capable Administration Node3 4.0
/fs1 client
/fs2 active server
Client-only Node4 4.0
/fs1 client
/fs2 client
4 Upgrade Node1 to 4.1:
Server-capable Administration Node1 4.1
Server-capable Administration Node2 4.1
/fs1 active server
/fs2 client
Server-capable Administration Node3 4.0
/fs1 client
/fs2 active server
Client-only Node4 4.0
/fs1 client
/fs2 client

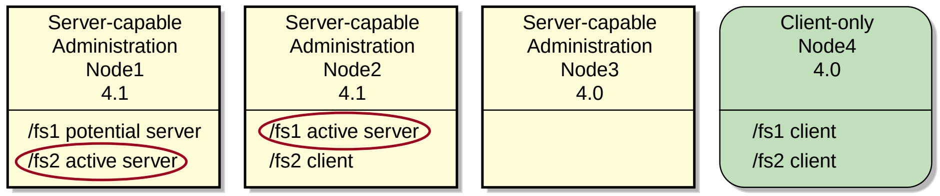
Figure 12-2 Example Rolling Upgrade Procedure (part 1)

5 On Node1, run `chkconfig` *`parameter`* `on` and then reset Node1:

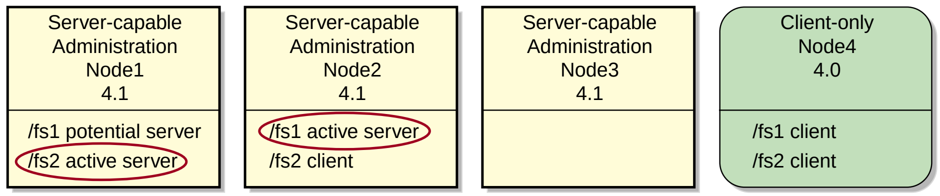
Note: Ensure that there will be no I/O that will be restarted from Node1 to /fs1 or /fs2 after Node1 is reset.



6 On Node3, run `chkconfig` *`parameter`* `off` and then reset Node3 to force recovery of /fs2 onto Node1:


Server-capable Administration Node1 4.1
/fs1 potential server
/fs2 active server
Server-capable Administration Node2 4.1
/fs1 active server
/fs2 client
Server-capable Administration Node3 4.0
Client-only Node4 4.0
/fs1 client
/fs2 client

7 Upgrade Node3 to 4.1:


Server-capable Administration Node1 4.1
/fs1 potential server
/fs2 active server
Server-capable Administration Node2 4.1
/fs1 active server
/fs2 client
Server-capable Administration Node3 4.1
Client-only Node4 4.0
/fs1 client
/fs2 client

8 On Node3, run `chkconfig` *`parameter`* `on` and then reset Node3:

Note: Ensure that there will be no I/O that will be restarted from Node3 to /fs2 after Node3 is reset.

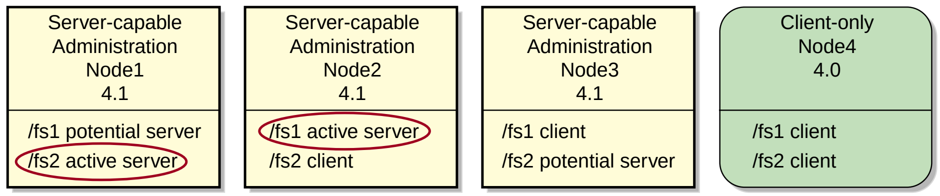
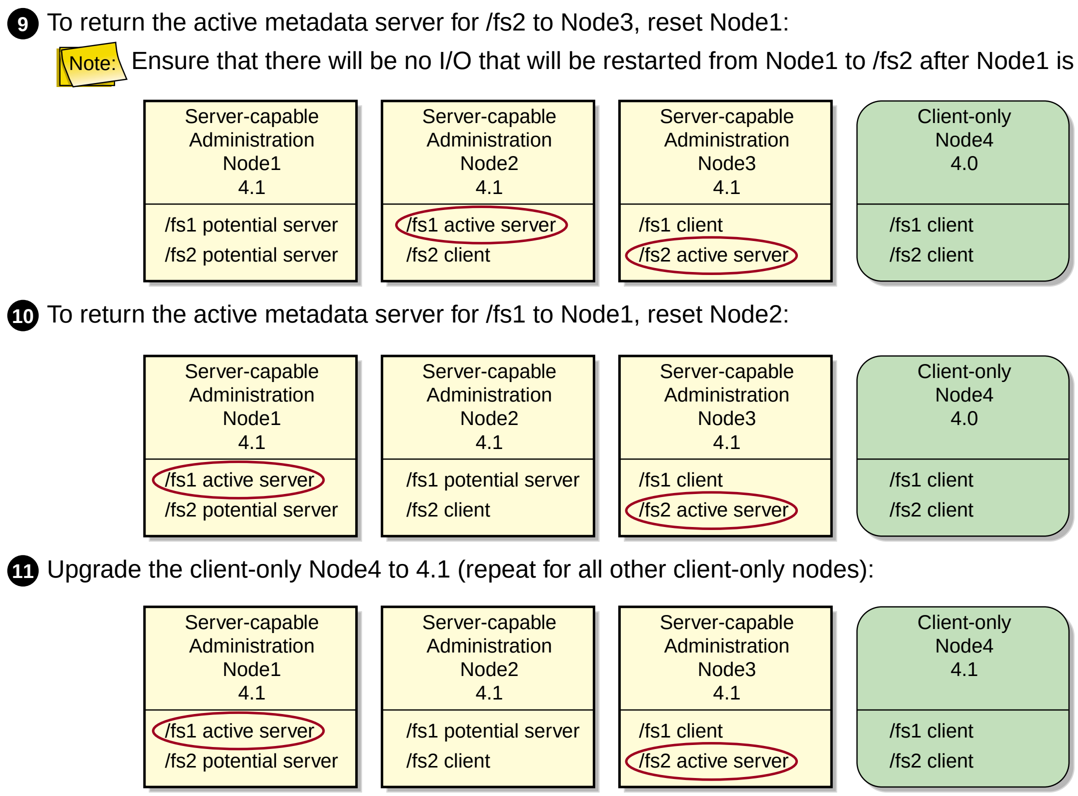


Figure 12-3 Example Rolling Upgrade Procedure (part 2)


9 To return the active metadata server for /fs2 to Node3, reset Node1:

Note: Ensure that there will be no I/O that will be restarted from Node1 to /fs2 after Node1 is reset.

10 To return the active metadata server for /fs1 to Node1, reset Node2:

11 Upgrade the client-only Node4 to 4.1 (repeat for all other client-only nodes):

Figure 12-4 Example Rolling Upgrade Procedure (part 3)

CXFS Relocation Capability

Note: Relocation is disabled by default.

To use relocation, you must enable relocation on the active metadata server. To enable relocation, reset the `cxfs_relocation_ok` parameter as follows:

- Enable at run time:

```
MDS# sysctl -w fs.cxfs.cxfs_relocation_ok=1
```

- Enable at reboot by adding the following line to `/etc/modprobe.conf.local`:

  ```
  options sgi-cxfs cxfs_relocation_ok=1
  ```

To disable relocation, do the following:

- Disable at run time:

  ```
  MDS# sysctl -w fs.cxfs.cxfs_relocation_ok=0
  ```

- Disable at reboot by adding the following line to `/etc/modprobe.conf` or `/etc/modprobe.conf.local`:

  ```
  options sgi-cxfs cxfs_relocation_ok=0
  ```

See also "Relocation" on page 25.

CXFS and Cluster Administration Initialization Commands

Table 12-1 summarizes the `/etc/init.d` initialization commands used for the CXFS control daemon and the cluster administration daemons. For more information about commands, see "CXFS Tools" on page 36.

Table 12-1 Initialization Commands

Command	Arguments	Description

`service cxfs_client`	`start`
	`stop`
	`restart`
	`status`
		Starts, stops, restarts (stops and then starts), or gives the status (`running` or `stopped`) of `cxfs_client` daemon (the client daemon) on the local node

`service cxfs_cluster`	`start`
	`stop`
	`restart`
	`status`
		Starts, stops, restarts, or gives the status of `fs2d`, `cmond`, `cad`, and `crsd` (the cluster administration daemons) on the local node

`service cxfs`	`start`
	`stop`
	`restart`
	`status`
		Starts, stops, restarts, or gives the status of `clconfd` (the CXFS server-capable administration node control daemon) on the local node

`service grio2`	`start`
	`stop`
	`restart`
	`status`
		Starts, stops, restarts, or gives the status of `ggd2` (the GRIOv2 daemon) on the local node

Switch Manipulation Using `hafence`

To add or modify a switch, run the following commands on a server-capable administration node:

```
admin# /usr/cluster/bin/hafence -a -s switchname -u username -p password -m mask [-L vendorname]
```

To raise the fence for a node:

```
admin# /usr/cluster/bin/hafence -r nodename
```

To lower the fence for a node:

```
admin# /usr/cluster/bin/hafence -l nodename
```

To query switch status:

```
admin# /usr/cluster/bin/hafence -q -s switchname
```

Usage notes:

- `-a` adds or changes a switch in the cluster database.
- `-l` lowers the fence for the specified node.
- `-L` specifies the vendor name, which loads the appropriate plug-in library for the switch. If you do not specify the vendor name, the default is `brocade`.
- `-m` specifies one of the following:
  - A list of ports in the switch that will never be fenced. The list has the following form, beginning with the # symbol, separating each port number with a comma, and enclosed within quotation marks:

    "#*port*,*port*,*port*..."

    Each *port* is a decimal integer in the range 0 through 1023. For example, the following indicates that port numbers 2, 4, 5, 6, 7, and 23 will never be fenced:

    ```
    -m "#2,4,5,6,7,23"
    ```
  - A hexadecimal string that represents ports in the switch that will never be fenced. Ports are numbered from 0. If a given bit has a binary value of 0, the port that corresponds to that bit is eligible for fencing operations; if 1, then the port that corresponds to that bit will always be excluded from any fencing operations. For an example, see Figure 10-5 on page 196.

Server-capable administration nodes automatically discover the available HBAs and, when fencing is triggered, fence off all of the Fibre Channel HBAs when the `Fence` or `FenceReset` fail policy is selected. However, masked HBAs will not be fenced. Masking allows you to prevent the fencing of devices that are attached to the SAN but are not shared with the cluster, to ensure that they remain available regardless of CXFS status. You would want to mask HBAs used for access to tape storage, or HBAs that are only ever used to access local (nonclustered) devices.

- `-p` specifies the password for the specified *username*.
- `-q` queries switch status.
- `-r` raises the fence for the specified node.
- `-s` specifies the hostname of the Fibre Channel switch; this is used to determine the IP address of the switch.
- `-u` specifies the user name to use when sending a `telnet` message to the switch.

For example, the following defines a QLogic switch named `myqlswitch` and uses no masking:

```
admin# /usr/cluster/bin/hafence -a -s myqlswitch -u admin -p *** -L qlogic
```

Note: Vendor plugin libraries should be installed in a directory that is in the platform-specific search path of the dynamic linker, typically the same location as the fencing library, `libcrf.so`. The above command line will attempt to load the `libcrf_qlogic.so` library.

The following masks port numbers 2 and 3:

```
admin# /usr/cluster/bin/hafence -a -s myqlswitch -u admin -p *** -m "#2,3" -L qlogic
```

The following lowers the fence for `client1`:

```
admin# /usr/cluster/bin/hafence -l client1
```

The following raises the fence for `client1`:

```
admin# /usr/cluster/bin/hafence -r client1
```

The following queries port status for all switches defined in the cluster database:

```
admin# /usr/cluster/bin/hafence -q
```

For more information, see the `hafence`(1M) man page. See the release notes for supported switches.

CXFS Port Usage

CXFS uses the following ports:

- Fencing requires TCP port 23 for `telnet` access.
- The RPC port mapper requires UDP port 111 and TCP port 111.
- The `fs2d` daemon is RPC-based and is dynamically assigned on a TCP port in the range of 600-1023. The instance of `fs2d` that determines the cluster database membership also uses TCP port 5449.
- The `crsd` daemon defaults to UDP port 7500 and is set in `/etc/services`:

  ```
  sgi-crsd         7500/tcp
  ```
- The CXFS kernel uses ports 5450 through 5453 (TCP for ports 5450 and 5451, UDP for ports 5052 and 5053).
- The server-capable administration node that is the quorum leader uses UDP port 5449.
- The `cad` daemon defaults to TCP port 9000 and is set in `/etc/services`:

  ```
  sgi-cad         9000/tcp
  ```

For more information, see Appendix C, "IP Filtering for the CXFS Private Network" on page 445.

`chkconfig` Arguments

Table 12-2 summarizes the CXFS `chkconfig` arguments for server-capable administration nodes. These settings can be modified by the CXFS GUI or by the administrator. These settings only control the processes, not the cluster. Stopping the processes that control the cluster will not stop the cluster (that is, will not drop the cluster membership or lose access to CXFS filesystems and cluster volumes), and starting the processes will start the cluster **only** if the CXFS services are marked as activated in the database.

On SGI ProPack nodes, `chkconfig` settings are saved by updating various symbolic links in the `/etc/rc.`*n* directories.

The following shows the settings of the arguments on server-capable administration nodes:

```
admin# chkconfig --list | grep cxfs fam
cxfs_cluster    0:off   1:off   2:on    3:on    4:on    5:on    6:off
cxfs            0:off   1:off   2:on    3:on    4:on    5:on    6:off
fam             0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

Table 12-2 `chkconfig` Arguments for Server-Capable Administration Nodes

Argument	Description
`cxfs_cluster`	Controls the cluster administration daemons (`fs2d`, `crsd`, `cad`, and `cmond`). If this argument is `off`, the database daemons will not be started at the next reboot and the local copy of the database will not be updated if you make changes to the cluster configuration on the other nodes. This could cause problems later, especially if a majority of nodes are not running the database daemons. If the database daemons are not running, the cluster database will not be accessible locally and the node will not be configured to join the cluster.
`cxfs`	Controls the `clconfd` daemon and whether or not the `cxfs_shutdown` command is used during a system shutdown. The `cxfs_shutdown` command attempts to withdraw from the cluster gracefully before rebooting. Otherwise, the reboot is seen as a failure and the other nodes must recover from it. **Note:** `clconfd` cannot start unless `fs2d` is already running.
`fam`	Starts the file alteration monitoring (`fam`) service, which is required to use the CXFS GUI on SGI ProPack nodes.

Granting Task Execution Privileges for Users

The CXFS GUI lets you grant or revoke access to a specific GUI task for one or more specific users. By default, only `root` may execute tasks in the GUI. Access to the task is only allowed on the node to which the GUI is connected; if you want to allow access on another node in the pool, you must connect the GUI to that node and grant access again.

Note: You cannot grant or revoke tasks for users with a user ID of 0.

GUI tasks operate by executing underlying privileged commands that are normally accessible only to `root`. When granting access to a task, you are in effect granting access to all of its required underlying commands, which results in also granting access to the other GUI tasks that use the same underlying commands. The `cxfs_admin` command provides similar functionality with the `allow|deny` subcommands.

For instructions about granting or revoking GUI privileges, see "Privileges Tasks with the GUI" on page 211.

To see which tasks a specific user can currently access, select **View: Users**. Select a specific user to see details about the tasks available to that user.

To see which users can currently access a specific task, select **View: Task Privileges**. Select a specific task to see details about the users who can access it and the privileged commands it requires.

Transforming a Server-Capable Administration Node into a Client-Only Node

You should install a node as a server-capable administration node only if you intend to use it as a potential metadata server. All other nodes should be installed as client-only nodes. See "Make Most Nodes Client-Only" on page 49.

To transform a server-capable administration node into a client-only node, do the following:

1. Ensure that the node is not listed as a potential metadata server for any filesystem.
2. Stop the CXFS services on the node.

3. Modify the cluster so that it no longer contains the node.
4. Delete the node definition.
5. Remove the packages listed in "CXFS Software Products Installed on Server-Capable Administration Nodes" on page 35 from the node.
6. Reboot the node to ensure that all previous node configuration information is removed.
7. Install client-only software as documented in the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.
8. Define the node as a client-only node.
9. Modify the cluster so that it contains the node if you are using the GUI. (This step is handled by `cxfs_admin` automatically.)
10. Start CXFS services on the node.

For more information about these tasks, see:

- Chapter 10, "Reference to GUI Tasks" on page 147
- Chapter 11, "Reference to `cxfs_admin` Tasks" on page 217

CXFS Mount Scripts

On server-capable administration nodes, the following scripts are provided for execution by the `clconfd` daemon prior to and after a CXFS filesystem is mounted or unmounted:

```
/var/cluster/clconfd-scripts/cxfs-pre-mount
/var/cluster/clconfd-scripts/cxfs-post-mount
/var/cluster/clconfd-scripts/cxfs-pre-umount
/var/cluster/clconfd-scripts/cxfs-post-umount
```

The following script is run when needed to reprobe the Fibre Channel controllers on server-capable administration nodes:

```
/var/cluster/clconfd-scripts/cxfs-reprobe
```

You can customize these scripts to suit a particular environment. For example, an application could be started when a CXFS filesystem is mounted by extending the `cxfs-post-mount` script. The application could be terminated by changing the `cxfs-pre-umount` script.

On server-capable administration nodes, these scripts also allow you to use NFS to export the CXFS filesystems listed in `/etc/exports` if they are successfully mounted.

The appropriate daemon executes these scripts before and after mounting or unmounting CXFS filesystems specified in the `/etc/exports` file. The files must be named **exactly** as above and must have `root` execute permission.

Note: The `/etc/exports` file describes the filesystems that are being exported to NFS clients. If a CXFS mount point is included in the `exports` file, the empty mount point is exported unless the filesystem is re-exported after the CXFS mount using the `cxfs-post-mount` script.

The following arguments are passed to the files:

- `cxfs-pre-mount`: filesystem device name and CXFS mounting point
- `cxfs-post-mount`: filesystem device name, CXFS mounting point, and exit code
- `cxfs-pre-umount`: filesystem device name and CXFS mounting point
- `cxfs-post-umount`: filesystem device name, CXFS mounting point, and exit code

Because the filesystem device name is passed to the scripts, you can write the scripts so that they take different actions for different filesystems; because the exit codes are passed to the `-post-` files, you can write the scripts to take different actions based on success or failure of the operation.

The `clconfd` or `cxfs_client` daemon checks the exit code for these scripts. In the case of failure (nonzero), the following occurs:

- For `cxfs-pre-mount` and `cxfs-pre-umount`, the corresponding mount or unmount is not performed
- For `cxfs-post-mount` and `cxfs-post-umount`, `clconfd` will retry the entire operation (including the `-pre-` script) for that operation

This implies that if you **do not** want a filesystem to be mounted on a host, the `cxfs-pre-mount` script should return a failure for that filesystem while the `cxfs-post-mount` script returns success.

Note: After the filesystem is unmounted, the mount point is removed.

For information about the mount scripts on client-only nodes, see the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

Using Hierarchical Storage Management (HSM) Products

The HSM application must make all of its DMAPI interface calls through the active metadata server. The CXFS client nodes do not provide a DMAPI interface to CXFS mounted filesystems. A CXFS client routes all of its communication to the HSM application through the metadata server. This generally requires that the HSM application run on the CXFS metadata server.

To use HSM with CXFS, do the following:

- Install the `sgi-dmapi` and `sgi-xfsprogs` packages from the SUSE Linux Enterprise Server (SLES) installation source. For client-only nodes, no additional software is required.
- Use the `dmi` option when mounting a filesystem to be managed.
- Start the HSM application on the active metadata server for each filesystem to be managed.

Discovering the Active Metadata Server

This section discusses how to discover the active metadata server using various tools:

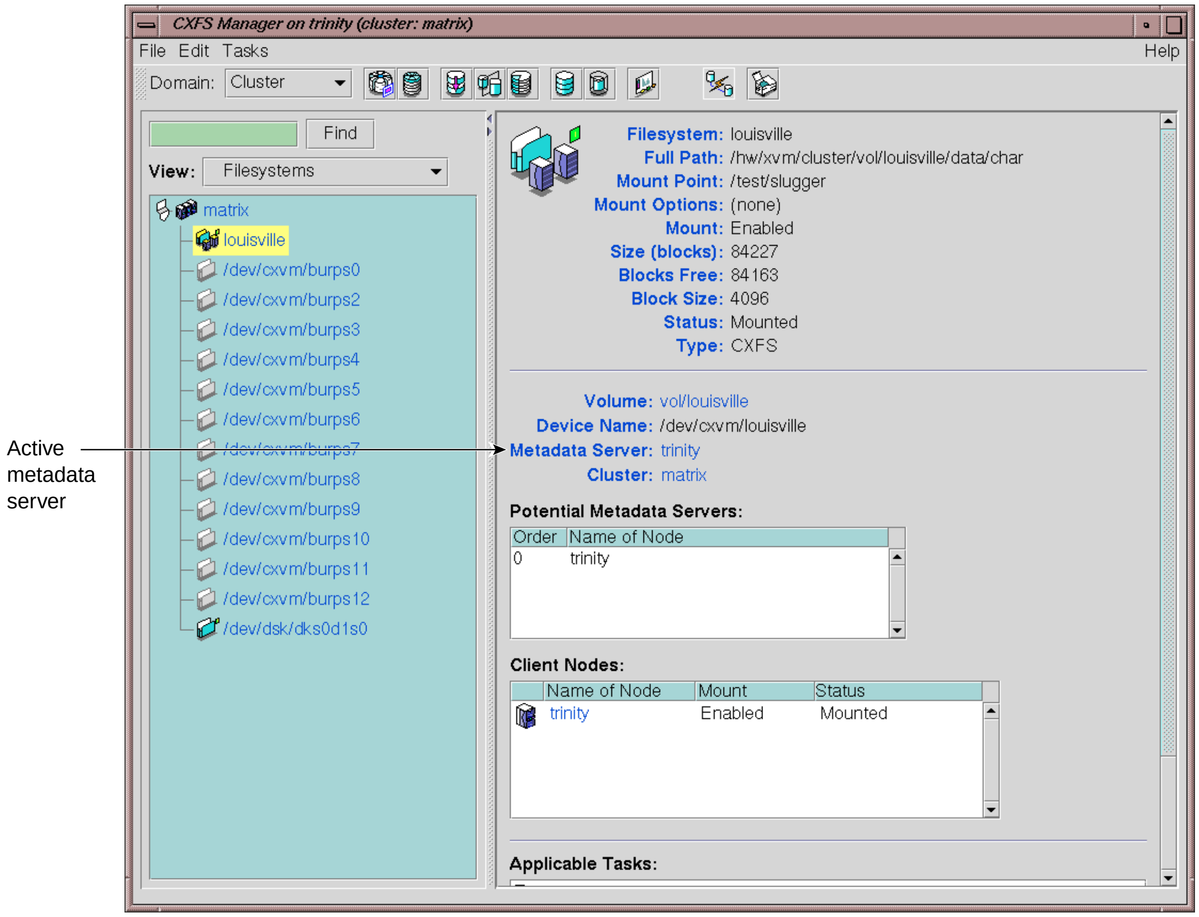
- "Discovering the Active Metadata Server with the CXFS GUI" on page 293
- "Discovering the Active Metadata Server with `cxfs_admin`" on page 295
- "Discovering the Active Metadata Server with `clconf_info`" on page 296

Discovering the Active Metadata Server with the CXFS GUI

To discover the active metadata server for a filesystem by using the GUI, do the following:

1. Select **View: Filesystems**
2. In the view area, click the name of the filesystem you wish to view. The name of the active metadata server is displayed in the details area to the right.

Figure 12-5 shows an example.



matrix
- louisville
- /dev/cxvm/burps0
- /dev/cxvm/burps2
- /dev/cxvm/burps3
- /dev/cxvm/burps4
- /dev/cxvm/burps5
- /dev/cxvm/burps6
- /dev/cxvm/burps7
- /dev/cxvm/burps8
- /dev/cxvm/burps9
- /dev/cxvm/burps10
- /dev/cxvm/burps11
- /dev/cxvm/burps12
- /dev/dsk/dks0d1s0

Filesystem: louisville
Full Path: /hw/xvm/cluster/vol/louisville/data/char
Mount Point: /test/slugger
Mount Options: (none)
Mount: Enabled
Size (blocks): 84227
Blocks Free: 84163
Block Size: 4096
Status: Mounted
Type: CXFS

Volume: vol/louisville
Device Name: /dev/cxvm/louisville
Metadata Server: trinity
Cluster: matrix

Potential Metadata Servers:

Order	Name of Node
0	trinity

Client Nodes:

Name of Node	Mount	Status
trinity	Enabled	Mounted

Applicable Tasks:

Figure 12-5 GUI Window Showing the Metadata Server

Discovering the Active Metadata Server with `cxfs_admin`

To discover the active metadata server for a filesystem by using the `cxfs_admin`, do the following:

- To show information for all filesystems, including their active metadata servers:

  ```
  show server
  ```

  For example:

  ```
  cxfs_admin:mycluster> show server
  filesystem:concatfs:status:server=mds1
  filesystem:mirrorfs:status:server=mds1
  filesystem:stripefs:status:server=mds2
  ```

- To show the active metadata server for a specific filesystem:

  `show [filesystem:]`*filesystem*`:status:server`

  In the above, you could abbreviate `status` to `*`. For example, if `concatfs` is a unique name in the cluster database:

  ```
  cxfs_admin:mycluster> show concatfs:*:server
  filesystem:concatfs:status:server=mds1
  ```

Discovering the Active Metadata Server with `clconf_info`

You can use the `clconf_info` command to discover the active metadata server for a given filesystem. For example, the following shows that `cxfs7` is the metadata server:

```
cxfs6 # clconf_info

Event at [2004-04-16 09:20:59]

Membership since Fri Apr 16 09:20:56 2004

____________ ______ ________ ______ ______
Node         NodeID Status   Age    CellID
____________ ______ ________ ______ ______
cxfs6             6 up       0           2
cxfs7             7 up       0           1
cxfs8             8 up       0           0
____________ ______ ________ ______ ______
1 CXFS FileSystems
/dev/cxvm/concat0 on /concat0  enabled  server=(cxfs7)  2 client(s)=(cxfs8,cxfs6)
```

Shutdown of the Database and CXFS

This section tells you how to perform the following:

- "Cluster Database Shutdown" on page 297
- "Normal CXFS Shutdown: Stop CXFS Services or Disable the Node" on page 298
- "Forced CXFS Shutdown: Revoke Membership of Local Node" on page 299

For more information about states, Chapter 14, "Monitoring Status" on page 341. If there are problems, see Chapter 15, "Troubleshooting" on page 353.

Cluster Database Shutdown

A *cluster database shutdown* terminates the following user-space daemons that manage the cluster database:

```
cad
clconfd
cmond
crsd
fs2d
```

After shutting down the database on a node, access to the shared filesystems remains available and the node is still a member of the cluster, but the node is not available for database updates. Rebooting of the node results in a restart of all services (restarting the daemons, joining cluster membership, enabling cluster volumes, and mounting CXFS filesystems).

To perform a cluster database shutdown, enter the following:

```
admin# killall -TERM clconfd
admin# service cxfs_cluster stop
```

If you also want to disable the daemons from restarting at boot time, enter the following:

```
admin# /sbin/chkconfig cxfs off
admin# /sbin/chkconfig cxfs_cluster off
```

For more information, see "`chkconfig` Arguments" on page 288.

Node Status and Cluster Database Shutdown

A cluster database shutdown is appropriate when you want to perform a maintenance operation on the node and then reboot it, returning it to `ACTIVE` status (as displayed by the GUI) or `stable` status (as displayed by `cxfs_admin`).

If you perform a cluster database shutdown, the node status will be `DOWN` in the GUI or `inactive` in `cxfs_admin`, which has the following impacts:

- The node is still considered part of the cluster, but unavailable.
- The node does not get cluster database updates; however, it will be notified of all updates after it is rebooted.

Missing cluster database updates can cause problems if the kernel portion of CXFS is active. That is, if the node continues to have access to CXFS, the node's kernel level will not see the updates and will not respond to attempts by the remaining nodes to propagate these updates at the kernel level. This in turn will prevent the cluster from acting upon the configuration updates.

Note: If the cluster database is shut down on more than half of the server-capable administration nodes, changes cannot be made to the cluster database.

Restart the Cluster Database

To restart the cluster database, enter the following:

```
admin# service cxfs_cluster start
admin# service cxfs start
```

Normal CXFS Shutdown: Stop CXFS Services or Disable the Node

You should perform a *normal CXFS shutdown* in the GUI or disable a node in `cxfs_admin` when you want to stop CXFS services on a node and remove it from the CXFS kernel membership quorum.

A normal CXFS shutdown in the GUI does the following:

- Unmounts all the filesystems except those for which it is the active metadata server; those filesystems for which the node is the active metadata server will become inaccessible from the node after it is shut down.
- Terminates the CXFS kernel membership of this node in the cluster.
- Marks the node as `INACTIVE` in the GUI and `disabled` in `cxfs_admin`.

The effect of this is that cluster disks are unavailable and no cluster database updates will be propagated to this node. Rebooting the node leaves it in the shutdown state.

If the node on which you shut down CXFS services is an active metadata server for a filesystem, then that filesystem will be recovered by another node that is listed as one of its potential metadata servers. The server that is chosen must be a filesystem client; other filesystem clients will experience a delay during the recovery process.

If the node on which the CXFS shutdown is performed is the sole potential metadata server (that is, there are no other nodes listed as potential metadata servers for the filesystem), then you should unmount the filesystem from all nodes before performing the shutdown.

The GUI task can operate on all nodes in the cluster or on the specified node; the `cxfs_admin disable` command operates on just a single specified node.

To perform a normal CXFS shutdown, see

- "Stop CXFS Services with the GUI" on page 189
- "Disable a Node with `cxfs_admin`" on page 244

When You Should Not Perform Stop CXFS Services

You should not stop CXFS services under the following circumstances:

- If CXFS services are running on the *local node* (the server-capable administration node on which `cxfs_admin` is running or the node to which the CXFS GUI is connected)
- If stopping CXFS services on the node will result in loss of CXFS kernel membership quorum
- If the node is the only available potential metadata server for one or more active CXFS filesystems

To achieve a CXFS shutdown under these conditions, you must perform a forced CXFS shutdown. See "Forced CXFS Shutdown: Revoke Membership of Local Node" on page 299.

Rejoining the Cluster after Stopping CXFS Services

The node will not rejoin the cluster after a reboot. The node will rejoin the cluster only after CXFS services are explicitly reactivated with the CXFS GUI or after the noded is enabled using `cxfs_admin`.

Forced CXFS Shutdown: Revoke Membership of Local Node

A *forced CXFS shutdown* (or *administrative CXFS stop*) is appropriate when you want to shutdown the local node even though it may drop the cluster below its CXFS kernel membership quorum requirement.

CXFS does the following:

- Shuts down all CXFS filesystems on the local node. Any attempts to access the CXFS filesystems will result in an I/O error (you may need to manually unmount the filesystems).
- Removes this node from the CXFS kernel membership.
- Marks the node as `DOWN` in the GUI or `inactive` in `cxfs_admin`.
- Disables access from the local node to cluster-owned XVM volumes.
- Treats the stopped node as a failed node and executes the fail policy defined for the node in the cluster database. See "Fail Policies" on page 57.

Caution: A forced CXFS shutdown may cause the cluster to fail if the cluster drops below CXFS kernel membership quorum.

If you do a forced CXFS shutdown on an active metadata server, it loses membership immediately. At this point, another potential metadata server must take over (and recover the filesystems) or quorum is lost and a forced CXFS shutdown follows on all nodes.

If you do a forced CXFS shutdown that forces a loss of quorum, the remaining part of the cluster (which now must also do an administrative stop) will **not** reset the departing node.

To perform an administrative stop, see

- "Revoke Membership of the Local Node with the GUI" on page 192
- "Disable a Node with `cxfs_admin`" on page 244

Node Status and Forced CXFS Shutdown

After a forced CXFS shutdown, the node is still considered part of the configured cluster and is taken into account when propagating the cluster database and when computing the cluster database (`fs2d`) membership quorum. (This could cause a loss of quorum for the rest of the cluster, causing the other nodes to do a forced CXFS shutdown). The state is `INACTIVE` in the GUI or `inactive` in `cxfs_admin`.

It is important that this node stays accessible and keeps running the cluster infrastructure daemons to ensure database consistency. In particular, if more than half the nodes in the pool are down or not running the infrastructure daemons, cluster database updates will stop being propagated and will result in inconsistencies. To be safe, you should remove those nodes that will remain unavailable from the cluster and pool.

Rejoining the Cluster after a Forced CXFS Shutdown

After a forced CXFS shutdown, the local node will not resume CXFS kernel membership until the node is rebooted or until you explicitly allow CXFS kernel membership for the local node.

See:

- "Allow Membership of the Local Node with the GUI" on page 193
- "Disable a Node with `cxfs_admin`" on page 244
- "Enable a Node with `cxfs_admin`" on page 244

If you perform a forced CXFS shutdown on a server-capable administration node, you must restart CXFS on that node before it can return to the cluster. If you do this while the cluster database still shows that the node is in the cluster and is activated, the node will restart the CXFS kernel membership daemon. Therefore, you may want to do this after resetting the database or after stopping CXFS services.

Reset Capability and a Forced CXFS Shutdown

Caution: If you perform an administrative CXFS stop on a server-capable administration node with system reset capability and the stop will not cause loss of cluster quorum, the node will be reset (rebooted) by the appropriate node.

For more information about resets, see "System Reset" on page 50.

Avoiding a CXFS Restart at Reboot

If the following `chkconfig` arguments are turned off, the `clconfd` and `cxfs_client` daemons on server-capable administration nodes and client-only nodes, respectively, will not be started at the next reboot and the kernel will not be configured to join the cluster:

- SGI ProPack administration nodes: `cxfs`
- Client-only nodes: `cxfs_client`

It is useful to turn these arguments off before rebooting if you want to temporarily remove the nodes from the cluster for system or hardware upgrades or for other maintenance work.

For example, do the following:

```
admin# /sbin/chkconfig grio2 off (If running GRIOv2)
admin# /sbin/chkconfig cxfs off
admin# /sbin/chkconfig cxfs_cluster off
admin# reboot
```

For more information, see "`chkconfig` Arguments" on page 288.

Log File Management

CXFS log files should be rotated at least weekly so that your disk will not become full.

A package that provides CXFS daemons also supplies scripts to rotate the log files for those daemons via `logrotate`. SGI installs the following scripts on server-capable administration nodes:

```
/etc/logrotate.d/cluster_admin
/etc/logrotate.d/cluster_control
/etc/logrotate.d/cxfs_cluster
/etc/logrotate.d/grio2
```

SGI installs the following script on client-only nodes:

```
/etc/logrotate.d/cxfs_client
```

To customize log rotation, edit these scripts..

For information about log levels, see "Configure Log Groups with the GUI" on page 192.

Filesystem Maintenance

Although filesystem information is traditionally stored in `/etc/fstab`, the CXFS filesystems information is relevant to the entire cluster and is therefore stored in the replicated cluster database instead.

As the administrator, you will supply the CXFS filesystem configuration by using the CXFS GUI or `cxfs_admin`. The information is then automatically propagated consistently throughout the entire cluster. The cluster configuration daemon mounts the filesystems on each node according to this information, as soon as it becomes available.

A CXFS filesystem will be automatically mounted on all the nodes in the cluster. You can add a new CXFS filesystem to the configuration when the cluster is active.

Whenever the cluster configuration daemon detects a change in the cluster configuration, it does the equivalent of a `mount -a` command on all of the filesystems that are configured.

Caution: You must not modify or remove a CXFS filesystem definition while the filesystem is mounted. You must unmount it first and then mount it again after making the modifications.

Mounting Filesystems

You supply mounting information with the CXFS GUI or `cxfs_admin`.

Caution: Do not attempt to use the `mount` command to mount a CXFS filesystem. Doing so can result in data loss and/or corruption due to inconsistent use of the filesystem from different nodes.

When properly defined and mounted, the CXFS filesystems are automatically mounted on each node by the local cluster configuration daemon, `clconfd`, according to the information collected in the replicated database. After the filesystems configuration has been entered in the database, no user intervention is necessary.

Mount points cannot be nested when using CXFS. That is, you cannot have a filesystem within a filesystem, such as `/usr` and `/usr/home`.

Unmounting Filesystems

To unmount CXFS filesystems, use the CXFS GUI or `cxfs_admin`. These tools unmount a filesystem from all nodes in the cluster. Although this action triggers an unmount on all the nodes, some might fail if the filesystem is busy. On active metadata servers, the unmount cannot succeed before all of the CXFS clients have successfully unmounted the filesystem. All nodes will retry the unmount until it succeeds, but there is no centralized report that the filesystem has been unmounted on all nodes.

To verify that the filesystem has been unmounted from all nodes, do one of the following:

- Check the `SYSLOG` files on the metadata servers for a message indicating that the filesystem has been unmounted.
- Run the CXFS GUI or `cxfs_admin` on the metadata server, disable the filesystem from the server, and wait until the GUI shows that the filesystem has been fully disabled. (It will be an error if it is still mounted on some CXFS clients; the GUI will show which clients are left.)

Growing Filesystems

To grow a CXFS filesystem, do the following:

1. Unmount the CXFS filesystem using the CXFS GUI or `cxfs_admin`.
2. Change the domain of the XVM volume from a cluster volume to a local volume using the XVM `give` command. See the *XVM Volume Manager Administrator's Guide*.
3. Mount the filesystem as an XFS filesystem using the `mount` command.
4. Use the `xfs_growfs` command or the CXFS GUI task; see "Grow a Filesystem with the GUI" on page 201.
5. Unmount the XFS filesystem.
6. Change the domain of the XVM volume back to a cluster volume using the `give` command. See the *XVM Volume Manager Administrator's Guide*.

7. Mount the filesystem as a CXFS filesystem by using the GUI or `cxfs_admin`

Dump and Restore

You must perform the backup of a CXFS filesystem from the active metadata server for that filesystem. The `xfsdump` and `xfsrestore` commands make use of special system calls that will only function on the active metadata server. The filesystem can have active clients during a dump process.

In a clustered environment, a CXFS filesystem may be directly accessed simultaneously by many CXFS clients and the active metadata server. A filesystem may, over time, have a number of metadata servers. Therefore, in order for `xfsdump` to maintain a consistent inventory, it must access the inventory for past dumps, even if this information is located on another node. SGI recommends that the inventory be made accessible by potential metadata server nodes in the cluster using one of the following methods:

- Relocate the inventory to a shared filesystem. For example, where *shared_filesystem* is replaced with the actual name of the filesystem to be shared:
  - On the node currently containing the inventory, enter the following:

    ```
    inventoryadmin# cd /var/lib
    inventoryadmin# cp -r xfsdump /shared_filesystem
    inventoryadmin# mv xfsdump xfsdump.bak
    inventoryadmin# ln -s /shared_filesystem/xfsdump xfsdump
    ```

  - On all other server-capable administration nodes in the cluster, enter the following:

    ```
    otheradmin# cd /var/lib
    otheradmin# mv xfsdump xfsdump.bak
    otheradmin# ln -s /shared_filesystem/xfsdump xfsdump
    ```

- Export the directory using an NFS shared filesystem. For example:
  - On the server-capable administration node currently containing the inventory, add `/var/lib/xfsdump` to `/etc/exports` and then enter the following:

    ```
    inventoryadmin# exportfs -a
    ```

– On all other server-capable administration nodes in the cluster, enter the following:

```
otheradmin# cd /var/lib
otheradmin# mv  xfsdump  xfsdump.bak
otheradmin# ln -s /net/hostname/var/lib/xfsdump  xfsdump
```

Note: It is the `/var/lib/xfsdump` directory that should be shared, rather than the `/var/lib/xfsdump/inventory` directory. If there are inventories stored on various nodes, you can use `xfsinvutil` to merge them into a single common inventory, prior to sharing the inventory among the nodes in the cluster.

Hardware Changes and I/O Fencing

If you use I/O fencing and then make changes to your hardware configuration, you must verify that switch ports are properly enabled so that they can discover the WWPN of the HBA for I/O fencing purposes.

You must check the status of the switch ports involved whenever any of the following occur:

- An HBA is replaced on a node
- A new node is plugged into the switch for the first time
- A Fibre Channel cable rearrangement occurs

Note: The affected nodes should be shutdown before rearranging cables.

To check the status, use the following command on a server-capable administration node:

```
admin# hafence -v
```

If any of the affected ports are found to be disabled, you must manually enable them before starting CXFS on the affected nodes:

1. Connect to the switch using `telnet`.
2. Use the `portenable` command to enable the port.

3. Close the `telnet` session.

After the port is enabled, the metadata server will be able to discover the new (or changed) WWPN of the HBA connected to that port and thus correctly update the switch configuration entries in the cluster database.

Private Network Failover

This section provides an example of modifying a cluster to provide private network failover by using the `cxfs_admin` command.

Do the following:

1. Create the failover network subnets. For example:

```
cxfs_admin:mycluster> create failover_net network=192.168.0.0 mask=255.255.255.0
cxfs_admin:mycluster> create failover_net network=192.168.1.0 mask=255.255.255.0
```

2. Disable all nodes (which shuts down the cluster):

```
cxfs_admin:mycluster> disable *
```

3. Update each node to include a private network. For example:

```
cxfs_admin:mycluster> modify red private_net=192.168.0.1,192.168.1.1
cxfs_admin:mycluster> modify yellow private_net=192.168.0.2,192.168.1.2
```

4. Enable all nodes:

```
cxfs_admin:mycluster> enable *
```

For more information, see Chapter 11, "Reference to `cxfs_admin` Tasks" on page 217.

Cluster Member Removal and Restoration

This section discusses removing and restoring cluster members for maintenance:

- "Manually Starting/Stopping CXFS" on page 308
- "Removing a Metadata Server from the Cluster" on page 309
- "Restoring a Metadata Server to the Cluster" on page 310

- "Removing a Single Client-Only Node from the Cluster" on page 310
- "Restoring a Single Client-Only Node to the Cluster" on page 312
- "Stopping CXFS for the Entire Cluster" on page 313
- "Restarting the Entire Cluster" on page 314

These procedures are the safest way to perform these tasks but in some cases are not the most efficient. They should be followed if you have been having problems using standard operating procedures (performing a stop/start of CXFS services or a simple host shutdown or reboot).

Manually Starting/Stopping CXFS

Note: If you are going to perform maintenance on a potential metadata server, you should first shut down CXFS services on it. Disabled nodes are not used in CXFS kernel membership calculations, so this action may prevent a loss of quorum.

On server-capable administration nodes, the `service cxfs_cluster` script will be invoked automatically during normal system startup and shutdown procedures. (On client-only nodes, the path to the `cxfs_client` script varies by platform; see *CXFS Administration Guide for SGI InfiniteStorage*.) This script starts and stops the processes required to run CXFS.

To start up CXFS processes manually on a server-capable administration node, enter the following commands:

```
admin# service grio2 start  (if running GRIOv2)
admin# service cxfs_cluster start
admin# service cxfs start
```

To stop CXFS processes manually on a server-capable administration node, enter the following commands:

```
admin# service grio2 stop  (stops GRIOv2 daemons)
admin# service cxfs stop (stops the CXFS server-capable administration node control daemon)
admin# service cxfs_cluster stop (stops the cluster administration daemons)
```

Note: There is also a `restart` option that performs a stop and then a start.

Removing a Metadata Server from the Cluster

If you have a cluster with multiple active metadata servers and you must perform maintenance on one of them, you must stop CXFS services on it.

To remove an active metadata server (`admin1` for example) from the cluster, do the following:

1. Enable relocation by using the `cxfs_relocation_ok` system tunable parameter. See "Relocation" on page 25.
2. For each filesystem for which `admin1` is the active metadata server, manually relocate the metadata services from `admin1` to one of the other potential metadata servers by using the CXFS GUI or `cxfs_admin`. For example:

   ```
   cxfs_admin:mycluster> relocate fs1 server=node2
   ```

3. Disable relocation. See "Relocation" on page 25.

   Note: If you do not perform steps 1–3 in a system reset configuration, `admin1` will be reset shortly after losing its membership. The machine will also be configured to reboot automatically instead of stopping in the PROM. This means that you must watch the console and intervene manually to prevent a full reboot.

   In a fencing configuration, `admin1` will lose access to the SAN when it is removed from the cluster membership.

4. Stop CXFS services for `admin1` by using the CXFS GUI or `cxfs_admin` running on another metadata server. For example:

   ```
   cxfs_admin:mycluster> disable admin1
   ```

5. Shut down `admin1`.

If you do not want the cluster administration daemons and the CXFS control daemon to run during maintenance, execute the following commands:

```
admin1# /sbin/chkconfig grio2 off (if running GRIOv2)
admin1# /sbin/chkconfig cxfs off
admin1# /sbin/chkconfig cxfs_cluster off
```

If you do an upgrade of the cluster software, these arguments will be automatically reset to `on` and the cluster administration daemons and the CXFS control daemon will be started.

For more information, see "`chkconfig` Arguments" on page 288.

Restoring a Metadata Server to the Cluster

To restore a metadata server to the cluster, do the following:

1. Allow the cluster administration daemons, CXFS control daemon, and GRIOv2 daemon (if using GRIOv2) to be started upon reboot:

   ```
   admin1# /sbin/chkconfig cxfs on
   admin1# /sbin/chkconfig cxfs_cluster on
   admin1# /sbin/chkconfig grio2 on (if using GRIOv2)
   ```

2. Immediately start cluster administration daemons on the node:

   ```
   exMD# service cxfs_cluster start
   ```

3. Immediately start the CXFS control daemon on the node:

   ```
   admin1# service cxfs start
   ```

4. Immediately start the GRIOv2 daemon on the node (if using GRIOv2):

   ```
   admin1# service grio2 start
   ```

5. Start CXFS services on this node from another server-capable administration node:

```
otheradmin# cmgr -c "start cx_services on node admin1 for cluster clustername force"
```

Removing a Single Client-Only Node from the Cluster

To remove a single client-only node from the cluster, do the following:

1. Verify that the configuration is consistent among active metadata servers in the cluster by running the following on each active metadata server and comparing the output:

   ```
   MDS# /usr/cluster/bin/clconf_info
   ```

   If the client is not consistent with the metadata servers, or if the metadata servers are not consistent, then you should abort this procedure and address the health of the cluster. If a client is removed while the cluster is unstable, attempts to get the client to rejoin the cluster are likely to fail. For this reason, you should make sure that the cluster is stable before removing a client.

2. Flush the system buffers on the client you want to remove in order to minimize the amount of buffered information that may be lost:

   ```
   client# sync
   ```

3. Stop CXFS services on the client. For example:

   ```
   client# service cxfs_client stop
   client# chkconfig cxfs_client off
   ```

4. Verify that CXFS services have stopped:

   - Verify that the CXFS client daemon is not running on the client (success means no output):

     ```
     client# ps -ef | grep cxfs_client
     client#
     ```

   - Monitor the `cxfs_client` log on the client you wish to remove and look for filesystems that are unmounting successfully. For example:

```
Apr 18 13:00:06 cxfs_client: cis_setup_fses Unmounted green0: green0 from /cxfs/green0
```

   - Monitor the `SYSLOG` on the active metadata server and look for membership delivery messages that do not contain the removed client. For example, the following message indicates that cell 2 (`client`), the node being shut down, is not included in the membership:

```
Apr 18 13:01:03 5A:o200a unix: NOTICE: Cell 2 (client) left the membership
Apr 18 13:01:03 5A:o200a unix: NOTICE: Membership delivered for cells 0x3
Apr 18 13:01:03 5A:o200a unix: Cell(age): 0(7) 1(5)
```

   - Use the following command to show that filesystems are not mounted:

     ```
     client# df -hl
     ```

5. Verify that the configuration is consistent and does not contain the removed client by running the following on each active metadata server and comparing the output:

   ```
   admin# /usr/cluster/bin/clconf_info
   ```

Restoring a Single Client-Only Node to the Cluster

To restore a single client-only node to the cluster, do the following:

1. Verify that the configuration is consistent among active metadata servers in the cluster by running the following on each active metadata server and comparing the output:

   ```
   MDS# /usr/cluster/bin/clconf_info
   ```

   If the client is not consistent with the metadata servers, or if the metadata servers are not consistent, then you should abort this procedure and address the health of the cluster. If a client is removed while the cluster is unstable, attempts to get the client to rejoin the cluster are likely to fail. For this reason, you should make sure that the cluster is stable before removing a client.

2. Start CXFS on the client-only node:

   ```
   client# chkconfig cxfs_client on
   client# service cxfs_client start
   ```

   Note: The path to `cxfs_client` varies across the operating systems supported by CXFS. For more information, see *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

3. Verify that CXFS has started:

   - Verify that the CXFS client daemon is running on the client-only node:

```
client# ps -ef | grep cxfs_client
    root        716          1  0 12:59:14 ?       0:05 /usr/cluster/bin/cxfs_client
```

   - Monitor the `SYSLOG` on the active metadata server and look for a cell discovery message for the client and a membership delivered message containing the client cell. For example (line breaks added for readability):

```
Apr 18 13:07:21 4A:o200a unix: WARNING: Discovered cell 2 (woody)
 [priority 1 at 128.162.240.41 via 128.162.240.34]
Apr 18 13:07:31 5A:o200a unix: NOTICE: Cell 2 (client) joined the membership
Apr 18 13:07:31 5A:o200a unix: NOTICE: Membership delivered for cells 0x7
Apr 18 13:07:31 5A:o200a unix: Cell(age): 0(9) 1(7) 2(1)
```

- Monitor the `cxfs_client` log on the client you restored and look for filesystem mounts that are processing successfully. For example:

```
Apr 18 13:06:56 cxfs_client: cis_setup_fses Mounted green0: green0 on /cxfs/green0
```

- Use the following command to show that filesystems are mounted:

  ```
  client# df -hl
  ```

4. Verify that the configuration is consistent and contains the client by running the following on each active metadata server and comparing the output:

   ```
   MDS# /usr/cluster/bin/clconf_info
   ```

Stopping CXFS for the Entire Cluster

To stop CXFS for the entire cluster, do the following:

1. Stop CXFS services on a client-only node:

   ```
   client# service cxfs_client stop
   ```

   Repeat this step on each client-only node.

2. Stop GRIOv2 services on each server-capable administration node that is running GRIOv2:

   ```
   admin# service grio2 stop
   ```

   Repeat this step on each server-capable administration node that is running GRIOv2.

3. Stop CXFS services on a server-capable administration node:

   ```
   admin# service cxfs stop
   ```

   Repeat this step on each server-capable administration node.

4. Stop the cluster daemons on a server-capable administration node:

   ```
   admin# service cxfs_cluster stop
   ```

   Repeat this step on each server-capable administration node.

Restarting the Entire Cluster

To restart the entire cluster, do the following:

1. Start the cluster daemons on a server-capable administration node:

   ```
   admin# service cxfs_cluster start
   ```

   Repeat this step on each server-capable administration node.

2. Start CXFS services on a server-capable administration node:

   ```
   admin# service cxfs start
   ```

   Repeat this step on each server-capable administration node.

3. Start GRIOv2 services on a each server-capable administration node (if running GRIOv2):

   ```
   admin# service grio2 start
   ```

   Repeat this on each server-capable administration node (if running GRIOv2).

4. Start CXFS services on a client-only node:

   ```
   client# service cxfs_client start
   ```

   Repeat this step on each client-only node.

XVM Volume Mapping to Storage Targets

The `cxfs-enumerate-wwns` script enumerates the worldwide names (WWNs) on the host that are known to CXFS. You can use the `cxfs-enumerate-wwns` script to map XVM volumes to storage targets:

```
admin# /var/cluster/clconfd-scripts/cxfs-enumerate-wwns | grep -v "#"| sort -u
```

XVM Failover V2

This section discusses the following:

- "XVM Failover Concepts" on page 315
- "`failover2.conf` File Concepts" on page 316

- "Generating a `failover2.conf` File" on page 317
- "Failover V2 Examples" on page 318
- "XVM Commands Related to Failover V2" on page 320
- "RAID Units and XVM Failover V2" on page 321

XVM Failover Concepts

The example in Figure 12-6 shows two RAID controllers and the LUNs they own. All LUNs are visible from each controller, therefore, each LUN can be accessed by each path. However, the controller for RAID A is preferred for LUN 0 and LUN 2, and the controller for RAID B is preferred for LUN 1 and LUN 3.

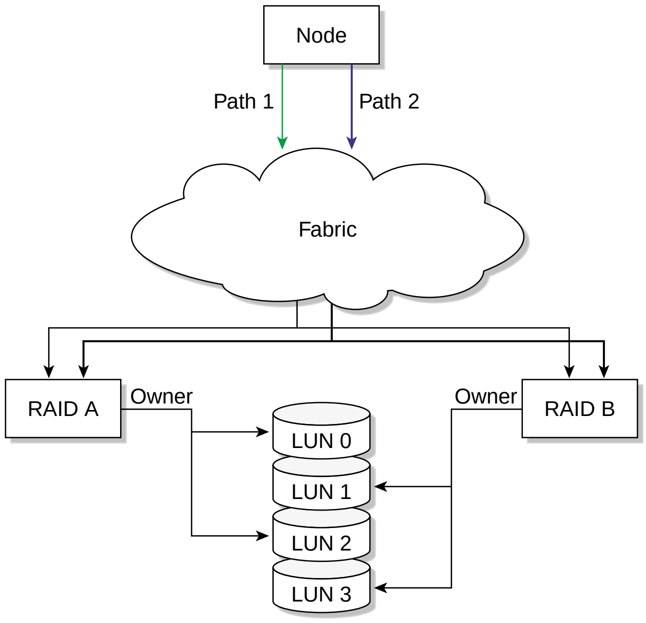


Figure 12-6 Disk Paths

Note: The Mac OS X platform provides dynamic load balancing between all paths to the same RAID controller. In this case, the system will only show one path per controller to each LUN with local HBA ports or individual paths not visible.

`failover2.conf` File Concepts

XVM failover V2 stores path information in the `failover2.conf` file. You must configure the `failover2.conf` file for each node. The entries in this file define failover attributes associated with a path to the storage. Entries can be in any order.

The `failover2.conf` file uses the following keywords:

- `preferred` indicates the best path for accessing each XVM physvol. This is the path that will be used at startup barring failure. There is no default preferred path.
- `affinity` groups all of the paths to a particular RAID controller that can be used in harmony without causing LUN ownership changes for a LUN between RAID controllers, which would result in poor disk performance. An `affinity` group for a LUN should not contain paths that go to different RAID groups. The `affinity` value also determines the order in which these groups will be tried in the case of a failure, from lowest number to highest number. The valid range of `affinity` values is `0` (lowest) through `15` (highest). The path used starts with the `affinity` of the currently used path and increases from there. For example, if the currently used path is `affinity=2`, all `affinity=2` paths are tried, then all `affinity=3`, then all `affinity=4`, and so on; after `affinity=15`, failover V2 wraps back to `affinity=0` and starts over. Before you configure the `failover2.conf` file, the initial value for all paths defaults to `affinity=0`.

  SGI recommends that the `affinity` values for a particular RAID controller be identical on every node in the CXFS cluster.

  You may find it useful to specify affinities starting at `1`. This makes it easy to spot paths that have not yet been configured because they are assigned a default of `affinity=0`. For example, if you added a new HBA but forgot to add its paths to the `failover2.conf` file, all of its paths would have an `affinity=0`, which could result in LUN ownership changes if some paths point to controller A and others point to controller B. Using this convention would not avoid this problem, but would make it easier to notice. If you use this convention, you must do so for the entire cluster.

Note: If you use the method where you do not use `affinity=0` and you do not define all of the paths in the `failover2.conf` file, you will have a affinity group using an unknown controller. If in the example where you are using `affinity=1` and `affinity=2`, if you are using `affinity=2` as your current path and there is a failover, you will failover to `affinity=0`, which could use the same RAID controller and thus fail again or might use the other RAID controller. If there are multiple unspecified paths in the `affinity=0` group, you might be mixing different RAID controllers in the same `affinity` group. This is only a performance issue, but you should fix any paths using the default `affinity=0` value by adding them to the `failover2.conf` file and using an appropriate affinity value.

You can use the `affinity` value in association with the XVM `foswitch` command to switch an XVM physvol to a physical path of a defined `affinity` value.

For more information, see:

- The example file installed in `/etc/failover2.conf.example`
- The comments in the `failover2.conf` file
- "XVM Commands Related to Failover V2" on page 320
- Platform-specific examples of `failover2.conf` in the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*
- *XVM Volume Manager Administrator's Guide*

Generating a `failover2.conf` File

The easiest method to generate a `failover2.conf` file is to run the following command on platforms other than Windows:[1]

```
MDS# xvm show -v phys | grep affinity > templatefile
```

The entries in the output only apply to already-labeled devices. Values within `< >` angle brackets are comments; you can delete them or ignore them.

1 For information about generating a `failover2.conf` file for Windows, see the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

If all nodes have correctly configured `failover2.conf` files, an affinity change in one node will signal all other nodes in the cluster that the node has changed affinity for a LUN, allowing the other nodes to change to the same affinity (the same RAID controller). You can also use the `foswitch -cluster` command to cause all nodes in a cluster to either return to their preferred paths or move to a specific affinity. See "XVM Commands Related to Failover V2" on page 320.

Failover V2 Examples

This section provides the following examples:

- "Example Using Two Affinities" on page 318
- "Example Using Four Affinities" on page 319
- "Example for Two RAID Controllers" on page 320

Example Using Two Affinities

The following example groups the paths for `lun3` and the paths for `lun4`:

```
/dev/xscsi/pci0004:00:01.1/node200900a0b813b982/port1/lun3/disc  affinity=1
/dev/xscsi/pci0004:00:01.1/node200900a0b813b982/port2/lun3/disc  affinity=1
/dev/xscsi/pci0004:00:01.0/node200900a0b813b982/port1/lun3/disc  affinity=1
/dev/xscsi/pci0004:00:01.0/node200900a0b813b982/port2/lun3/disc  affinity=1   preferred
/dev/xscsi/pci0004:00:01.1/node200800a0b813b982/port1/lun3/disc  affinity=3
/dev/xscsi/pci0004:00:01.0/node200800a0b813b982/port1/lun3/disc  affinity=3
/dev/xscsi/pci0004:00:01.1/node200800a0b813b982/port2/lun3/disc  affinity=3
/dev/xscsi/pci0004:00:01.0/node200800a0b813b982/port2/lun3/disc  affinity=3

/dev/xscsi/pci0004:00:01.1/node200900a0b813b982/port1/lun4/disc, affinity=1
/dev/xscsi/pci0004:00:01.1/node200900a0b813b982/port2/lun4/disc, affinity=1
/dev/xscsi/pci0004:00:01.0/node200900a0b813b982/port1/lun4/disc, affinity=1
/dev/xscsi/pci0004:00:01.0/node200900a0b813b982/port2/lun4/disc, affinity=1
/dev/xscsi/pci0004:00:01.1/node200800a0b813b982/port1/lun4/disc, affinity=3
/dev/xscsi/pci0004:00:01.1/node200800a0b813b982/port2/lun4/disc, affinity=3 preferred
/dev/xscsi/pci0004:00:01.0/node200800a0b813b982/port1/lun4/disc, affinity=3
/dev/xscsi/pci0004:00:01.0/node200800a0b813b982/port2/lun4/disc, affinity=3
```

The order of paths in the file is not significant. Paths to the same LUN are detected automatically. Without this file, all paths to each LUN would have `affinity=0` and there would be no preferred path. Setting a `preferred` path ensures that multiple paths will be used for performance. If no path is designated as `preferred`, the path used to the LUN is arbitrary based on the order of device discovery. There is no interaction between the preferred path and the affinity values.

This file uses affinity to group the RAID controllers for a particular path. Each controller has been assigned an affinity value. It shows the following:

- There is one PCI card with two ports off of the HBA (`pci04.01.1` and `pci04.01.0`)
- There are two RAID controllers, `node200800a0b813b982` and `node200900a0b813b982`
- Each RAID controller has two ports that are identified by `port1` or `port2`
- Each LUN has eight paths (via two ports on a PCI card, two RAID controllers, and two ports on the controllers)
- There are two affinity groups for each LUN, `affinity=1` and `affinity=3`
- There is a `preferred` path for each LUN

Failover will exhaust all paths to `lun3` from RAID controller `node200900a0b813b982` (with `affinity=1` and the `preferred` path) before moving to RAID controller `node200800a0b813b982` paths (with `affinity=3`)

Example Using Four Affinities

The following example uses four affinities to associate the two HBA ports with each of the available two ports on the RAID's two controllers:

```
/dev/xscsi/pci0004:00:01.1/node200900a0b813b982/port1/lun4/disc  affinity=1
/dev/xscsi/pci0004:00:01.1/node200900a0b813b982/port2/lun4/disc  affinity=2
/dev/xscsi/pci0004:00:01.0/node200900a0b813b982/port1/lun4/disc  affinity=1
/dev/xscsi/pci0004:00:01.0/node200900a0b813b982/port2/lun4/disc  affinity=2
/dev/xscsi/pci0004:00:01.1/node200800a0b813b982/port1/lun4/disc  affinity=4
/dev/xscsi/pci0004:00:01.1/node200800a0b813b982/port2/lun4/disc  affinity=3 preferred
/dev/xscsi/pci0004:00:01.0/node200800a0b813b982/port1/lun4/disc  affinity=4
/dev/xscsi/pci0004:00:01.0/node200800a0b813b982/port2/lun4/disc  affinity=3
```

Each affinity associates the two host adapter ports with a single RAID controller port. The declaration of these eight associations completely defines all of the available paths to a single RAID LUN.

These eight associations also represent the order in which the paths are tried in a failover situation. Failover begins by trying the other paths within the current affinity and proceeds in a incremental manner through the affinities until either a working path is discovered or all possible paths have been tried. The paths will be tried in the following order:

1. `affinity=3` (the affinity of the current path), which is associated with RAID controller A port 2
2. `affinity=4`, which is associated with RAID controller A port 1
3. `affinity=1`, which is associated with raid controller B port 1
4. `affinity=2`, which is associated with raid controller B port 2

Example for Two RAID Controllers

The following example for IRIX shows two RAID controllers, `200800a0b818b4de` and `200900a0b818b4de` for `lun4vol`:

```
/dev/dsk/200800a0b818b4de/lun4vol/c2p2 affinity=1 preferred
/dev/dsk/200800a0b818b4de/lun4vol/c2p1 affinity=1
/dev/dsk/200900a0b818b4de/lun4vol/c2p2 affinity=3
/dev/dsk/200900a0b818b4de/lun4vol/c2p1 affinity=3
```

XVM Commands Related to Failover V2

The following are useful XVM commands related to failover V2:

```
xvm help -verbose foconfig
xvm help -verbose foswitch
xvm help -verbose show
xvm foconfig -init
xvm foswitch -cluster -preferred physvol/name   (switch phys/name in all nodes in cluster to preferred path)
xvm foswitch -preferred physvol
xvm foswitch -affinity 1 physvol
xvm foswitch -dev newdev
xvm foswitch -cluster -affinity 1 phys
xvm foswitch -cluster -setaffinity X phys/name  (switch phys/name in cluster to affinity "X")
xvm show -verbose physvol
xvm show -verbose physvol | fgrep affinity > templatefile
```

For details, see the *XVM Volume Manager Administrator's Guide*.

Note: The `xvm` command is provided on all CXFS platforms. However, client-only nodes support only read-only commands.

RAID Units and XVM Failover V2

This section discusses the following:

- "TP9100, RM610/660, and Failover V2" on page 321
- " TP9300, TP9500, TP9700, and S330 and Failover V2" on page 321
- "SGI InfiniteStorage 220 and Failover V2" on page 322

For more information about firmware levels, see "RAID Firmware" on page 78.

TP9100, RM610/660, and Failover V2

The TP9100 and RM610/660 RAID units do not have any host type failover configuration. Each LUN should be accessed via the same RAID controller for each node in the cluster because of performance reasons. These RAIDs behave and have the same characteristics as the `SGIAVT` mode discussed below.

TP9100 1 GB and 2 GB `SGIAVT` mode requires that the array is set to `multitid`.

TP9300, TP9500, TP9700, and S330 and Failover V2

The TP9300, TP9500, and TP9700 RAID `SGIAVT` mode has the concept of LUN ownership by a single RAID controller. However, LUN ownership change will take place if any I/O for a given LUN is received by the RAID controller that is not the current owner. The change of ownership is automatic based on where I/O for a LUN is received and is not done by a specific request from a host failover driver. The concern with this mode of operation is that when a node in the cluster changes I/O to a different RAID controller than that used by the rest of the cluster, it can result in severe performance degradation for the LUN because of the overhead involved in constantly changing ownership of the LUN.

Failover V2 requires that you configure TP9300, TP9500, TP9700, and S330 RAID units with `SGIAVT` host type and the 06.12.18.*xx* code or later be installed.

TP9700 use of `SGIAVT` requires that 06.15.17*xx.* code or later be installed.

SGI InfiniteStorage 220 and Failover V2

XVM failover V2 support requires SGI ProPack 5 SP 1 or later.

GPT Labels

You can create these labels on SGI ProPack server-capable administration nodes and Linux third-party clients. The GPT label puts header data in sector 1 of a LUN, leaving sector 0 for a master boot record. Partition information is stored in a variable number of sectors, starting at sector 2. XVM requires two partitions on a GPT-labeled LUN, one for XVM metadata and the other for the user data. XVM assumes ownership of the LUN and access to a particular LUN could be fenced.

Note: CXFS supports a GPT-labeled LUN greater than 2 TB in size. However, being able to label a LUN does not mean that the system is able to recognize and use it. The operating systems in the cluster will determine whether you can actually use a LUN of a given size. If a LUN is set up as greater than 2–TB in size but if the operating system of a node in a cluster cannot support a greater-than–2–TB LUN, then this node will not be able to share or even access data on this LUN.

When creating a GPT partition table for XVM to use, the first partition size should be at least 2 MB (just large enough to hold the XVM metadata, such as volume and slice information). The second partition for the volume data should be the rest of the LUN. You can place the start of the second partition anywhere after the first partition that will give good performance, such as on a boundary of the RAID's stripe width.

If the operating system is capable of specifying the start of a partition as a sector number, place the start of data exactly on a boundary for good performance. For SUSE Linux Enterprise Server (SLES) 10, you can use the `mkpart` command to `parted` to specify the sector number using an `s` suffix. For example, to make a partition starting at 2–MB into the LUN and ending at 961085440 (0x32490000), also a 2–MB byte boundary:

```
(parted) mkpart primary 4096s 961085440s
```

You can also use the `unit s` command to `parted` to set the input and display so that they default to sectors as the unit size. The partition for XVM data (partition 2) should have a start sector and length that is a common multiple of the RAID LUN's stripe width and the 16–KB page size for Altix or 4–KB page size for Altix XE. (If the partition is not made this way, the `xvm slice` command has options that you can use to place the slice on these boundaries.) If LUNs are then concatenated, I/O will be less likely to span RAID stripe-width boundaries or cause a read-modify-write inside the RAID if partial stripes are written.

For example, using a size with 2–MB boundaries:

```
(parted) unit s
(parted) print
Disk geometry for /dev/sdg: 0s - 71687371s
Disk label type: gpt
Number  Start   End     Size    File system  Name                 Flags
1       34s     3906s   3873s
(parted) mkpart
Partition type?  [primary]?
File system type?  [ext2]? xfs
Start? 4096
End? 71687168
(parted) print
Disk geometry for /dev/sdg: 0s - 71687371s
Disk label type: gpt
Number  Start   End     Size    File system  Name                 Flags
1       34s     3906s   3873s
2       4096s   71687168s 71683204s
```

For operating systems other than SLES 10, see the operating system documentation.

For more information, see the *XVM Volume Manager Administrator's Guide*.

Generation of Streaming Workload for Video Streams

To generate streaming workload for SD/HD/2K/4K formats of video streams, you can use the `frametest`(1) command. Each frame is stored in a separate file. You can also use this tool to simulate the reading and writing video streams by streaming applications. The tool also generates the performance statistics for the reading and writing operation, so it can be very useful for performance analysis for streaming applications.

For example, to do a multithreaded (4 threads) write test of 20,000 HD frames, as fast as possible (the `dir` directory should contain 20,000 HD frames created by a previous write test):

```
# frametest -t4 -w hd -n20000 -x frametest_w_t4_hd_20000_flatout.csv dir
```

To use 24 frames per second using a buffer of 24 frames:

```
# frametest -t4 -n20000 -f24 -q24 -g frametest_r_t4_hd_20000_24fps_24buf.csv dir
```

For details about `frametest` and its command-line options, see the `frametest`(1)

Note: Using the `frametest` command on AIX requires that the `posix_aio0` device is available. For more information, see the Solaris chapter in *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

Frame Files Defragmentation and Analysis

The `framesort` utility provides easy file-layout analysis and advanced file-sequence reorganization:

File-layout analysis shows the following:

- How well the specified files are allocated
- How many same-sized files are interleaved
- The number of runs where files are allocated in consecutive order or in reverse consecutive order

File-sequence reorganization makes files with consecutive filenames be placed consecutively in storage. It can also align files to their stripe-unit boundary. After rearrangement, files can can gain higher retrieval bandwidth, which is essential for frame playback.

Note: On IRIX systems, files larger than the specified `maxdmasz` size are not processed. (`maxdmasz` is the system tunable parameter that sets the maximum direct-memory access size). For more information about `maxdmasz`, see *IRIX Admin: System Configuration and Operation*.

For example, the following command line will do analysis and rearrangement recursively starting from directory `movie1`. It also displays the progress status and verbose information. If the percentage of poorly organized files is equal to or greater than 15%, the rearrangement is triggered:

```
# framesort -rdgvva 15 movie1
```

For details about command-line arguments, see the `framesort`(1) man page.

Disk Layout Optimization for Approved Media Customers

This section discusses the following:

- "Ideal Frame Layout" on page 325
- "Multiple Streams of Real-Time Applications" on page 326
- "The `filestreams` Mount Option" on page 328

Ideal Frame Layout

An ideal frame layout is one in which frames for each stream are written sequentially on disk to maximize bandwidth and minimize latency:

- Minimize seek times while reading and writing
- Maximize RAID prefetch into cache for reads
- Maximize RAID coalescing writes into larger writes to each disk

Figure 12-7 shows an ideal frame layout

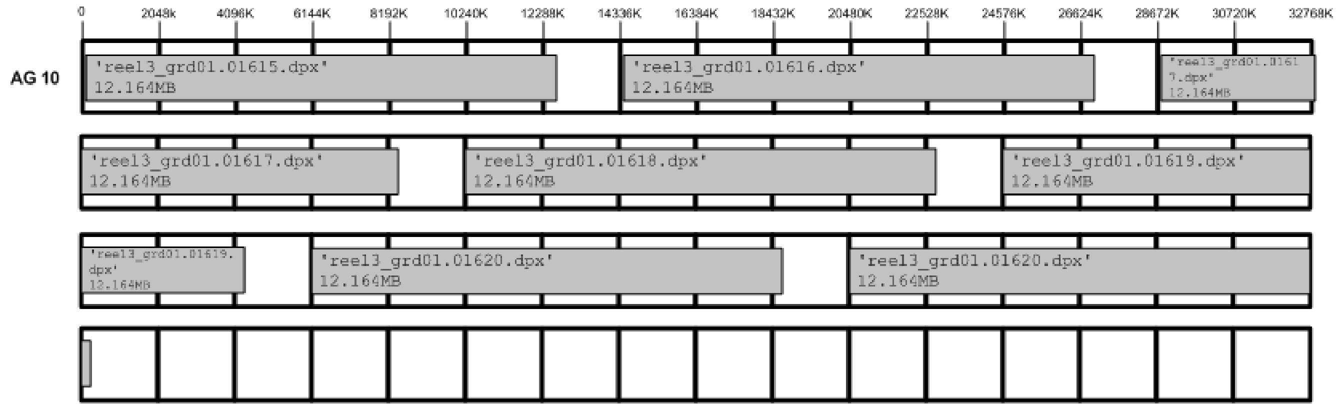

0
2048k
4096K
6144K
8192K
10240K
12288K
14336K
16384K
18432K
20480K
22528K
24576K
26624K
28672K
30720K
32768K
AG 10
'reel3_grd01.01615.dpx' 12.164MB
'reel3_grd01.01616.dpx' 12.164MB
'reel3_grd01.0161 7.dpx' 12.164MB
'reel3_grd01.01617.dpx' 12.164MB
'reel3_grd01.01618.dpx' 12.164MB
'reel3_grd01.01619.dpx' 12.164MB
'reel3_grd01.01619. dpx' 12.164MB
'reel3_grd01.01620.dpx' 12.164MB
'reel3_grd01.01620.dpx' 12.164MB

Figure 12-7 Ideal Frame Layout

With multithreaded applications (such as `frametest`), there will be multiple requests in flight simultaneously. As each frame is requested, data from upcoming frames will be prefetched into cache. Figure 12-8 shows an example of a 4-thread frametest read (2-MB stripe unit / 1-GB cache size/ prefetch = x1 / 16 slices).

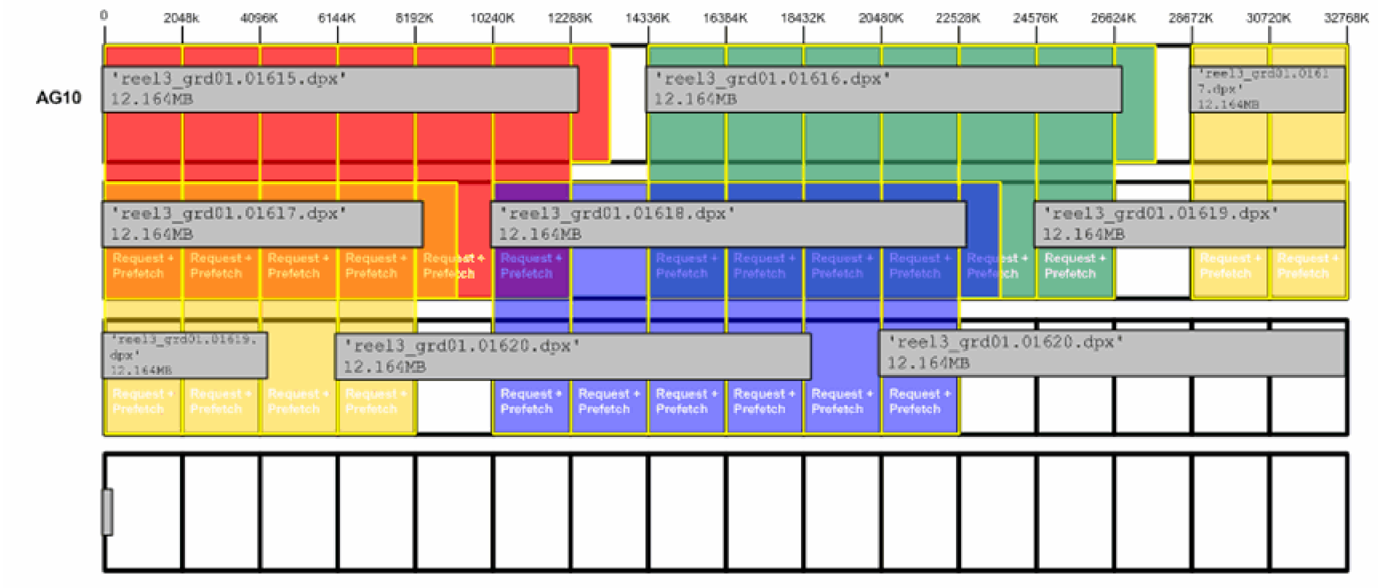

0
2048k
4096K
6144K
8192K
10240K
12288K
14336K
16384K
18432K
20480K
22528K
24576K
26624K
28672K
30720K
32768K
AG10
'reel3_grd01.01615.dpx' 12.164MB
'reel3_grd01.01616.dpx' 12.164MB
'reel3_grd01.01617.dpx' 12.164MB
'reel3_grd01.01618.dpx' 12.164MB
'reel3_grd01.01619.dpx' 12.164MB
'reel3_grd01.01620.dpx' 12.164MB
Request + Prefetch

Figure 12-8 Ideal Frame Layout with RAID Prefetch

Multiple Streams of Real-Time Applications

When there are multiple streams of real-time applications, frames from each stream are interleaved into the same region. Frames are not written sequentially but will jump forwards and backwards in the filesystem. The RAID is unable to support many real-time streams and is unable to maintain frame rates due to additional back-end I/O. Filesystems allocate files based on algorithms to utilize free space, not to maximize RAID performance when reading streams back. Figure 12-9 shows an example.

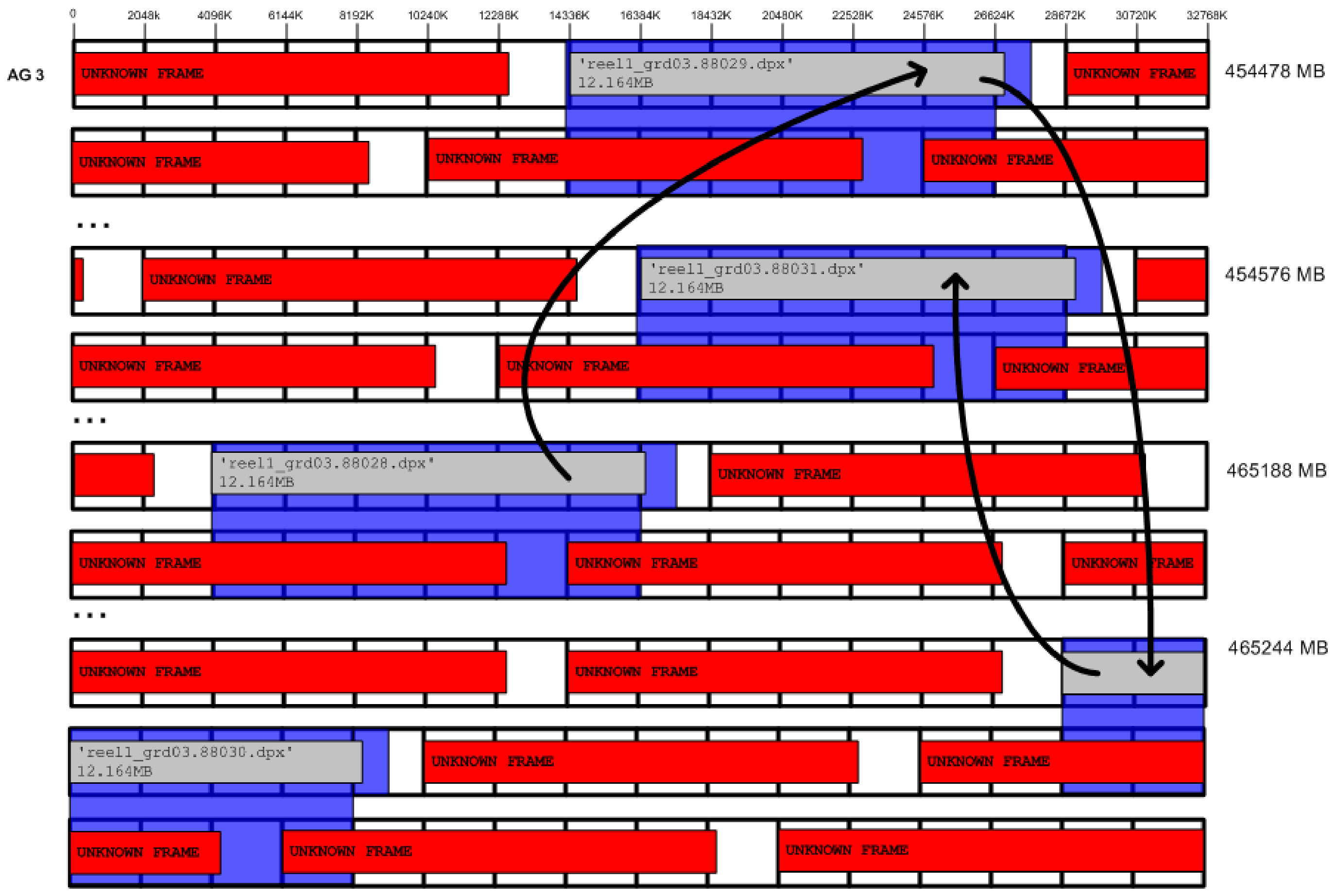
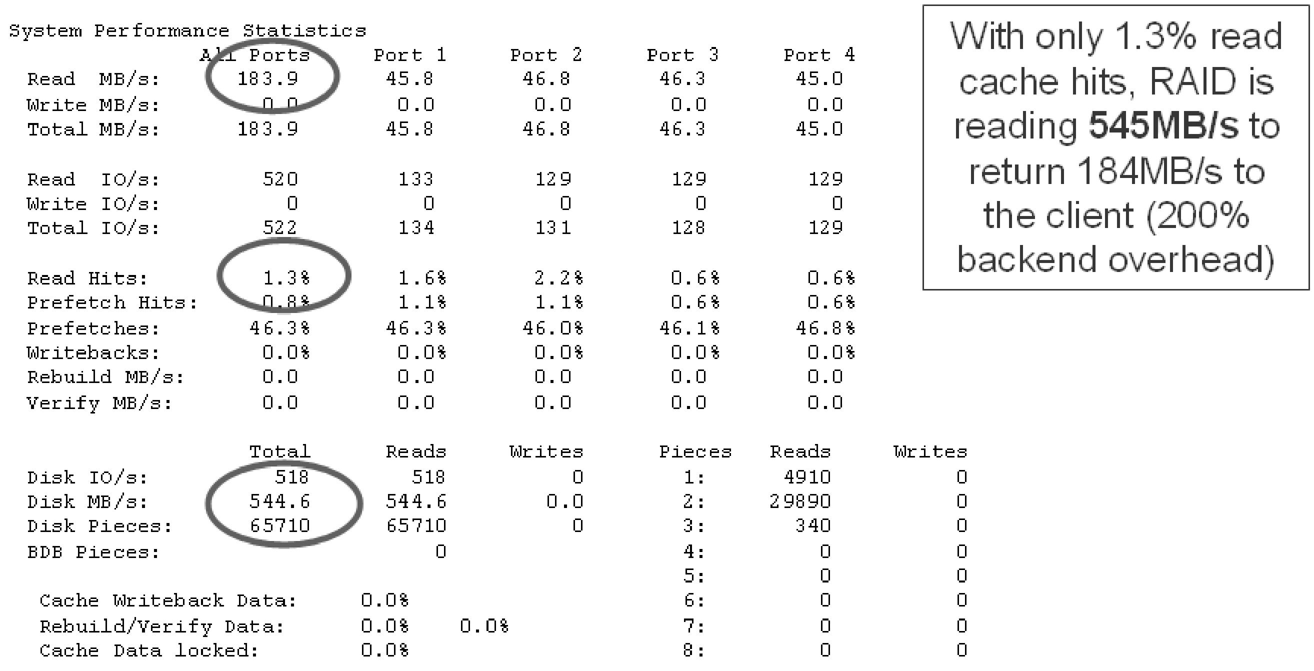


Figure 12-9 Multiple Streams of Real Applications

Figure 12-10 shows an example of poor cache utilization.



```
System Performance Statistics
                  All Ports     Port 1      Port 2      Port 3      Port 4
 Read  MB/s:       183.9         45.8        46.8        46.3        45.0
 Write MB/s:         0.0          0.0         0.0         0.0         0.0
 Total MB/s:       183.9         45.8        46.8        46.3        45.0

 Read  IO/s:         520          133         129         129         129
 Write IO/s:           0            0           0           0           0
 Total IO/s:         522          134         131         128         129

 Read Hits:          1.3%         1.6%        2.2%        0.6%        0.6%
 Prefetch Hits:      0.8%         1.1%        1.1%        0.6%        0.6%
 Prefetches:        46.3%        46.3%       46.0%       46.1%       46.8%
 Writebacks:         0.0%         0.0%        0.0%        0.0%        0.0%
 Rebuild MB/s:       0.0          0.0         0.0         0.0         0.0
 Verify MB/s:        0.0          0.0         0.0         0.0         0.0

                   Total        Reads      Writes      Pieces    Reads      Writes
 Disk IO/s:          518          518           0         1:      4910           0
 Disk MB/s:        544.6        544.6         0.0         2:     29890           0
 Disk Pieces:      65710        65710           0         3:       340           0
 BDB Pieces:                        0                     4:         0           0
                                                          5:         0           0
  Cache Writeback Data:     0.0%                          6:         0           0
  Rebuild/Verify Data:      0.0%     0.0%                 7:         0           0
  Cache Data locked:        0.0%                          8:         0           0
```

With only 1.3% read cache hits, RAID is reading **545MB/s** to return 184MB/s to the client (200% backend overhead)

Figure 12-10 Poor Cache Utilization

The `filestreams` Mount Option

Approved media customers can use the XFS `filestreams` mount option with CXFS to maximize the ability of storage to support multiple real-time streams of video data. It is appropriate for workloads that generate many files that are created and accessed in a sequential order in one directory.

Caution: SGI must validate that your RAID model and RAID configuration can support the use of the `filestreams` mount option to achieve real-time data transfer and that your application is appropriate for its use. Use of this feature is complex and is reserved for designs that have been approved by SGI.

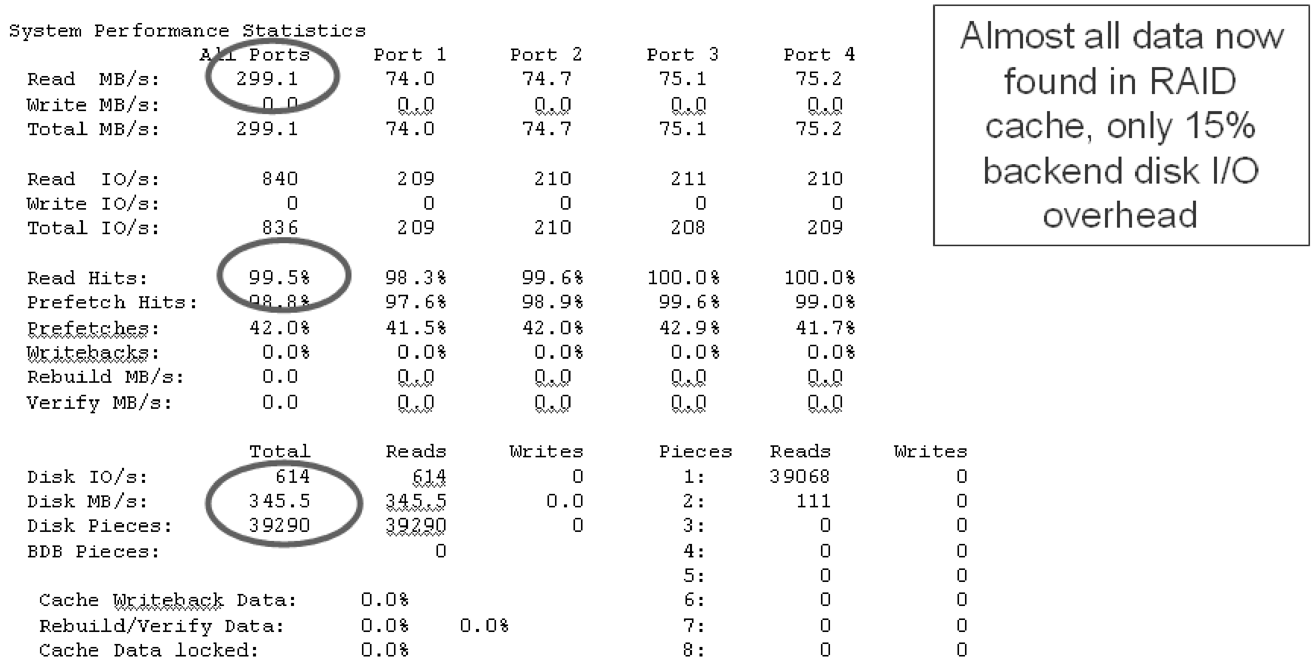
The `filestreams` mount option changes the behavior of the XFS allocator in order to optimize disk layout. It selects an XFS disk block allocation strategy that does the following:

- Identifies streams writing into the same directory and locks down a region of the filesystem for that stream, which prevents multiple streams from using the same allocation groups
- Allocates the file data sequentially on disk in the order that the files are created, space permitting
- Uses different regions of the filesystem for files in different directories

Using the `filestreams` mount option can improve both bandwidth and latency when accessing the files because the RAID will be able to access the data in each directory sequentially. Therefore, multiple writers may be able to write into the same filesystem without interleaving file data on disk. Filesystem can be filled up to approximately 94% before performance degrades. Deletion of projects does not fragment filesystem, therefore there is no need to rebuild filesystem after each project.

You can safely enable the `filestreams` mount option on an existing filesystem and later disable it without affecting compatibility. (The mount option affects where data is located in the filesystem; it does not change the format of the filesystem.) However, you may not get the full benefit of `filestreams` due to preexisting filesystem fragmentation.

Figure 12-11 shows an example of excellent cache utilization that allows for more streams.



```
System Performance Statistics
                    All Ports      Port 1      Port 2      Port 3      Port 4
  Read  MB/s:        299.1         74.0        74.7        75.1        75.2
  Write MB/s:          0.0          0.0         0.0         0.0         0.0
  Total MB/s:        299.1         74.0        74.7        75.1        75.2

  Read  IO/s:          840          209         210         211         210
  Write IO/s:            0            0           0           0           0
  Total IO/s:          836          209         210         208         209

  Read Hits:          99.5%        98.3%       99.6%      100.0%      100.0%
  Prefetch Hits:      98.8%        97.6%       98.9%       99.6%       99.0%
  Prefetches:         42.0%        41.5%       42.0%       42.9%       41.7%
  Writebacks:          0.0%         0.0%        0.0%        0.0%        0.0%
  Rebuild MB/s:        0.0          0.0         0.0         0.0         0.0
  Verify MB/s:         0.0          0.0         0.0         0.0         0.0

                     Total        Reads      Writes      Pieces   Reads      Writes
  Disk IO/s:           614          614          0           1:   39068          0
  Disk MB/s:         345.5        345.5        0.0           2:     111          0
  Disk Pieces:       39290        39290          0           3:       0          0
  BDB Pieces:                         0                      4:       0          0
                                                             5:       0          0
   Cache Writeback Data:      0.0%                           6:       0          0
   Rebuild/Verify Data:       0.0%     0.0%                  7:       0          0
   Cache Data locked:         0.0%                           8:       0          0
```

Almost all data now found in RAID cache, only 15% backend disk I/O overhead

Figure 12-11 Excellent Cache Utilization

For more information, contact SGI Support.

Chapter 13

Cluster Database Management

This chapter contains the following:

Performing Cluster Database Backup and Restoration

You should perform a database backup whenever you want to save the database and be able to restore it to the current state at a later point.

You can use the following methods to restore the database:

- If the database is accidentally deleted from a server-capable administration node, use the `fs2d` daemon to replicate the database from another server-capable administration node. See "Restoring a Deleted Database from Another Node" on page 331.
- If you want to be able to recreate the current configuration, use the `config` command in `cxfs_admin`. You can then recreate this configuration by using the output file and the `cxfs_admin -f` option or running the script generated as described in "Saving and Recreating the Current Configuration with `cxfs_admin`" on page 269.
- If you want to retain a copy of the database and all node-specific information such as local logging, use the `cdbBackup` and `cdbRestore` commands. You should periodically backup the cluster database on all server-capable administration nodes using the `cdbBackup` command either manually or by adding an entry to the `root crontab` file. See "Using `cdbBackup` and `cdbRestore` for the Cluster Database and Logging Information" on page 333.

Restoring a Deleted Database from Another Node

If the database has been accidentally deleted from an individual server-capable administration node, you can restore it by synchronizing with the database on another server-capable administration node.

Note: Do not use this method if the cluster database has been **corrupted**, because the database on another node will also be corrupted. In the case of corruption, you must reinstate a backup copy of the database. See "Saving and Recreating the Current Configuration with `cxfs_admin`" on page 269.

Do the following:

1. Stop the CXFS service on the server-capable administration node with the deleted database by running the following command in `cxfs_admin`:

   Caution: If you omit this step, the target node might be reset by another server-capable administration node.

   ```
   cxfs_admin:cluster> disable nodename
   ```

2. *(If running GRIOv2)* Stop the GRIOv2 daemon (`ggd2`) by running the following command on the node with the deleted database:

   ```
   admin# service grio2 stop
   ```

3. Stop the CXFS control daemon (`clconfd`) by running the following command on the node where step 2 was executed:

   Caution: Running this command will completely shut down all CXFS filesystem access on the local node.

   ```
   admin# service cxfs stop
   ```

4. Stop the CXFS cluster administration daemons (`cad`, `cmond`, `crsd`, and `fs2d`) by running the following command on the node where step 2 was executed:

   ```
   admin# service cxfs_cluster stop
   ```

5. Run `cdbreinit` on the node where step 2 was executed:

   ```
   admin# /usr/cluster/bin/cdbreinit
   ```

6. Wait for the following message to be logged to the `syslog` :

   ```
   fs2d[PID]: Finished receiving CDB sync series from machine nodename
   ```

7. Start the CXFS control daemon by running the following command on the node where step 2 was executed:

   ```
   admin# service cxfs start
   ```

8. *(If running GRIOv2)* Start the GRIOv2 daemon (`ggd2`) by running the following command on the node where step 2 was executed:

   ```
   admin# service grio2 start
   ```

The `cdbreinit` command will restart cluster daemons automatically. The `fs2d` daemon will then replicate the cluster database to the node from which it is missing.

Using `cdbBackup` and `cdbRestore` for the Cluster Database and Logging Information

The `cdbBackup` and `cdbRestore` commands backup and restore the cluster database and node-specific information, such as local logging information. You must run these commands individually for each server-capable administration node.

To perform a backup of the cluster, use the `cdbBackup` command on each server-capable administration node.

Caution: Do not make configuration changes while you are using the `cdbBackup` command.

To perform a restore, run the `cdbRestore` command on each server-capable administration node. You can use this method for either a missing or a corrupted cluster database. Do the following:

1. Stop CXFS services on all nodes in the cluster.
2. Stop the cluster administration daemons on each server-capable administration node.
3. Remove the old database by using the `cdbreinit` command on each server-capable administration node.
4. Stop the cluster administration daemons again (these were restarted automatically by `cdbreinit` in the previous step) on each node.
5. Use the `cdbRestore` command on each server-capable administration node.

6. Start the cluster administration daemons on each server-capable administration node.

For example, to backup the current database, clear the database, and then restore the database to all server-capable administration nodes, do the following as directed:

On each server-capable administration node:
```
# /usr/cluster/bin/cdbBackup
```

On one server-capable administration node:
```
cmgr> stop cx_services for cluster clusterA
```

On each server-capable administration node (if running GRIOv2):
```
# service grio2 stop
```

On each server-capable administration node:
```
# service cxfs stop
```

On each server-capable administration node:
```
# service cxfs_cluster stop
```

On each server-capable administration node:
```
# /usr/cluster/bin/cdbreinit
```

On each node server-capable administration (again):
```
# service cxfs_cluster stop
```

On each server-capable administration node:
```
# /usr/cluster/bin/cdbRestore
```

On each server-capable administration node:
```
# service cxfs_cluster start
```

On each server-capable administration node:
```
# service cxfs start
```

On each server-capable administration node (if running GRIOv2):
```
# service grio2 start
```

For more information, see the `cdbBackup`(1M) and `cdbRestore`(1M) man pages.

Validating the Cluster Configuration with `cxfs-config`

The `cxfs-config` command validates configuration information in the cluster database. You can run it on any server-capable administration node in the cluster.

By default, `cxfs-config` displays the following:

- Cluster name and cluster ID
- Tiebreaker node
- Networks for CXFS failover networks
- Nodes in the pool:
  - Node ID
  - Cell ID (as assigned by the kernel when added to the cluster and stored in the cluster database)
  - Status of CXFS services (configured to be enabled or disabled)
  - Operating system
  - Node function
- CXFS filesystems:
  - Name, mount point (`enabled` means that the filesystem is configured to be mounted; if it is not mounted, there is an error)
  - Device name
  - Mount options
  - Potential metadata servers
  - Nodes that should have the filesystem mounted (if there are no errors)
  - Switches:
    - Switch name, user name to use when sending a `telnet` message, mask (a hexadecimal string representing a 64-bit port bitmap that indicates the list of ports in the switch that will not be fenced)
    - Ports on the switch that have a client configured for fencing at the other end
- Warnings or errors

For example:

```
thump# /usr/cluster/bin/cxfs-config
Global:
    cluster: topiary (id 1)
    cluster state: enabled
    tiebreaker: <none>

Networks:
    net 0: type tcpip  192.168.0.0      255.255.255.0
    net 1: type tcpip  134.14.54.0      255.255.255.0

Machines:
    node leesa: node 6     cell 1  enabled  Linux32 client_only
        hostname: leesa.example.com
        fail policy: Fence
        nic 0: address: 192.168.0.164 priority: 1 network: 0
        nic 1: address: 134.14.54.164 priority: 2 network: 1

    node thump: node 1     cell 0  enabled  Linux64    server_admin
        hostname: thump.example.com
        fail policy: Fence
        nic 0: address: 192.168.0.186 priority: 1 network: 0
        nic 1: address: 134.14.54.186 priority: 2 network: 1

Filesystems:
    fs dxm: /mnt/dxm             enabled
        device = /dev/cxvm/tp9500a4s0
        options = []
        servers = thump (0)
        clients = leesa, thump

Switches:
    switch 0: admin@asg-fcsw1      mask 0000000000000000
        port 8: 210000e08b0ead8c thump

    switch 1: admin@asg-fcsw0      mask 0000000000000000

Warnings/errors:
    enabled machine leesa has fencing enabled but is not present in switch database
```

The following options are of particular interest:

- `-all` lists all available information
- `-ping` contacts each NIC in the machine list and displays if the packets is transmitted and received. For example:

```
node leesa: node 6     cell 1  enabled  Linux32 client_only
   fail policy: Fence
   nic 0: address: 192.168.0.164 priority: 1
       ping: 5 packets transmitted, 5 packets received, 0.0% packet loss
       ping: round-trip min/avg/max = 0.477/0.666/1.375 ms
   nic 1: address: 134.14.54.164 priority: 2
       ping: 5 packets transmitted, 5 packets received, 0.0% packet loss
       ping: round-trip min/avg/max = 0.469/0.645/1.313 ms
```

- `-xfs` lists XFS information for each CXFS filesystem, such as size. For example:

```
Filesystems:
    fs dxm: /mnt/dxm             enabled
        device = /dev/cxvm/tp9500a4s0
        options = []
        servers = thump (0)
        clients = leesa, thump
        xfs:
            magic: 0x58465342
            blocksize: 4096
            uuid: 3459ee2e-76c9-1027-8068-0800690dac3c
            data size 17.00 Gb
```

- `-xvm` lists XVM information for each CXFS filesystem, such as volume size and topology. For example:

```
Filesystems:
    fs dxm: /mnt/dxm             enabled
        device = /dev/cxvm/tp9500a4s0
        options = []
        servers = thump (0)
        clients = leesa,  thump
        xvm:
            vol/tp9500a4s0                   0 online,open
                subvol/tp9500a4s0/data     35650048 online,open
```

```
                    slice/tp9500a4s0           35650048 online,open

            data size: 17.00 Gb
```

- `-check` performs extra verification, such as XFS filesystem size with XVM volume size for each CXFS filesystem. This option may take a few moments to execute.

For a complete list of options, see the `cxfs-config`(1m) man page.

The following example shows errors reported by `cxfs-config`:

```
aiden # /usr/cluster/bin/cxfs-config -check -all
Global:
    cluster: BP (id 555)
    cluster state: enabled
    tiebreaker:
Networks:
    net 0: type tcpip  10.11.0.0        255.255.255.0
    net 1: type tcpip  128.162.242.0    255.255.255.0

Machines:
    node aiden: node 27560 cell 0  enabled  Linux64    server_admin
        hostname: aiden.example.com
        fail policy: Fence, Shutdown
        nic 0: address: 10.11.0.241 priority: 1 network: 0
            ping: 5 packets transmitted, 5 packets received, 0.0% packet loss
            ping: round-trip min/avg/max = 0.136/0.171/0.299 ms
        nic 1: address: 128.162.242.12 priority: 2 network: 1
            ping: 5 packets transmitted, 5 packets received, 0.0% packet loss
            ping: round-trip min/avg/max = 0.130/0.171/0.303 ms

    node brigid: node 31867 cell 2  enabled  Linux64    server_admin
        hostname: brigid.example.com
        fail policy: Fence, Shutdown
        nic 0: address: 10.11.0.240 priority: 1 network: 0
            ping: 5 packets transmitted, 5 packets received, 0.0% packet loss
            ping: round-trip min/avg/max = 0.303/0.339/0.446 ms
        nic 1: address: 128.162.242.11 priority: 2 network: 1
            ping: 5 packets transmitted, 5 packets received, 0.0% packet loss
            ping: round-trip min/avg/max = 0.336/0.430/0.799 ms

    node flynn: node 1     cell 1  enabled  Linux64 client_only
```

```
        hostname: flynn.example.com
        fail policy: Fence, Shutdown
        nic 0: address: 10.11.0.234 priority: 1 network: 0
            ping: 5 packets transmitted, 5 packets received, 0.0% packet loss
            ping: round-trip min/avg/max = 0.323/0.370/0.539 ms
        nic 1: address: 128.162.242.189 priority: 2 network: 1
            ping: 5 packets transmitted, 5 packets received, 0.0% packet loss
            ping: round-trip min/avg/max = 0.283/0.312/0.424 ms

Filesystems:
    fs concatfs: /concatfs            enabled
        device = /dev/cxvm/concatfs
        force = true
        options = [rw,quota]
        servers = aiden (0), brigid (2)
        clients = aiden, brigid, flynn
        xvm:
            vol/concatfs                      0 online,open
                subvol/concatfs/data     2836134016 online,open
                subvol/concatfs/data     2836134016 online,open
                    concat/concat0           2836134016 online,tempname,open
                        slice/lun2s0             1418067008 online,open
                        slice/lun3s0             1418067008 online,open

            data size: 1.32 TB
        xfs:
            magic: 0x58465342
            blocksize: 4096
            uuid: 9616ae39-3a50-1029-8896-080069056bf5
            data size 1.32 TB

    fs stripefs: /stripefs            enabled
        device = /dev/cxvm/stripefs
        force = true
        options = [rw,quota]
        servers = aiden (0), brigid (2)
        clients = aiden, brigid, flynn
        xvm:
            vol/stripefs                      0 online,open
                subvol/stripefs/data     2836133888 online,open
                    stripe/stripe0           2836133888 online,tempname,open
```

```
                        slice/lun0s0             1418067008 online,open
                        slice/lun1s0             1418067008 online,open

            data size: 1.32 TB
        xfs:
            magic: 0x58465342
            blocksize: 4096
            uuid: 9616ae38-3a50-1029-8896-080069056bf5
            data size 1.32 TB

Switches:
    switch 0: 32 port brocade admin@fcswitch12         port 12: 210000e08b00e6eb brigid
        port 28: 210000e08b041a3a aiden

    switch 1: 32 port brocade admin@fcswitch13         port 7: 210000e08b08793f flynn
        port 12: 210100e08b28793f flynn

cxfs-config warnings/errors:
    server aiden fail policy must not contain "Shutdown" for cluster with
even number of enabled servers and no tiebreaker
    server brigid fail policy must not contain "Shutdown" for cluster with
even number of enabled servers and no tiebreaker
```

Monitoring Status

You can view the system status in the following ways:

Note: Administrative tasks must be performed using one of the following tools:

- The CXFS GUI when it is connected to a server-capable administration node (a node that has the `cluster_admin` software package installed)
- `cxfs_admin` (you must logged in as `root` on a host that has permission to access the CXFS cluster database)

You must run administration commands on a server-capable administration node; you run the `cxfs_info` status command on a client-only node.

- Monitor log files. See "Status in Log Files" on page 342.
- Use the CXFS GUI or the `tail` command to view the end of the `/var/log/messages` system log file on a server-capable administration node. (You can also view the system log file on client-only nodes.)
- Keep continuous watch on the state of a cluster using the GUI view area or the following `cxfs_admin` command:

  ```
  cxfs_admin -i clustername -r -c "status interval=seconds"
  ```

- Query the status of an individual node or cluster using the GUI or `cxfs_admin`.
- Manually test the filesystems with the `ls` command.
- Monitor the system with Performance Co-Pilot. You can use Performance Co-Pilot to monitor the read/write throughput and I/O load distribution across all disks and for all nodes in the cluster. The activity can be visualized, used to generate alarms, or archived for later analysis. You can also monitor XVM statistics. See the following:
  - *Performance Co-Pilot for Linux User's and Administrator's Guide*
  - *Performance Co-Pilot for Linux Programmer's Guide*
  - `dkvis`(1), `pmie`(1), `pmieconf`(1), and `pmlogger`(1) man pages

Note: You must manually install the XVM statistics for the Performance Co-Pilot package; it is not installed by default.

The following sections describe the procedures for performing some of these tasks:

- "Status in Log Files" on page 342
- "Cluster, Node, and CXFS Filesystem Status" on page 343
- "I/O Fencing Status" on page 349
- "XVM Statistics" on page 351

Status in Log Files

You should monitor the following for problems:

- Server-capable administration node log: `/var/log/messages`. Look for a `Membership delivered` message to indicate that a cluster was formed.
- Events from the GUI and `clconfd`: `/var/cluster/ha/log/cad_log`
- Kernel status: `/var/cluster/ha/log/clconfd_`*hostname*
- Command line interface log:`/var/cluster/ha/log/cli_`*hostname*
- Monitoring of other daemons:`/var/cluster/ha/log/cmond_log`
- Reset daemon log: `/var/cluster/ha/log/crsd_`*hostname*
- Output of the diagnostic tools such as the serial and network connectivity tests: `/var/cluster/ha/log/diags_`*hostname*
- Cluster database membership status: `/var/cluster/ha/log/fs2d_log`
- System administration log, `/var/lib/sysadm/salog`, which contains a list of the commands run by the GUI:

For information about client-only nodes, see *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

Cluster, Node, and CXFS Filesystem Status

You can monitor system status with the following tools:

- "CXFS GUI and Status" on page 343
- "`cxfs_admin` and Status" on page 344
- "`cxfs_info` and Status" on page 346
- "`clconf_info` and Status" on page 347

Also see "Key to Icons and States" on page 165.

CXFS GUI and Status

You can monitor the status of the cluster, individual nodes, and CXFS filesystems by using the CXFS GUI connected to a server-capable administration node. For complete details about the GUI, see Chapter 10, "Reference to GUI Tasks" on page 147.

The easiest way to keep a continuous watch on the state of a cluster is to use the view area and choose the following:

Edit
  > Expand All

The cluster status can be one of the following:

- **ACTIVE**, which means the cluster is up and running.
- **INACTIVE**, which means that CXFS services have not been started.
- **ERROR**, which means that some nodes are in a **DOWN** state; that is, the cluster **should** be running, but it is not.
- **UNKNOWN**, which means that the state cannot be determined because CXFS services are not running on the node performing the query.

To query the status of a node, you provide the logical name of the node. The node status can be one of the following:

- **UP**, which means that CXFS services are started and the node is part of the CXFS kernel membership. For more information, see "CXFS Kernel Membership, Quorum, and Tiebreaker" on page 432.

- **DOWN**, which means that although CXFS services are started and the node is defined as part of the cluster, the node is not in the current CXFS kernel membership.
- **INACTIVE**, which means that CXFS services have not been started
- **UNKNOWN**, which means that the state cannot be determined because CXFS services are not running on the node performing the query.

State information is exchanged by daemons that run only when CXFS services are started. A given server-capable administration node must be running CXFS services in order to report status on other nodes.

For example, CXFS services must be started on `node1` in order for it to show the status of `node2`. If CXFS services are started on `node1`, then it will accurately report the state of all other nodes in the cluster. However, if `node1`'s CXFS services are not started, it will report the following states:

- **INACTIVE** for its own state, because it can determine that the start CXFS services task has not been run
- **UNKNOWN** as the state of all other nodes, because the daemons required to exchange information with other nodes are not running, and therefore state cannot be determined

You can use the view area to monitor the status of the nodes. Select **View: Nodes and Cluster**.

`cxfs_admin` and Status

You can monitor the status of the cluster, individual nodes, and CXFS filesystems by using the `cxfs_admin` command on any host that has `monitor` access the CXFS cluster database. For complete details about `cxfs_admin`, see Chapter 11, "Reference to `cxfs_admin` Tasks" on page 217.

To query node and cluster status, use the following `cxfs_admin` command on any host that has `monitor` access (see "Setting `cxfs_admin` Access Permissions" on page 233) to the CXFS cluster database:

```
status
```

To continuously redisplay an updated status, enter an interval in seconds:

```
status interval=seconds
```

For example, to redisplay every 8 seconds:

```
cxfs_admin:mycluster> **status interval=8**
```

To stop the updates, send an interrupt signal (usually Ctrl+C).

The most common states for nodes include:

- `Disabled`: The node is not allowed to join the cluster
- `Inactive`: The node is not in cluster membership
- `Stable`: The node is in membership and has mounted all of its filesystems

A node can have other transient states, such as `Establishing membership`.

The most common states for filesystems include:

- `Mounted`: All enabled nodes have mounted the filesystem
- `Unmounted`: All nodes have unmounted the filesystem

The cluster can have one of the following states:

- `Stable`
- `node(s) not stable`
- `filesystem(s) not stable`
- `node(s), filesystem(s) not stable`

Any other state (not mentioned above) requires attention by the administrator.

For example (a `*` character indicates a server-capable administration node):

```
cxfs_admin:cc_test2> status
Cluster    : cc_test2
Tiebreaker : minnesota
Licenses   : enterprise   allocated 27 of 256
             workstation  allocated 3 of 50
------------------  --------  ---------------------------------------------
Node                Cell ID   Status
------------------  --------  ---------------------------------------------
cc-xe *             0         Stable
cc-xe2 *            1         Stable
aiden               2         Stable
```

```
brenna              3         Stable
cc-mac1             4         Stable
cc-vista            5         Inactive
cc-win2003          6         Stable
cc-win64            7         Inactive
cc-winxp            8         Stable
cxfsibm2            9         Mounted 0 of 2 filesystems
cxfssun4            10        Stable
gaeth               11        Stable
liam                12        Stable
lorcan              13        Stable
minnesota           14        Stable
nasca               15        Stable
padraig             16        Stable

------------------  ------------------  ------------------------------------
Filesystem          Mount Point         Status
------------------  ------------------  ------------------------------------
concatfs            /mnt/concatfs       cxfsibm2 trying to mount
mirrorfs            /mnt/mirrorfs       Unmounted
stripefs            /mnt/stripefs       cxfsibm2 trying to mount

------------------  ----------  --------------------------------------------
Switch              Port Count  Known Fenced Ports
------------------  ----------  --------------------------------------------
fcswitch12          32          None
fcswitch13          32          None
```

`cxfs_info` and Status

You can monitor the status of the cluster, individual nodes, and CXFS filesystems by using the `cxfs_info` command on a client-only node.

The `cxfs_info` command provides information about the cluster status, node status, and filesystem status. `cxfs_info` is run from a client-only node. The path to `cxfs_info` varies by platform.

You can use the `-e` option to display information continuously, updating the screen when new information is available; use the `-c` option to clear the screen between updates. For less verbose output, use the `-q` (quiet) option.

For example, on a Solaris node named `cxfssun4`:

```
cxfssun4# /usr/cxfs_cluster/bin/cxfs_info
cxfs_client status [timestamp Sep 03 12:16:06 / generation 18879]

Cluster:
    sun4 (4) - enabled
Local:
    cxfssun4 (2) - enabled, state: stable, cms: up, xvm: up, fs: up
Nodes:
    cxfs27     enabled  up    1
    cxfs28     enabled  up    0
    cxfsnt4    enabled  up    3
    cxfssun4   enabled  up    2
    mesabi     enabled  DOWN  4
Filesystems:
    lun1s0     enabled  mounted          lun1s0              /lun1s0
    mirror0    disabled unmounted        mirror0             /mirror0
```

`clconf_info` and Status

You can monitor the status of the cluster, individual nodes, and CXFS filesystems by using the `clconf_info` command on a server-capable administration node, assuming that the cluster is up.

The `clconf_info` command has the following options:

`-e`	Waits for events from `clconfd` and displays the new information
`-n` *nodename*	Displays information for the specified logical node name
`-p`	Persists until the membership is formed
`-q`	(Quiet mode) Decreases verbosity of output. You can repeat this option to increase the level of quiet; that is, `-qq` specifies more quiet (less output) than `-q`.
`-s`	Sorts the output alphabetically by name for nodes and by device for filesystems. By default, the output is not sorted.

`-v`	(Verbose mode) Specifies the verbosity of output (`-vv` specifies more verbosity than `-v`). The default output for `clconf_info` is the maximum verbosity.

For example:

```
admin# /usr/cluster/bin/clconf_info

Event at [2004-04-16 09:20:59]

Membership since Fri Apr 16 09:20:56 2004
____________ ______ ________ ______ ______
Node         NodeID Status   Age    CellID
____________ ______ ________ ______ ______
leesa             0 inactive      -      0
whack             2 up           16      3
lustre            8 up            5      5
thud             88 up           16      1
cxfs2           102 DOWN          -      2
____________ ______ ________ ______ ______
2 CXFS FileSystems
/dev/cxvm/tp9500_0 on /mnt/cxfs0  enabled  server=(whack)  2 client(s)=(thud,lustre)  status=UP
/dev/cxvm/tp9500a4s0 on /mnt/tp9500a4s0  disabled  server=()  0 client(s)=()  status=DOWN
```

This command displays the following fields:

- `Node` is the node name.
- `NodeID` is the node ID.
- `Status` is the status of the node, which may be `up`, `DOWN`, or `inactive`.
- `Age` indicates how many membership transitions in which the node has participated. The age is 1 the first time a node joins the membership and will increment for each time the membership changes. This number is dynamically allocated by the CXFS software (the user does not define the age).
- `CellID` is the cell ID, which is allocated when a node is added into the cluster definition with the GUI or `cxfs_admin`. It persists until the node is removed from the cluster. The kernel also reports the cell ID in console messages.

You can also use the `clconf_info` command to monitor the status of the nodes in the cluster. It uses the same node states as the CXFS GUI. See "CXFS GUI and Status" on page 343.

For example:

```
admin# /usr/cluster/bin/clconf_info

Event at [2004-04-16 09:20:59]

Membership since Fri Apr 16 09:20:56 2004
____________ ______ ________ ______ ______
Node         NodeID Status   Age    CellID
____________ ______ ________ ______ ______
leesa             0 inactive      -      0
whack             2 up           16      3
lustre            8 up            5      5
thud             88 up           16      1
cxfs2           102 DOWN          -      2
____________ ______ ________ ______ ______
2 CXFS FileSystems
/dev/cxvm/tp9500_0 on /mnt/cxfs0  enabled  server=(whack)  2 client(s)=(thud,lustre)  status=UP
/dev/cxvm/tp9500a4s0 on /mnt/tp9500a4s0  disabled  server=()  0 client(s)=()  status=DOWN
```

I/O Fencing Status

To check the current fencing status, do one of the following:

- Select **View: Switches** in the GUI view area
- Use the `show switch` command within `cxfs_admin`
- Use the `hafence` command as follows:

  ```
  /usr/cluster/bin/hafence -q
  ```

For example, the following output shows that all nodes are enabled:

```
admin# /usr/cluster/bin/hafence -q
  Switch[0] "ptg-brocade" has 8 ports
    Port 1 type=FABRIC status=enabled  hba=210000e08b0102c6 on host thunderbox
    Port 2 type=FABRIC status=enabled  hba=210000e08b01fec5 on host whack
    Port 5 type=FABRIC status=enabled  hba=210000e08b027795 on host thump
    Port 6 type=FABRIC status=enabled  hba=210000e08b019ef0 on host thud
```

A fenced port shows `status=disabled`. For example:

```
admin# /usr/cluster/bin/hafence -q
  Switch[0] "brocade04" has 16 ports
    Port 4 type=FABRIC status=enabled  hba=210000e08b0042d8 on host o200c
    Port 5 type=FABRIC status=enabled  hba=210000e08b00908e on host cxfs30
    Port 9 type=FABRIC status=enabled  hba=2000000173002d3e on host cxfssun3
```

Verbose (`-v`) output would be as follows:

```
admin# /usr/cluster/bin/hafence -v
  Switch[0] "brocade04" has 16 ports
    Port 0 type=FABRIC status=enabled  hba=2000000173003b5f on host UNKNOWN
    Port 1 type=FABRIC status=enabled  hba=2000000173003adf on host UNKNOWN
    Port 2 type=FABRIC status=enabled  hba=210000e08b023649 on host UNKNOWN
    Port 3 type=FABRIC status=enabled  hba=210000e08b021249 on host UNKNOWN
    Port 4 type=FABRIC status=enabled  hba=210000e08b0042d8 on host o200c
    Port 5 type=FABRIC status=enabled  hba=210000e08b00908e on host cxfs30
    Port 6 type=FABRIC status=enabled  hba=2000000173002d2a on host UNKNOWN
    Port 7 type=FABRIC status=enabled  hba=2000000173003376 on host UNKNOWN
    Port 8 type=FABRIC status=enabled  hba=2000000173002c0b on host UNKNOWN
    Port 9 type=FABRIC status=enabled  hba=2000000173002d3e on host cxfssun3
    Port 10 type=FABRIC status=enabled  hba=2000000173003430 on host UNKNOWN
    Port 11 type=FABRIC status=enabled  hba=200900a0b80c13c9 on host UNKNOWN
    Port 12 type=FABRIC status=disabled hba=0000000000000000 on host UNKNOWN
    Port 13 type=FABRIC status=enabled  hba=200d00a0b80c2476 on host UNKNOWN
    Port 14 type=FABRIC status=enabled  hba=1000006069201e5b on host UNKNOWN
    Port 15 type=FABRIC status=enabled  hba=1000006069201e5b on host UNKNOWN
```

A status of `enabled` for an `UNKNOWN` host indicates that the port is connected to a system that is not a node in the cluster. A status of `disabled` for an `UNKNOWN` host indicates that the node has been fenced (disabled), and the port may or may not be connected to a node in the cluster. A status of `enabled` with a specific name host indicates that the port is not fenced and is connected to the specified node in the cluster.

To check current fail policy settings, use the `show failpolicy` command in `cxfs_admin`, the node information in the GUI, or the `cms_failconf` command as follows:

```
/usr/cluster/bin/cms_failconf -q
```

For example, the following output shows that all nodes except `thud` have the system default fail policy configuration. The node `thud` has been configured for fencing and resetting.

```
admin# /usr/cluster/bin/cms_failconf -q
CMS failure configuration:
        cell[0] whack    Reset Shutdown
        cell[1] thunder  Reset Shutdown
        cell[2] thud     Fence Reset
        cell[3] thump    Reset Shutdown
        cell[4] terry    Reset Shutdown
        cell[5] leesa    Reset Shutdown
```

XVM Statistics

You can use Performance Co-Pilot to monitor XVM statistics. To do this, you must enable the collection of statistics:

- To enable the collection of statistics for the local host, enter the following:

  ```
  pmstore xvm.control.stats_on 1
  ```

- To disable the collection of statistics for the local host, enter the following:

  ```
  pmstore xvm.control.stats_on 0
  ```

You can gather XVM statistics in the following ways:

- By using the SGI ProPack bit `pcp` RPM. It can be used to produce an ASCII report of selected metrics from the `xvm` group in the Performance Co-Pilot namespace of available metrics.

- By using the `pmgxvm` command provided.

  If you have the `pcp.sw.monitor` package, you can also use the `pmchart` command to view time-series data in the form of a moving graph. Figure 14-1 shows an example.

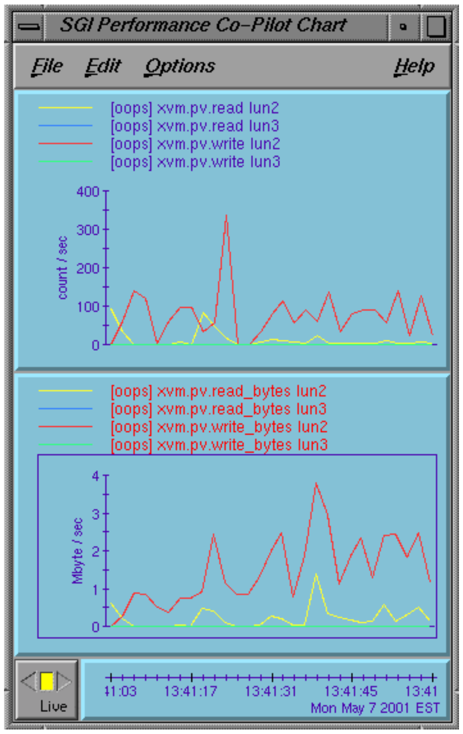


Figure 14-1 pmgxvm chart

Chapter 15

Troubleshooting

Configuring and administering a CXFS cluster can be a complex task. In general, most problems can be solved by rebooting a node. However, the topics in this chapter may help you avoid rebooting:

- "Troubleshooting Strategy" on page 353
- "Common Problems" on page 366
- "Understanding Error Messages" on page 382
- "Corrective Actions" on page 408
- "Reporting Problems to SGI" on page 416

You must connect the GUI to a node that has the `cluster_admin` software package installed. You can perform administrative tasks with `cxfs_admin` from any host with the appropriate access and network connection. See the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage* for additional troubleshooting information.

Troubleshooting Strategy

To troubleshoot CXFS problems, do the following:

- "Know the Troubleshooting Tools" on page 354
- "Identify the Cluster Status" on page 363
- "Eliminate a Residual Cluster" on page 364
- "Determine If a Node Is Fenced" on page 364
- "Locate the Problem" on page 365
- "Redirect Switch Logs" on page 365

To avoid problems in the first place, follow the recommendations in Chapter 2, "Best Practices" on page 43.

Know the Troubleshooting Tools

This section provides an **overview** of the tools required to troubleshoot CXFS:

Caution: Many of the commands listed are beyond the scope of this book and are provided here for quick reference only. See the other guides and man pages referenced for complete information before using these commands.

- "Physical Storage Tools" on page 354
- "Cluster Configuration Tools" on page 357
- "Cluster Control Tools" on page 358
- "Networking Tools" on page 359
- "Cluster/Node Status Tools" on page 359
- "Performance Monitoring Tools" on page 360
- "Kernel Status Tools" on page 361
- "Log Files" on page 362
- "Gather Cluster Configuration with `cxfsdump`" on page 362

Physical Storage Tools

Understand the following physical storage tools:

- To display the hardware inventory:

  ```
  hwinfo --short
  ```

  If the output is not what you expected, do a probe for devices and perform a SCSI bus reset, using the following commands:

  – QLogic SCSI or Fibre Channel: use the following to probe the LUN on the specified *hostname*:

```
echo "- - -" > /sys/class/scsi_host/hostname/scan
```

Each "-" character is a wildcard for bus, target, and LUN, respectively. Newer SCSI and all FC controllers have a single bus per function, but two functions in the dual-port controllers. For example, if you added a new LUN to a RAID (and the RAID is target 3) for a host named `host3`:

```
# echo "0 3 -" > /sys/class/scsi_host/host3/scan
```

QLogic Fibre Channel: use the following to discover and build a new table for the LUN, where `3` is the host number:

```
# echo "scsi-qlascan" >/proc/scsi/qla2xxx/3
```

– LSI: use the `lsiutil` tool to scan the HBA, selecting option `8` to scan for devices:

```
# lsiutil

LSI Logic MPT Configuration Utility, Version 1.41, November 23, 2005

4 MPT Ports found

     Port Name         Chip Vendor/Type/Rev    MPT Rev  Firmware Rev
 1.  /proc/mpt/ioc0    LSI Logic 53C1030 B2      102      01032710
 2.  /proc/mpt/ioc1    LSI Logic 53C1030 B2      102      01032710
 3.  /proc/mpt/ioc2    LSI Logic FC949X A1       105      01030300
 4.  /proc/mpt/ioc3    LSI Logic FC949X A1       105      01030300

Select a device:  [1-4 or 0 to quit] 3

 1.  Identify firmware, BIOS, and/or FCode
 2.  Download firmware (update the FLASH)
 4.  Download/erase BIOS and/or FCode (update the FLASH)
 8.  Scan for devices
10.  Change IOC settings (interrupt coalescing)
13.  Change FC Port settings
16.  Display logged-in devices
20.  Diagnostics
21.  RAID actions
22.  Reset bus
23.  Reset target
30.  Beacon on
31.  Beacon off
60.  Show non-default settings
61.  Restore default settings
98.  Reset FC link
```

```
99.  Reset port

Main menu, select an option:  [1-99 or e for expert or 0 to quit] 8

FC949X’s link is online, type is fabric direct attach, speed is 2 Gbaud

 B___T___L  Type       Vendor   Product          Rev         WWPN       PortId
 0 127   0  Disk       SGI      TP9300           0612  200d00a0b8131841 021500
 0 127   1  Disk       SGI      TP9300           0612
 0 127   2  Disk       SGI      TP9300           0612
 0 127  31  Disk       SGI      Universal Xport  0612
 0 128   0  Disk       SGI      TP9300           0612  200c00a0b8131841 021400
 0 128   1  Disk       SGI      TP9300           0612
 0 128   2  Disk       SGI      TP9300           0612

 0 128  31  Disk       SGI      Universal Xport  0612

 0 129   0  Disk       SGI      TP9100 F PSEUDO  5903  23000050cc007d2c 021300

 0 130   0  Disk       SGI      TP9100 F PSEUDO  5903  22000050cc007d2c 021200
            FC949X Port                                100000062b0e4248 021700
            FCP Initiator                              210000e08b1058d4 021000
            FCP Initiator                              210100e08b3058d4 021100
            FCP Initiator                              100000062b0e4249 021600
            Non-FCP                                    20fc006069c021b6 fffffc
            Non-FCP                                    2007006069c021b6 fffffe
```

You can run the `cxfs-reprobe` script look for devices and perform a SCSI bus reset if necessary. `cxfs-reprobe` will also issue an XVM probe to tell XVM that there may be new devices available:

– On server-capable administration nodes:

```
admin# /var/cluster/clconfd-scripts/cxfs-reprobe
```

– On client-only nodes:

```
client# /var/cluster/cxfs_client-scripts/cxfs-reprobe
```

- To show the physical volumes, use the `xvm` command. For example, from an SGI ProPack node:

```
propack# /sbin/xvm show -v phys/
```

The path to the `xvm` command varies by platform. For more information, see the appendix in *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*. For details about XVM, see the *XVM Volume Manager Administrator's Guide*.

Cluster Configuration Tools

Understand the following cluster configuration tools:

- To configure XVM volumes, use the `xvm` command. On a server-capable administration node:

  ```
  /sbin/xvm
  ```

  The path to the `xvm` command varies by platform. For more information, see the appendix in *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*. For details about XVM, see the *XVM Volume Manager Administrator's Guide*.

- To configure CXFS nodes and cluster, use the CXFS GUI or `cxfs_admin`:

  – The GUI:

    ```
    /usr/sbin/cxfsmgr
    ```

    See "GUI Features" on page 155 and Chapter 10, "Reference to GUI Tasks" on page 147.

  – The `cxfs_admin` command:

    ```
    /usr/cluster/bin/cxfs_admin
    ```

    See "Initial Setup with the `cxfs_admin` Command" on page 135 and Chapter 11, "Reference to `cxfs_admin` Tasks" on page 217.

- To reinitialize the database, use the `cdbreinit` command:

  ```
  /usr/cluster/bin/cdbreinit
  ```

  See "Recreating the Cluster Database" on page 413.

- To check the cluster configuration, use the following command from a server-capable administration node in the cluster:

  ```
  /usr/cluster/bin/cxfs-config -all -check
  ```

SGI recommends that you run this command after any significant configuration change or whenever problems occur. For more information, see "Validating the Cluster Configuration with `cxfs-config`" on page 335.

Cluster Control Tools

Understand the cluster control tools:

- "Cluster Administration Daemons" on page 40
- These commands are useful if you know that filesystems are available but are not indicated as such by the cluster status, or if cluster quorum is lost. However, note that `service cxfs stop` and `service cxfs stop` will cause CXFS to completely shut down on the local node.

  See the following:

  – "Ensure Cluster Database Membership Quorum Stability" on page 46

  – "Restarting CXFS Services" on page 409

  – "Clearing the Cluster Database" on page 409

  – "Stopping and Restarting Cluster Administration Daemons" on page 412
- "CXFS Services" on page 28

  Running this command on the metadata server will cause its filesystems to be recovered by another potential metadata server. See "Cluster Services Tasks with the GUI" on page 188.

  Note: Relocation and recovery are supported only when using standby nodes. Relocation is disabled by default.

- To revoke and allow CXFS kernel membership on the local node, forcing recovery on the metadata server for the local node, use the GUI or the following `cxfs_admin` command:

  `cxfs_admin:`*clustername*`>` **`disable node:`*nodename***

  Wait until recovery is complete before issuing a subsequent:

  `cxfs_admin:`*clustername*`>` **`enable node:`*nodename***

The local node cannot rejoin the CXFS kernel membership until recovery is complete.

Also see the following:

- "Revoke Membership of the Local Node with the GUI" on page 192
- "Allow Membership of the Local Node with the GUI" on page 193
- "Disable a Node with `cxfs_admin`" on page 244
- "Enable a Node with `cxfs_admin`" on page 244

Networking Tools

Understand the following networking tools:

- Send packets to network hosts using the `ping`(1) command
- Show network status using the `netstat`(1) command

Cluster/Node Status Tools

Understand the following cluster/node status tools:

- To show which cluster daemons are running:

  ```
  ps -ef | grep cluster
  ```

  See "Verify that the Cluster Daemons are Running" on page 128.

- To see cluster and filesystem status, use one of the following:

  - GUI:

    ```
    /usr/sbin/cxfsmgr
    ```

    See "Display a Cluster with the GUI" on page 188.

  - `cxfs_admin` command:

    ```
    /usr/cluster/bin/cxfs_admin -c status
    ```

    See "Display a Cluster with `cxfs_admin`" on page 252.

  - `cxfs_info` command on an client-only node (see *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*)

- To see the mounted filesystems:

  ```
  /bin/mount
  /bin/df
  ```

  You can also use the `df` command to report the number of free disk blocks

- To show volumes:

  ```
  /sbin/xvm show vol/
  ```

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

Performance Monitoring Tools

Understand the following performance monitoring tools:

- To monitor system activity:

  ```
  /usr/bin/sar
  ```

- To monitor system input/output device loading on an SGI ProPack node, use the `iostat`(1) command. For example, to monitor at 2–second intervals for 10000000 times:

  ```
  propack# iostat 2 1000000
  ```

- To monitor process status, memory consumption, paging activity, block I/O operations, interrupts, context switches, and processor usage on an SGI ProPack node, use the `vmstat`(8) command. For example, to monitor at 1–second intervals for 1000 times:

  ```
  propack# vmstat -a -n 1 1000
  ```

- To monitor the statistics for an XVM volume, use the `xvm` command:

  ```
  # /sbin/xvm change stat on {concatname|stripename|physname}
  ```

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

- To monitor system performance, use Performance Co-Pilot (PCP). See the PCP documentation and the `pmie`(1) and `pmieconf`(1) man pages.

Kernel Status Tools

Understand the following kernel status tools (this may require help from SGI service personnel):

- To determine SGI ProPack kernel status, use the KDB built-in kernel debugger.

  When kdb is enabled, a system panic will cause the debugger to be invoked and the keyboard LEDs will blink. The `kdb` prompt will display basic information. To obtain a stack trace, enter the `bt` command at the `kdb` prompt:

  ```
  kdb> bt
  ```

  To get a list of current processes, enter the following:

  ```
  kdb> ps
  ```

  To backtrace a particular process, enter the following, where *PID* is the process ID:

  ```
  kdb> btp PID
  ```

  To get a dump, enter the following:

  ```
  kdb> sr d
  ```

  To exit the debugger, enter the following:

  ```
  kdb> go
  ```

  If the system will be run in graphical mode with `kdb` enabled, SGI highly recommends that you use kdb on a serial console so that the kdb prompt can be seen.

- To invoke internal kernel routines that provide useful debugging information, use the `idbg` command:

  ```
  # /usr/sbin/idbg
  ```

- Use the appropriate version of `lcrash` and load the CXFS `kerntypes`:

```
# lcrash -x /boot/sgi-cxfs-kerntypes-kernelversion-architecturetype
```

Note: Do not use the version of `lcrash` that is shipped with SLES. Use the version of `lcrash` that is available from Supportfolio.

Log Files

Understand the log files discussed in "Status in Log Files" on page 342.

Gather Cluster Configuration with `cxfsdump`

Before reporting a problem to SGI, you should use the `cxfsdump` command to gather configuration information about the CXFS cluster, such as network interfaces, CXFS registry information, I/O, and cluster database contents. This will allow SGI support to solve the problem more quickly.

Note: In cluster mode (the default), the `cxfsdump` command requires `rsh/ssh` and `rcp/scp` access across all nodes in the cluster. You can use the `-secure` option to use secure remote connections.

You should run `cxfsdump` from a server-capable administration node in the cluster:

```
admin# /usr/cluster/bin/cxfsdump
```

The output will be placed in a file in the directory `/var/cluster/cxfsdump-data` directory on the server-capable administration node on which the `cxfsdump` command was run. The `cxfsdump` command will report the name and location of the file when it is finished.

To gather information about just the local node, use the `cxfsdump -local` option.

On Windows nodes, use the following menu selection to access the `\Program Files\CXFS\cxfsdump.exe` command:

Start
  > Programs
    > CXFS
      > CXFS Dump

You can configure the location of the dump by selecting the directory from a **browse for folder** dialog or type in the path in the edit field.

On Windows nodes, the `cxfsdump /?` command displays a help message. The `cxfsdump -help` command displays a help message on other nodes.

For more information about client-only nodes, see *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

Identify the Cluster Status

When you encounter a problem, identify the cluster status by answering the following questions:

- Are the cluster daemons running? See "Verify that the Cluster Daemons are Running" on page 128.
- Is the cluster state consistent on each node? Run the GUI or `cxfs_admin` command on each server-capable administration node and compare.
- Which nodes are in the CXFS kernel membership? Check the cluster status and the `/var/log/messages` file on server-capable administration nodes.
- Which nodes are in the cluster database (`fs2d`) membership? See the `/var/cluster/ha/log/fs2d_log` files on each server-capable administration node.
- Is the database consistent on all server-capable administration nodes? Determine this logging in to each server-capable administration node and examining the `/var/cluster/ha/log/fs2d_log` file and database checksum.
- Log onto the various CXFS client nodes or use the GUI view area display with details showing to answer the following:
  - Are the devices available on all nodes? Use the following:
    - The `xvm` command to show the physical volumes:

      ```
      xvm:cluster> show -v phys/
      ```
    - Is the client-only node in the cluster? Use the `cxfs_info` command.
    - List the contents of the `/dev/cxvm` directory with the `ls` command:

      ```
      # ls /dev/cxvm
      ```
    - Use the `hinv` command to display the hardware inventory. See "Physical Storage Tools" on page 354.
  - Are the filesystems mounted on all nodes? Use `mount`, the GUI, and the `cxfs_admin` and `clconf_info` commands.
  - Which node is the metadata server for each filesystem? Use the GUI or the `cxfs_admin` command to determine. See "Discovering the Active Metadata Server" on page 293.

- Is the metadata server in the process of recovery? Look at the following file: `/var/log/messages`

  Messages such as the following indicate that recovery status:

  – In process:

```
Mar 13 11:31:02 1A:p2 unix: ALERT: CXFS Recovery: Cell 1: Client Cell 0 Died, Recovering </scratch/p9/local>
```

  – Completed:

```
Mar 13 11:31:04 5A:p2 unix: NOTICE: Signaling end of recovery cell 1
```

- If filesystems are not mounting, do they appear online in XVM? You can use the following `xvm` command:

```
xvm:cluster> show vol/*
```

Eliminate a Residual Cluster

Before you start configuring another new cluster, make sure no nodes are still in a CXFS membership from a previous cluster. Enter the following to check for a `cmsd` kernel thread:

```
admin# ps -ef | grep cmsd
```

If the output shows a `cmsd` kernel thread, perform a forced CXFS shutdown by entering the following:

```
admin# service cxfs stop
```

Then check for a `cmsd` kernel thread again.

After waiting a few moments, if the `cmsd` kernel thread still exists, you must reboot the machine or leave it out of the new cluster definition. It will not be able to join a new cluster in this state and it may prevent the rest of the cluster from forming a new CXFS membership.

Determine If a Node Is Fenced

To determine if a node is fenced, log in to a server-capable administration node and use the `cxfs_admin status` command or the `hafence`(1M) command.

The following messages are logged when fencing changes:

```
Raising fence on cell cellID (nodename)
```

```
Lowering fence on cell cellID (nodename)
```

Locate the Problem

To locate the problem, do the following:

- Examine the log files (see "Log Files" on page 362):
  - Search for errors in all log files. See "Status in Log Files" on page 342. Examine all messages within the timeframe in question.
  - Trace errors to the source. Try to find an event that triggered the error.
- Use detailed information from the view area in the GUI to drill down to specific configuration information.
- Run the **Test Connectivity** task in the GUI. See "Test Node Connectivity with the GUI" on page 185.
- Determine how the nodes of the cluster see the current CXFS kernel membership by entering the following command on each server-capable administration node:

  ```
  /usr/cluster/bin/clconf_info
  ```

- Check the `/var/log/messages` file on each server-capable administration node to make sure the CXFS filesystems have been successfully mounted or unmounted.

  If a mount/unmount fails, the error will be logged and the operation will be retried after a short delay.

- Get a dump of the cluster database. You can extract such a dump with the following command:

  ```
  /usr/cluster/bin/cdbutil -c 'gettree #' > dumpfile
  ```

Redirect Switch Logs

Brocade switch problems can cause CXFS to behave abnormally. For easier troubleshooting, use the `syslogdipadd` function on the switch to redirect its `syslogd` information to up to six potential metadata servers in the cluster. SGI recommends logging to at least two potential metadata servers on which you troubleshoot issues and look for error messages. The `syslogd` information is the same as that given by `errshow` command on the switch.

For example, on each switch, define the metadata server nodes MDS1 and MDS2 to which the switch can redirect its `syslogd` output:

```
switch:admin > syslogdipadd ipaddress_MDS1
switch:admin > syslogdipadd ipaddress_MDS2
```

The entries from the switch can be sorted because they are prefixed by the switch name, which is standard `syslogd` behavior.

Common Problems

The following are common problems and solutions:

- "Client Membership Loss" on page 367
- "Node is Permanently Fenced" on page 369
- "Cannot Access Filesystem" on page 369
- "Log Files Consume Too Much Disk Space" on page 369
- "Unable to Define a Node" on page 370
- "System is Hung" on page 370
- "Node is Detected but Never Joins Membership" on page 370
- "Cell ID Count and `Membership delivered` Messages" on page 370
- "You Cannot Log In" on page 371
- "I/O Error in Filesystem" on page 371
- "Cannot Mount Filesystems" on page 372
- "GUI Displays Invalid Filesystems" on page 372
- "Multiple `client_timeout` Values" on page 372
- "No HBA WWPNs are Detected" on page 373
- "XFS Internal Errors in System Log File" on page 375

- "Multiple Ethernet Interfaces on Altix Systems" on page 375
- "Clients Unable to Remount Filesystems" on page 376
- "Forced Filesystem Shutdown Messages and XFS File Corruption" on page 376
- "GUI Will Not Run" on page 377
- "IPMI Issues" on page 378
- "`cxfs_admin` Output is Not Current" on page 380
- "`clconfd` Is Not Running" on page 380
- "Inappropriate Node Membership Loss Due to CXFS Kernel Heartbeat Issues" on page 381
- "Slow Access to Files" on page 382

Client Membership Loss

The following messages indicate that a client has lost membership (line breaks added here for readability):

```
Mar 15 10:55:35 5A:mvcxfs2 kernel: Error -1 reading mesg header channel 0 cell 4 (mvcxfs17)
  [priority 1 at 192.168.17.173 via 192.168.17.48]
Mar 15 10:55:35 4A:mvcxfs2 kernel: Error receiving messages from cell 4 (mvcxfs17) tcpchannel 0
  [priority 1 at 192.168.17.173 via 192.168.17.48]
Mar 15 10:55:36 5A:mvcxfs2 kernel: Error -1 reading mesg header channel 1 cell 4 (mvcxfs17)
  [priority 1 at 192.168.17.173 via 192.168.17.48]
Mar 15 10:55:36 4A:mvcxfs2 kernel: Error receiving messages from cell 4 (mvcxfs17) tcpchannel 1
  [priority 1 at 192.168.17.173 via 192.168.17.48]
Mar 15 10:55:36 5A:mvcxfs2 kernel: Error -1 reading mesg header channel 1 cell 4 (mvcxfs17)
  [priority 2 at 163.154.17.173 via 163.154.17.48]
Mar 15 10:55:36 4A:mvcxfs2 kernel: Error receiving messages from cell 4 (mvcxfs17) tcpchannel 1
  [priority 2 at 163.154.17.173 via 163.154.17.48]
Mar 15 10:55:36 4A:mvcxfs2 kernel: Transport failure cell 4
  [priority 2 at 163.154.17.173 via 163.154.17.48] 0 of 2 interfaces up
Mar 15 10:55:36 6A:mvcxfs2 kernel: Heartbeat Monitor:Failure time-stamp 295789 ticks:Last heartbeat
  time-stamp 289940 ticks:Time-stamp delta 5849 ticks (5 seconds):Heartbeat timeout 5120 ticks (5 seconds)
```

The `Error receiving` and `Error reading` messages indicate that the message channel went down. The last message, which includes the `Heartbeat Monitor` string, contains other strings that give a clues as to why the channel was disconnected. Table 15-1 on page 368 lists all of the possible strings that may be included.

Table 15-1 Error Strings

String	Description
`Aggregate Recover Transport`	Failover has forced the transport down because the remote node has detected an error on the transport.
`Aggregate Send`	An error has occurred while attempting to send a message on the underlying socket. The most likely reason is that the message channel has been disconnected by the remote end.
`Cell Up`	An error occurred while attempting to establish a connection with the remote node.
`disable heartbeat`	A configuration change has eliminated the node from the cluster or the local node is shutting down CXFS.
`Failure time-stamp`	The time-stamp in ticks of when the error was detected.
`Heartbeat Processing`	A CXFS kernel heartbeat has been received from the node that indicates it has dropped the local node from its set of known nodes.
`Heartbeat Monitor`	A CXFS kernel heartbeat timeout has been detected.
`Heartbeat timeout`	The configured timeout in ticks and in seconds for the CXFS kernel heartbeat.
`Last heartbeat time-stamp`	The time-stamp in ticks when the last CXFS kernel heartbeat from the remote node was received.
`Message Failure`	One of the following: • An internal messaging error (for example, a corrupt header has been received) . This brings down all transports connected to the remote node. This is a serious error that indicates a problem in the local node, the remote node, or the network that is causing corruption. • A socket error has occurred while attempting to send a message. The most likely reason is that the message channel has been disconnected by the remote end.

String	Description
`Receive Thread`	A socket error has occurred when attempting to receive a message. The most likely reason is that the message channel has been disconnected by the remote end.
`Time-stamp delta`	The difference in ticks and in seconds. If this delta is greater than the configured CXFS kernel heartbeat timeout, then it is definitively a heartbeat timeout.

In the above example, the last message indicates that there is a heartbeat timeout because the string `Heartbeat Monitor` is included. The message also indicates that the error was detected at 295789 ticks (`Failure time-stamp` string) and that the configured timeout is 5120 ticks or 5 seconds (the `Heartbeat timeout` string). The delta is 5849 ticks or 5 seconds (the `Time-stamp delta` string), therefore it is a heartbeat timeout because the delta is greater than the configured heartbeat timeout.

Node is Permanently Fenced

If you are unable to raise the fence on a node, it may be that the switch ports are unable to determine the WWPN. See "Hardware Changes and I/O Fencing" on page 306.

Cannot Access Filesystem

If you cannot access a filesystem, check the following:

- Is the filesystem enabled? Check the GUI and the `cxfs_admin status` command.
- Were there mount errors?

Log Files Consume Too Much Disk Space

If the log files are consuming too much disk space, you should rotate them; see "Log File Management" on page 302. You may also want to consider choosing a less-verbose log level; see the following:

- "`cad.options` on Server-Capable Administration Nodes" on page 120
- "`fs2d.options` on Server-Capable Administration Nodes" on page 121

- "Configure Log Groups with the GUI" on page 192

Unable to Define a Node

If you are unable to define a node, it may be that there are hostname resolution problems. See "Hostname Resolution and Network Configuration Rules" on page 103.

System is Hung

The following may cause the system to hang:

- Overrun disk drives.
- CXFS kernel heartbeat was lost. In this case, you will see a message that mentions `withdrawl of node`.
- As a last resort, do a non-maskable interrupt (NMI) of the system and contact SGI. (The NMI tells the kernel to panic the node so that an image of memory is saved and can be analyzed later.) For more information, see the owner's guide for the node. Make available the `/var/log/messages` system log file on server-capable administration nodes.

Node is Detected but Never Joins Membership

If a node is detected in the system log file but it never receives a `Membership delivered` message, it is likely that there is a network problem.

See "Configuring System Files" on page 119.

Cell ID Count and `Membership delivered` Messages

The `Membership delivered` messages in the system log file include a bitmask with a bit set for the cell IDs of nodes that are members of the new CXFS membership. The `Membership delivered` messages are followed by one or more messages starting with `Cell(age):` that print the individual cell IDs and the ages of their membership. `0x`*XXX* is a binary bitmask of cells included in the membership. In the following example, cell 0 has been in the last 21 CXFS memberships:

```
NOTICE: Membership delivered for cells 0x3.
Cell(age): 0(21) 1(12)
```

If the `Membership delivered` messages are appearing frequently in the system log file, it may indicate a network problem:

- Nodes that are stable and remain in the membership will have a large membership version number.
- Nodes that are having problems will be missing from the messages or have a small membership version number.

See "Configuring System Files" on page 119.

You Cannot Log In

If you cannot log in to a server-capable administration node, you can use one of the following commands, assuming the node you are on is listed in the other nodes' `.rhosts` files:

```
rsh hostname ksh -i
rsh hostname csh -i
```

I/O Error in Filesystem

The following message indicates a problem (output lines wrapped here for readability):

```
ALERT: I/O error in filesystem ("/mnt") metadata dev 0xbd block 0x41df03 ("xlog_iodone")
ALERT:     b_error 0 b_bcount 32768 b_resid 0
NOTICE: xfs_force_shutdown(/mnt,0x2) called from line 966 of file ../fs/xfs/xfs_log.c.
  Return address = 0xc0000000008626e8
ALERT: I/O Error Detected.  Shutting down filesystem: /mnt
ALERT: Please umount the filesystem, and rectify the problem(s)
```

You can fix this problem using `xfs_repair` only if there is no metadata in the XFS log. See "Forced Filesystem Shutdown Messages and XFS File Corruption" on page 376, for the appropriate procedure.

I/O errors can also appear if the node is unable to access the storage. This can happen for several reasons:

- The node has been physically disconnected from the SAN
- A filesystem shutdown due to loss of membership

- A filesystem shutdown due to lost of the metadata server
- The node has been fenced out of the SAN

Cannot Mount Filesystems

If you are unable to raise the fence on a node, it may be that the switch ports are unable to determine the WWPN. See "Hardware Changes and I/O Fencing" on page 306.

If you have defined filesystems and then rename your cluster (by deleting the old cluster and defining a new cluster), CXFS will not be able to mount the existing filesystems. This happens because the clustered XVM volume on which your CXFS filesystem resides is not accessible to the new cluster, and the volumes are therefore considered as foreign.

In order to mount the filesystem on the new cluster, you must use the XVM `steal` command to bring the clustered XVM volume into the domain of the new cluster. For more information, see the *XVM Volume Manager Administrator's Guide*.

GUI Displays Invalid Filesystems

If you create new slices on a previously sliced disk that have the same starting blocks as slices already existing on the disk, and if the old slices had filesystems, then the GUI will display those old filesystems even though they may not be valid.

Multiple `client_timeout` Values

A `client_timeout` value is set by the `clconfd` and `cxfs_client` daemons. The value depends on the order in which filesystems are mounted on the various nodes. The value adapts to help ensure that all filesystems get mounted in a timely manner. The value has no effect on the filesystem operation after it is mounted.

The value for `client_timeout` may differ among nodes, and therefore having multiple values is not really a problem.

The `retry` value is forced to be 0 and you cannot change it.

Caution: You should not attempt to change the `client_timeout` value. Improperly setting the values for client_timeout and `retry` could cause the `mount` command to keep waiting for a server and could delay the availability of the CXFS filesystems.

No HBA WWPNs are Detected

On most platforms, the `cxfs_client` software automatically detects the world wide port names (WWPNs) of any supported host bus adapters (HBAs) in the system that are connected to a switch that is configured in the cluster database. These HBAs will then be available for fencing.

However, if no WWPNs are detected, there will be messages logged to the `cxfs_client` file.

If no WWPNs are detected, you can manually specify the WWPNs in the `/etc/fencing.conf` fencing file for the SGI ProPack platform. This method does not work if the WWPNs are partially discovered.

The fencing file enumerates the worldwide port name for all of the HBAs that will be used to mount a CXFS filesystem. There must be a line for the HBA WWPN as a 64-bit hexadecimal number.

Note: The WWPN is that of the HBA itself, **not** any of the devices that are visible to that HBA in the fabric.

If used, the fencing file must contain a simple list of WWPNs, one per line.

If you use the fencing file, you must update it whenever the HBA configuration changes, including the replacement of an HBA.

Do the following:

1. Set up the switch and HBA.
2. Follow the Fibre Channel cable on the back of the node to determine the port to which it is connected in the switch. Ports are numbered beginning with 0. (For example, if there are 8 ports, they will be numbered 0 through 7.)
3. Use the `telnet` command to connect to the switch and log in as user `admin` (the password is `password` by default).

4. Execute the `switchshow` command to display the switches and their WWPN numbers.

   For example:

   ```
   brocade04:admin> switchshow
   switchName:     brocade04
   switchType:     2.4
   switchState:    Online
   switchRole:     Principal
   switchDomain:   6
   switchId:       fffc06
   switchWwn:      10:00:00:60:69:12:11:9e
   switchBeacon:   OFF
   port  0: sw  Online        F-Port  20:00:00:01:73:00:2c:0b
   port  1: cu  Online        F-Port  21:00:00:e0:8b:02:36:49
   port  2: cu  Online        F-Port  21:00:00:e0:8b:02:12:49
   port  3: sw  Online        F-Port  20:00:00:01:73:00:2d:3e
   port  4: cu  Online        F-Port  21:00:00:e0:8b:02:18:96
   port  5: cu  Online        F-Port  21:00:00:e0:8b:00:90:8e
   port  6: sw  Online        F-Port  20:00:00:01:73:00:3b:5f
   port  7: sw  Online        F-Port  20:00:00:01:73:00:33:76
   port  8: sw  Online        F-Port  21:00:00:e0:8b:01:d2:57
   port  9: sw  Online        F-Port  21:00:00:e0:8b:01:0c:57
   port 10: sw  Online        F-Port  20:08:00:a0:b8:0c:13:c9
   port 11: sw  Online        F-Port  20:0a:00:a0:b8:0c:04:5a
   port 12: sw  Online        F-Port  20:0c:00:a0:b8:0c:24:76
   port 13: sw  Online        L-Port  1 public
   port 14: sw  No_Light
   port 15: cu  Online        F-Port  21:00:00:e0:8b:00:42:d8
   ```

   The WWPN is the hexadecimal string to the right of the port number. For example, the WWPN for port 0 is `2000000173002c0b` (you must remove the colons from the WWPN reported in the `switchshow` output to produce the string to be used in the fencing file).

5. Create the `/etc/fencing.conf` fencing file and add the WWPN for the port determined in step 2. (Comment lines begin with #.)

   For dual-ported HBAs, you must include the WWPNs of any ports that are used to access cluster disks. This may result in multiple WWPNs per HBA in the file; the numbers will probably differ by a single digit.

For example, if you determined that port 0 is the port connected to the switch, your fencing file should contain the following:

```
# WWPN of the HBA installed on this system
#
2000000173002c0b
```

6. After the node is added to the cluster, enable the fencing feature by using the CXFS GUI, `hafence` `cxfs_admin` on a server-capable administration node.

XFS Internal Errors in System Log File

After a filesystem has been defined in CXFS, running `mkfs` on it (or using "Make Filesystems with the GUI" on page 199) will cause XFS internal errors to appear in the system log file. For example (line breaks added for readability):

```
Aug 17 09:25:52 1A:yokohama-mds1 unix: ALERT: Filesystem "(NULL)": XFS internal error
xfs_mount_validate_sb(4) at line 237 of file ../fs/xfs/xfs_mount.c.
Caller 0xc000000000326ef4

Aug 17 09:14:52 6X:yokohama-mds1 clconfd[360]: < E clconf 11> CI_FAILURE, fsinfo_update(/dev/cxvm/work)
kernel returned 1010 (Filesystem is corrupted)
```

To avoid these errors, run `mkfs` before defining the filesystem in CXFS, or delete the CXFS filesystem before running `mkfs`.

Multiple Ethernet Interfaces on Altix Systems

In Altix systems with multiple Ethernet interfaces, the default behavior of the operating system is to dynamically assign interface names (such as `eth0`, `eth1`, and so on) at boot time. Therefore, the physical interface associated with the `eth0` device may change after a system reboot; if this occurs, it will cause a networking problem for CXFS. To avoid this problem, provide persistent device naming by using the `/etc/sysconfig/networking/eth0_persist` file to map specific Ethernet device names to specific MAC addresses. Adding lines of the format to the `eth0_persist` file:

`eth`*N* *MAC_ID*

For example:

```
eth0 08:00:69:13:dc:ec
eth1 08:00:69:13:72:e8
```

For more information about persistent naming, see *SGI ProPack for Linux Start Here*.

Clients Unable to Remount Filesystems

If you have multiple metadata servers in the cluster but only one potential metadata server defined for a given filesystem and that server goes down, the now server-less filesystem goes into a shutdown state. Although the clients maintain membership in the cluster, they will not remount the filesystem automatically when the potential metadata server comes back up. You must manually unmount the filesystem.

If there had been only one potential metadata server in the cluster, the filesystem's clients would have lost membership and gone through a forced shutdown, which automatically unmounts the filesystems.

Forced Filesystem Shutdown Messages and XFS File Corruption

Forced filesystem shutdown messages **do not** necessarily imply that `xfs_repair` should be run. Following is an example of a message that does indicate an XFS file corruption:

```
XFS read error in file system metadata block 106412416
```

When a filesystem is forcibly shut down, the log is not empty — it contains valuable metadata. You must replay it by mounting the filesystem. The log is only empty if the filesystem is unmounted cleanly (that is, not a forced CXFS shutdown, not a crash). You can use the following command line to see an example of the transactions captured in the log file:

```
xfs_logprint -t device
```

If you run `xfs_repair` before mounting the filesystem, `xfs_repair` will delete all of this valuable metadata.

You should run `xfs_ncheck` and capture the output to a file before running `xfs_repair`. If running `xfs_repair` results in files being placed in the `lost+found` directory, the saved output from `xfs_ncheck` may help you to identify the original names of the files.

Caution: Always contact SGI technical support before using `xfs_repair` on CXFS filesystems. See"Repair Filesystems with Care" on page 67.

If you think you have a filesystem with real corruption, do the following:

1. Mount the device in order to replay the log:

   ```
   mount device  any_mount_point
   ```

2. Unmount the filesystem:

   ```
   unmount device
   ```

3. Check the filesystem:

   ```
   xfs_check device
   ```

4. View the repairs that could be made, using `xfs_repair` in no-modify mode:

   ```
   xfs_repair -n device
   ```

5. Capture filesystem file name and inode pairs:

   ```
   xfs_ncheck device > xfs_ncheck.out
   ```

6. If you are certain that the repairs are appropriate, complete them:

   ```
   xfs_repair device
   ```

GUI Will Not Run

If the GUI will not run, check the following:

- Is the license key properly installed on the server-capable administration node? See the following:
  - "Verify the License" on page 128
  - "License Key Error" on page 388
- Are the cluster daemons running? See "Verify that the Cluster Daemons are Running" on page 128.

- Are the `tcpmux` and `tcpmux/sgi_sysadm` services enabled in the `/etc/xinetd.d/tcpmux` and `/etc/tcpmux.conf` files?
- Are the `inetd` or `tcp` wrappers interfering? This may be indicated by `connection refused` or `login failed` messages.
- Are you connecting to a server-capable administration node? The `cxfsmgr` command can only be executed on a server-capable administration node. The GUI may be run from another system via the Web if you connect the GUI to a server-capable administration node.

IPMI Issues

This section discusses the following IPMI issues:

- "BMC Does Not Respond to a `ping` Command" on page 378
- "`ipmitool` Command Fails" on page 378
- "Node is Not Reset" on page 380

BMC Does Not Respond to a `ping` Command

If the baseboard management controller (BMC) does not respond to a `ping`(8) command from a remote node, verify that the BMC has a valid IP address assigned. See step 4 in "BMC System Controller" on page 451.

Note: The BMC will not respond to the `ping` command when issued from the local node (the node containing the BMC).

`ipmitool` Command Fails

If an `ipmitool`(1) command issued to a local BMC device (the node containing the BMC) fails, check the following:

- Are the IPMI modules loaded? See step 2 in "BMC System Controller" on page 451.
- Does the IPMI device exist? The default device name is `/dev/ipmi0`.
- Has the `admin` user name and password been set on the BMC with the required `ADMINISTRATOR` privileges? See step 3 in "BMC System Controller" on page 451.

- Does the BMC have a valid IP address assigned? See step 4 in "BMC System Controller" on page 451.
- Does the `ipmitool` command line contain all of the required arguments, including the OEM identifier and the device path? The basic command line used for a local node is as follows:

  ```
  ipmitool -o intelplus -d /dev/ipmi0 command
  ```

  For example:

```
[root@linux root] ipmitool -o intelplus -d /dev/ipmi0 power status
Chassis Power is on
```

  For more information, see the `ipmitool`(1) man page.

If an `ipmitool`(1) command issued to the BMC from a remote node fails, check the following:

- Does the BMC respond to the `ping`(8) command? See "BMC Does Not Respond to a `ping` Command" on page 378.
- Is the correct version of `ipmitool` installed? See step 1 in "BMC System Controller" on page 451.
- Have the `admin` user name and password been set on the BMC with the required `ADMINISTRATOR` privileges? See step 3 in "BMC System Controller" on page 451.
- Does the `ipmitool` command contain all of the required arguments, including the `lanplus` interface, the OEM identifier, and the IP address (or alias) for the BMC? The basic command line used from a remote node is as follows:

```
ipmitool -I lanplus -o intelplus -H bmc-nodename -U admin -P admin_password command
```

  For example:

```
[root@linux root] ipmitool -I lanplus -o intelplus -H my-bmc-node \
-U admin -P mypassword power status
Chassis Power is on
```

  For more information, see the `ipmitool`(1) man page.

- Does the BMC IP address (or alias) specified with the `ipmitool -H` command respond to a `ping`(8)?

- Does the BMC have address resolution protocol (ARP) and gratuitous ARP configured, with the ARP interval set to 5 seconds? (An interval of 5 seconds is supported for CXFS.) See step 4 in "BMC System Controller" on page 451.

Node is Not Reset

If a node is not properly reset by CXFS, check the following:

- Does the node's failpolicy contain `Reset` or `FenceReset`? See the following:
  - "Modify a Node Definition with the GUI" on page 181
  - "Create or Modify a Node with `cxfs_admin`" on page 235
- Does the BMC respond to a `ping`(8) command from the node defined as the `reset_node`? See "BMC Does Not Respond to a `ping` Command" on page 378.
- Does `ipmitool`(1) work correctly from the node defined as the `reset_node`? Check the system log files for relevant error messages and see the following:
  - "`ipmitool` Command Fails" on page 378
  - "BMC System Controller" on page 451

`cxfs_admin` Output is Not Current

If the `cxfs_admin` output appears to be stale (such as after you manually change the port status, in which case the CXFS database is not informed), you can update the CXFS database by running the following command:

```
# hafence -U
```

`clconfd` Is Not Running

Sending `clconfd` a `SIGTERM` signal, the default signal sent by the `kill`(1) command, will cause the `clconfd` process to terminate. When the `clconfd` process terminates on a `SIGTERM` signal, it is not restarted by `cmond` and the node will remain in the CXFS cluster membership. All filesystem activity will continue without interruption. However, if `clconfd` is not running on one or more server-capable administration nodes in the cluster, configuration changes cannot be made in the cluster and CXFS recovery may hang, preventing nodes from joining the cluster membership.

Inappropriate Node Membership Loss Due to CXFS Kernel Heartbeat Issues

If you experience problems involving a node being inappropriately taken out of the cluster membership (reset, fenced, or shut down, depending upon how the node fail policy is set), it may be that the CXFS kernel heartbeat timeout setting is inappropriate, especially if your cluster includes a large SGI Altix system (one with more than 64 processors). If you suspect this problem, you should examine the `SYSLOG` messages to determine the current kernel heartbeat timeout. Membership issues can involve either:

- Heartbeats sent from server-capable administration nodes to all other nodes in the cluster
- Heartbeats sent from client-only nodes to server-capable administration nodes (more likely to have problems)

Every kernel heartbeat multicast packet includes a counter value that monotonically increases with every heartbeat sent by the system. You can use a `snoop/tcpdump` of the private network to look at the counter and determine if the issue is a missing heartbeat or a late heartbeat. A nonconsecutive heartbeat count indicates that the heartbeat was issued by the client-only node but never received by the server-capable administration node. It generally indicates a problem somewhere in the network stack or physical network layer.

Most client-only node heartbeat timeouts are caused by late heartbeats. This is a heartbeat multicast that is not sent out by the client-only node in a timely fashion. It is a strong indicator that the client-only node is suffering from some sort of resource constraint issue. Many common resources are subject to this problem, and detailed system performance analysis is usually required to determine the exact nature of the failure. The following are common problem areas:

- Kernel locks can be held for very long periods of time and prevent a client-only node from issuing a heartbeat multicast packet in a timely fashion.
- The CPU servicing interrupts for a CXFS private network interface may be too busy with other activities to service the CXFS heartbeat kernel thread.
- Memory thrashing and the associated high-priority kernel threads attempting to relieve the memory pressure may prevent the CXFS kernel heartbeat thread from being serviced or prevent the thread from allocating the memory it requires.
- Buffered I/O uses page cache and therefore contends for the same memory resources as user applications. Applications that have high I/O requirements can

put intense pressure on system memory management functions and cause memory thrashing when dealing with a large amount of dirty page cache pages.

- A poorly configured or over-stressed CXFS private network can cause dropped heartbeat packets. This can be a problem with any node and can effect multiple client-only nodes at the same time.

To avoid these problems, see "Avoid CXFS Kernel Heartbeat Issues on Large SGI Altix Systems" on page 60.

Slow Access to Files

If file access is slow, it could be due to one of the following problems:

- Ownership changes for a LUN between RAID controllers (*LUN flipping*) can result in poor disk performance. To determine if LUN flipping is a problem, compare the contents of the `/etc/failover2.conf` file and the output of the following command (which shows the LUNs designated as `preferred`):

  ```
  # xvm show -v phys | grep preferred
  ```

  If the preferred LUNs do not also show `current path`, there is a problem with ownership changes.

- File fragmentation can also decrease performance. Run the following command on the metadata server for the file with suspected fragmentation:

  ```
  xfs_bmap -v filename
  ```

Understanding Error Messages

This section describes some of the error messages you may see. In general, the example messages are listed first by type and then in alphabetical order, starting with the message identifier or text.

Sections are as follows:

- "Normal Messages" on page 383
- "Relocation Error" on page 385
- "Controller Disable Messages" on page 386
- "CMS Error Messages" on page 386

- "`clconfd` Daemon Death" on page 386
- "Out of Logical Swap Space" on page 387
- "Lost CXFS Membership" on page 387
- "License Key Error" on page 388
- "IP Address Error" on page 389
- "System Log File Errors" on page 389
- "Log File Error Messages" on page 401
- "`cxfs_admin` Errors" on page 407
- "Mount Errors" on page 408

Normal Messages

You can expect to see the following messages. They are normal and do not indicate a problem.

`NOTICE: Error reading mesg header 4 channel 1 cell 2`

Error number 4 (`EINTR`) on `MEMBERSHIP` message channel (channel 1; channel 0 is the main channel for CXFS and XVM data) for connection with node 2. The `EINTR` indicates that this message channel is purposely being torn down and does not indicate an error in itself. (Any other error number is a real error that will cause the local node to declare the other node failed.) This is an informative message; no corrective action is required.

`NOTICE: Membership delivered for cells 0x2`

Membership has been delivered for the specified node. `0x`*XXX* is a binary bitmask of cell numbers for which membership has been delivered; `0x2` equates to cell 1.

`Cell(age):  0(4) 1(2) 2(9)`

Shows the cell and its age (the number of memberships it has been part of). One or more of these messages always follows a `Membership delivered` message.

```
NOTICE: Cell 3 (client) has joined the membership
```

The node with the specified cell ID has joined the membership. This message precedes a `Membership delivered` message if a node joined the membership.

```
NOTICE: Cell 3 (client) has left the membership
```

This message precedes a `Membership delivered` message if a node has left the membership.

```
NOTICE: Resetting cells 0x4
```

The number here is a bitmask of node numbers on which a reset is being requested. `0x`*XXX* is a binary bitmask of cells being reset. In this case, `0x4` equates to cell 2. This is an informative message; no corrective action is required.

```
CI_FAILURE, Cell 1 Machine cxfs1:  server has no information
about a machine that has reset capabilities for this machine
```

A reset mechanism was not provided for this node. The node will not be automatically reset if it fails. If you do not have reset capability, this message can be ignored. System reset configuration is recommended for all potential metadata servers.

```
NOTICE: Error reading mesg header 4 channel 1 cell 2
```

The `mesg header 4` text indicates that this is just an informative message.

```
CI_CLCONFERR_INIT in ep_name() not binding socket
```

This message appears before the daemons start.

```
clconfd[16574]:  <<CI> E clconf 0> CI_CLCONFERR_INIT, in
ep_name():  not binding socket
```

This `clconfd` message appears when daemons are starting up.

date `<I0 clconfd clconf 610:0 clconfd_client.c:84> client registration:  clconfinfo, id 9119`
date`<I0 clconfd clconf 610:0 clconfd_service.c:781> sending reply configuration and membership msg to client:  clconfinfo, id 9119`
date `<I0 clconfd clconf 610:0 clconfd_client.c:96> client un-registration:  clconfinfo, id 9119`

These messages are issued if you run the `clconf_info` command. The `clconf_info` command first registers as a CXFS client with `clconfd`; it then gets a reply message to its request for configuration and membership status; finally, it unregisters when it is done.

date `<I0 clconfd clconf 610:0 clconfd_service.c:781 sending reply configuration and membership msg to client:  cad, id 602`

This message indicates that the `cad` daemon is polling `clconfd` for status regularly. `cad` does not register and unregister each time like `clconf_info` because it is a daemon and it does not exit after each request. You will see register/unregister messages for `cad` only when `cad` or `clconfd` restarts.

`dcvn_import_force:  error 1502 from invk_dsvn_obtain_exist`

This is a normal message sent during the recovery process.

`kernel:  cxfs_cconnect_loop:  cxfs_connect_find returns error = 110`

This message will be produced if a filesystem is not successfully mounted within the designated timeout period. The mount will be retried.

Relocation Error

If you try to relocate a filesystem and see an error similar to the following `cxfs_admin` example, it means that relocation has not been enabled:

```
Error returned from server: feature not enabled (12)
Command "relocate slice1C server=server1" failed during commit: feature not enabled
```

To allow the relocation to occur, you must enable relocation as specified in "Relocation" on page 25.

Controller Disable Messages

If you see messages such as the following on the console or in a message log, it means that the Fibre Channel switch is misconfigured:

```
controller disable is not supported on loop
```

CXFS fencing recovery operations do not support loop mode. Verify that all Fibre Channel switches are configured correctly. See the switch documentation for configuration information.

CMS Error Messages

The following messages may be logged by CMS.

`CMS excluded cells` *0xXXX* `with incomplete connectivity`

Generated when CMS delivers a membership that excluded some **new** cells that had not established connections with enough cells yet to be admitted. `0x`*XXX* is a binary bitmask of excluded cells.

`CMS calculation limited to last membership:configuration change incomplete on cells` *0xXXX*

Generated when the leader is attempting to make a configuration change current (that is, actually use the change on all nodes), but some cells in the cluster have not yet gotten the configuration change staged (uploaded and ready to be made current). `0x`*XXX* is a binary bitmask of cells that do not yet have the change in their configuration. Changes make their way through the cluster asynchronously, so this situation is expected. It can take a few attempts by the CMS leader before all nodes have the change staged. As long as this situation resolves eventually, there is no problem. For more information, use `idbg cms_info`.

`CMS calculation limited to last membership:recovery incomplete`

Generated when new members were disallowed due to recovery from the last cell failure that is still being processed.

`clconfd` Daemon Death

If the `clconfd` daemon exits immediately after it starts up, it usually means that the CXFS license key has not been properly installed. Check the end of the `clconfd` log file (`/var/cluster/ha/log/clconfd_`*nodename*) for error messages. For information about licensing error messages, see "License Key Error" on page 388.

You must properly install the license keys before you can use CXFS. If you increase the number of CPUs in your system, you may need a new license key. See Chapter 5, "CXFS License Keys" on page 89.

Out of Logical Swap Space

The following example system log file message indicates an oversubscribed system:

```
ALERT: inetd [164] - out of logical swap space during fork while
allocating uarea - see swap(1M)
Availsmem 8207 availrmem 427 rlx freemem 10, real freemem 9
```

See "Use System Capacity Wisely" on page 71.

Daemons could also be leaking memory in this case. You may need to restart them:

- On server-capable administration nodes:

  ```
  admin# service cxfs_cluster restart
  ```

- On client-only nodes:

  ```
  client# killall cxfs_client
  client# service cxfs_client start
  ```

Lost CXFS Membership

The following message in the system log file indicates a kernel-triggered revocation of CXFS membership:

```
Membership lost - withdrawing from cluster
```

You must allow CXFS membership for the local node in this situation. See "Allow Membership of the Local Node with the GUI" on page 193 or "Enable a Node with `cxfs_admin`" on page 244.

License Key Error

You will see the following error if you try to install CXFS on a server-capable administration node without a valid license key already in place:

```
Preparing...                ###########################################
[100%]
   1:cxfs_cluster           ###########################################
[100%]
cxfs                      0:off  1:off  2:off  3:on   4:off  5:on   6:off
cluster_cx-exitop: Added CXFS keys to /var/cluster/cdb/cdb.db
cluster_cx-exitop: Added CXFS administration access keys to
/var/cluster/cdb/cdb.db
cxfs license check failed - use '/usr/cluster/bin/cxfslicense -d' for
details

         * * * * * * * * * * I M P O R T A N T * * * * * * * * * * * * *

        CXFS is not properly licensed for this host.  Run
                '/usr/cluster/bin/cxfslicense -d'
        for more detailed license information.
        After fixing the license, please run
                '/bin/true; /etc/init.d/cxfs_cluster restart'.

cluster_cx-exitop: success
```

If you see the following message in the `/var/cluster/ha/log/clconfd_`*nodename* logfile, it means that the CXFS license key was not properly installed:

```
CXFS not properly licensed for this host.  Run
               '/usr/cluster/bin/cxfslicense -d'
       for detailed failure information.
```

If you do not have the CXFS license key properly installed, you will see an error on the console when trying to run CXFS. For example, on an SGI ProPack node:

```
Cluster services:CXFS not properly licensed for this host.  Run
        '/usr/cluster/bin/cxfslicense -d'
for detailed failure information.  After fixing the
license, please run '/etc/init.d/cxfs_cluster restart'.
```

An error such as the following example will appear in the system log file:

```
Mar  4 12:58:05 6X:typhoon-q32 crsd[533]: <<CI> N crs 0> Crsd restarted.
Mar  4 12:58:05 6X:typhoon-q32 clconfd[537]: <<CI> N clconf 0>
Mar  4 12:58:05 5B:typhoon-q32 CLCONFD failed the CXFS license check.Use the
Mar  4 12:58:05 5B:typhoon-q32    '/usr/cluster/bin/cxfslicense -d'
Mar  4 12:58:05 5B:typhoon-q32 command to diagnose the license problem.
```

If the `clconfd` daemon dies right after it starts up, this error may be present.

You must properly install the license key before you can use CXFS. See Chapter 5, "CXFS License Keys" on page 89.

IP Address Error

If you have conflicting cluster ID numbers at your site, you will see errors such as the following:

```
WARNING: mtcp ignoring  alive message from 1 with wrong ip addr 128.162.89.34
WARNING: mtcp ignoring  alive message from 0 with wrong ip addr 128.162.89.33
```

A cluster ID number must be unique. To solve this problem, make the cluster ID numbers unique.

This error can occur if you redefine the cluster configuration and start CXFS services while some nodes have stale information from a previous configuration.

To solve the problem, first try the steps in "Eliminate a Residual Cluster" on page 364. If that does not work, reboot the nodes that have stale information. You can determine which nodes have stale information as follows: stale nodes will complain about all of the nodes, but the up-to-date nodes will complain only about the stale nodes. The `/var/cluster/ha/log/clconfd_` log file on the stale nodes will also show error messages about `SGI_CMS_CONFIG_ID` failures.

If there are too many error messages to recognize the stale nodes, reboot every node.

System Log File Errors

CXFS logs both normal operations and critical errors to the system log file, as well as to individual log files for each log group.

The system log file on server-capable administration nodes is in `/var/log/messages`.

In general, errors in the system log file file take the following form:

timestamp priority_&_facility : hostname process[ID]: <internal_info> CODE message_text

For example:

```
Sep  7 11:12:59 6X:cxfs0 cli[5830]: < E clconf 0> CI_IPCERR_NOSERVER, clconf
ipc: ipcclnt_connect() failed, file /var/cluster/ha/comm/clconfd-ipc_cxfs0
```

Table 15-2 shows the parts of the preceding message.

Table 15-2 System Log File Error Message Format

Content	Part	Meaning
`Sep 7 11:12:59`	Time Stamp	September 7 at 11:12 AM.
`6X`	Facility and level	6X indicates an informational message. See `syslogd` and the file `/usr/include/sys/syslog.h`.
`cxfs0`	Node name	The node whose logical name is `cxfs0` is the node on which the process is running.
`cli[5830]`	Process[ID]	The process sending the message is `cli` and its process ID number is `5830`.
`<CI>E clconf 0`	Internal information: message source, logging subsystem, and thread ID	The message is from the cluster infrastructure (CI). `E` indicates that it is an error. The `clconf` command is the logging subsystem. `0` indicates that it is not multithreaded.
`CI_IPCERR_NOSERVER, clconf ipc`	Internal error code	Information about the type of message; in this case, a message indicating that the server is missing. No error code is printed if it is a normal message.
`ipcclnt_connect() failed, file /var/cluster/ha/comm/clconfd-ipc_cxfs0`	Message text	A connection failed for the `clconfd-ipc_cxfs0` file.

The following sections present only the message identifiers and text.

`cli` Error Messages

For all `cli` messages, only the last message from the command (which begins with `CLI private command failed`) is meaningful. You can ignore all other `cli` messages.

The following are example errors from the `cli` daemon.

```
CI_ERR_INVAL, CLI private command:  failed (Machine (cxfs0)
exists.)
```

You tried to create a new node definition with logical name `cxfs0`; however, that node name already exists in the cluster database. Choose a different name.

```
CI_ERR_INVAL, CLI private command:  failed (IP address
(128.162.89.33) specified for control network is cxfs0 is
assigned to control network of machine (cxfs0).)
```

You specified the same IP address for two different private networks of node `cxfs0`. Use a different IP address.

```
CI_FAILURE, CLI private command:  failed (Unable to validate
hostname of machine (cxfs0) being modified.)
```

The DNS resolution of the `cxfs0` name failed. To solve this problem, add an entry for `cxfs0` in `/etc/hosts` on all nodes.

```
CI_IPCERR_NOPULSE, CLI private command:  failed (Cluster state
is UNKNOWN.)
```

The command could not complete. This is a transient error. However, if it persists, stop and restart the cluster daemons; see "Stopping and Restarting Cluster Administration Daemons" on page 412.

`clconfd` Error Messages

The following errors are sent by the `clconfd` daemon.

```
CI_CONFERR_NOTFOUND, Could not access root node.
```

The cluster database is either non-existent or corrupted, or the database daemons are not responding. Check that the database does exist.

If you get an error or the dump is empty, re-create the database; for more information, see "Clearing the Cluster Database" on page 409.

If the database exists, restart the cluster daemons; see "Stopping and Restarting Cluster Administration Daemons" on page 412.

```
CI_ERR_NOTFOUND, Could not get Cellular status for local machine
(cxfs1)
```

The database is corrupted or cannot be accessed. Same actions as above.

```
CI_FAILURE, Call to open cdb for logging configuration when it
is already open.
```

This indicates a software problem requiring you to restart the daemons; see "Stopping and Restarting Cluster Administration Daemons" on page 412.

```
CI_FAILURE, Cell 1 Machine cxfs1:  server has no information
about a machine that has reset capabilities for this machine
```

A reset mechanism was not provided for this node. The node will not be automatically reset if it fails. To ensure proper failure handling, use the CXFS GUI or `cxfs_admin` to modify the node's definition and add reset information. System reset configuration is recommended for all potential metadata servers. See "Define a Node with the GUI" on page 171, or "Create or Modify a Node with `cxfs_admin`" on page 235.

```
CI_FAILURE, CMD(/sbin/umount -k /dev/xvm/bob1):  exited with
status 1 (0x1)
```

An error occurred when trying to unmount the `/dev/xvm/bob1` filesystem. Messages from the `umount` command are usually issued just before this message and provide more information about the reason for the failure.

```
CI_FAILURE, CMD(/sbin/clmount -o 'server_list=(cxfs0,cxfs1)'
/dev/xvm/bob2 /bob2):  exited with status 1 (0x1)
```

An error occurred when trying to mount the `/dev/xvm/bob2` filesystem. Messages from the `mount` command are usually issued just before this message and provide more information about the reason of the failure.

```
CI_FAILURE, CMD(/sbin/clmount -o ’server_list=(cxfs2,cxfs0)’
/dev/xvm/stripe4 /xvm/stripe4):  exited with status 1 (0x1)
```

You have tried to mount a filesystem without first running `mkfs`. You must use `mkfs` to construct the filesystem before mounting it. For more information, see the `mkfs`(8) man page.

```
CI_FAILURE, Could not write newincarnation number to CDB, error
= 9.
```

There was a problem accessing the cluster database. Retry the operation. If the error persists, stop and restart the cluster daemons; see "Stopping and Restarting Cluster Administration Daemons" on page 412.

If the problem persists, clear the database, reboot, and re-create the database. See "Clearing the Cluster Database" on page 409.

```
CI_FAILURE, Exiting, monitoring agent should revive me.
```

The daemon requires fresh data. It will be automatically restarted.

```
CI_FAILURE, No node for client (3) of filesystem (/dev/xvm/bob1)
on (/bob1).
```

(There may be many repetitions of this message.) The filesystem appears to still be mounted on a CXFS client node that is no longer in the cluster database. If you can identify the CXFS client node that used to be in the cluster and still has the filesystem mounted, reboot that node. Otherwise, reboot the entire cluster.

```
CI_FAILURE, No node for server (-1) of filesystem
(/dev/xvm/bob1) on (/bob1).
```

(There may be many repetitions of this message.) The filesystem appears to still be mounted on a server node that is no longer in the cluster database. If you can identify the server node that used to be in the cluster and still has the filesystem mounted, reboot that node. Otherwise, reboot the entire cluster.

```
CI_ FAILURE, Node cxfs0:  SGI_CMS_HOST_ID(tcp,128.162.8 >9.33)
error 149 (Operation already in progress)
```

The kernel already had this information; you can ignore this message.

```
CI_FAILURE, Unregistered from crs.
```

The `clconfd` daemon is no longer connected to the reset daemon and will not be able to handle resets of failed nodes. There is no corrective action.

```
CI_IPCERR_NOSERVER, Crs_register failed,will retry later.
Resetting not possible yet.
```

The `clconfd` daemon cannot connect to the reset daemon. It will not be able to handle resets of failed nodes. Check the reset daemon's log file (`/var/cluster/ha/log/crsd_`) for more error messages.

```
CI_FAILURE, | > > SGI_CMS_CONFIG_ID_AUX_V2 error 22 (Invalid
argument)
CI_FAILURE, | > > clconfd_kernel_config_thread:  failed to
update kernel config - retrying in 1 | > > second(s)
```

The previous configuration change has not fully propagated across the cluster and `clconfd` keeps trying until it succeeds. Possible causes include the following:

- The `cxfs_client` daemon is hung or is no longer running on one or more client-only nodes
- The `clconfd` daemon is hung or is no longer running on one or more server-capable administration nodes
- The cluster recovery is hung
- The local node is currently trying to join the cluster
- Other membership problems

If problems continue, you could try restarting cluster services.

```
Clconfd is out of membership, will restart after notifying
clients.
```

The `clconfd` daemon does not have enough information about the current state of the cluster. It will exit and be automatically restarted with fresh data.

```
CMD(/sbin/clmount -o ’server_list=(cxfs2,cxfs0)’
/dev/xvm/stripe4 /xvm/stripe4):  /dev/xvm/stripe4:  Invalid
argument
```

You have tried to mount a filesystem without first running `mkfs`. You must use `mkfs` to construct the filesystem before mounting it. For more information, see the `mkfs`(8) man page.

```
CMD(/sbin/clmount -o ’server_list=(cxfs0,cxfs1)’ /dev/xvm/bob2
/bob2):  /dev/xvm/bob2:  Invalid argumentSep 9 14:12:43 6X:cxfs0
clconfd[345]:  < E clconf 3> CI_FAILURE, CMD(/sbin/clmount -o
’server_list=(cxfs0,cxfs1)’ /dev/xvm/bob2 /bob2):  exited with
status 1 (0x1)
```

The first message comes from the `clmount` command (the internal CXFS mount command) and explains the error (an invalid argument was issued). The second message says that the mount failed.

```
CI_FAILURE, clconfd_status_init:  Failed to open status comm
module Wed Jan 2 15:06:40.128 EXITING.
```

This error message might indicate that there is a problem with CXFS multicast. You should also examine the `fs2d` log to see if a line such as the following specifies a valid multicast address for *IPaddress*:

```
fs2d  - CIS config: multi:IPaddress:5449, delay=1s, incarnation=0x41cc5f0bcdb1c235, flags=
```

If the address is not valid, there may be a problem with DNS or other name service. In this case, you must explicitly define the `cluster_mcast` value to the normal CXFS multicast default value of `224.0.0.250` in the `/etc/hosts` file and ensure that `etc/nsswitch.conf` is configured so that `files` are read first. For more information, see "Verifying Connectivity in a Multicast Environment" on page 413.

`crsd` Error Messages

The following errors are sent by the `crsd` daemon.

`CI_ERR_NOTFOUND, No logging entries found for group crsd, no logging will take place - Database entry #global#logging#crsd not found.`

No `crsd` logging definition was found in the cluster database. This can happen if you start cluster processes without creating the database. See "Recreating the Cluster Database" on page 413.

`CI_ERR_RETRY, Could not find machine listing.`

The `crsd` daemon could not find the local node in the cluster database. You can ignore this message if the local node definition has not yet been created.

`CI_ERR_SYS:125, bind() failed.`

The `sgi-crsd` port number in the `/etc/services` file is not unique, or there is no `sgi-crsd` entry in the file. For information about adding this entry, see "`/etc/services` on Server-Capable Administration Nodes" on page 120.

`CI_FAILURE, Entry for sgi-crsd is missing in /etc/services.`

The `sgi-crsd` entry is missing from the `/etc/services` file. For information about adding this entry, see "`/etc/services` on Server-Capable Administration Nodes" on page 120.

`CI_FAILURE, Initialization failed, exiting.`

A sequence of messages will be ended with this message; see the messages prior to this one in order to determine the cause of the failure.

`CI_ERR_INTR, BMC is busy, delaying 5 seconds.  Attempt 1 of 5.`

The `crsd` daemon was unable to contact the baseboard management controller (BMC) of the system being reset. There will be 5 attempts to connect. You can ignore this message if the connection is successful upon a subsequent attempt. If the reset is not successful after all 5 attempts, see "IPMI Issues" on page 378.

```
CI_CONFERR_NOTFOUND, Error reading and storing port info.
CI_CONFERR_NOTFOUND, Initialization failed, exiting.
```

This error may indicate that the node that is responsible for resetting another node has not been defined in the cluster database. If you use the `reset` policy, you must define the owner node before starting CXFS. See "Define a Node with the GUI" on page 171 or "Create or Modify a Node with `cxfs_admin`" on page 235.

`cmond` Error Messages

The following errors are sent by the `cmond` daemon.

```
Could not register for notification.cdb_error = 7
```

An error number of `7` indicates that the cluster database was not initialized when the cluster process was started.

This may be caused if you execute the `cdbreinit` on one server-capable administration node while some other server-capable administration nodes in the pool are still running `fs2d` and already have the node listed in the database.

Do the following:

1. Execute the following command on the nodes that show the error:

   ```
   errornode# /usr/cluster/bin/cdb-init-std-nodes
   ```

   This command will recreate the missing nodes without disrupting the rest of the database.

2. If the error persists, force the daemons to restart by executing the following command on a server-capable administration node:

   ```
   admin# service cxfs_cluster restart
   ```

   Verify that `cmond` is restarted.

3. If the error persists, reinitialize the database on just the node that is having problems.

4. If the error still persists, reinitialize all nodes in the cluster.

See "Recreating the Cluster Database" on page 413.

```
Process clconfd:343 of group cluster_cx exited, status = 3.
```

The `clconfd` process exited with status 3, meaning that the process will not be restarted by `cmond`. No corrective action is needed.

```
Process crsd:1790 of group cluster_control exited, status = 127
```

The `crsd` process exited with an error (nonzero) status. Look at the corresponding daemon logs for error messages.

`cxfslicense` Error Message

The following message will be output by the `cxfslicense -d` command if you execute it before rebooting the system:

```
error reading kernel XVM cluster mirror status. Check if XVM module is
started.
```

After you reboot the system and therefore load the XVM module, this message will no longer appear when you run `cxfslicense -d`.

`fs2d` Error Messages

The following errors are sent by the `fs2d` daemon.

```
Error 9 writing CDB info attribute for node
#cluster#elaine#machines#cxfs2#Cellular#status
```

An internal error occurred when writing to the cluster database. Retry the operation. If the error persists, stop and restart the cluster daemons; see "Stopping and Restarting Cluster Administration Daemons" on page 412.

If the problem persists, clear the database, reboot, and re-create the database. See "Clearing the Cluster Database" on page 409.

```
Error 9 writing CDB string value for node
#cluster#elaine#machines#cxfs2#Cellular#status
```

An internal error occurred when writing to the cluster database. Retry the operation. If the error persists, stop and restart the cluster daemons; see "Stopping and Restarting Cluster Administration Daemons" on page 412.

If the problem persists, clear the database, reboot, and re-create the database. See "Clearing the Cluster Database" on page 409.

```
Failed to update CDB for node
#cluster#elaine#Cellular#FileSystems#fs1#FSStatus
```

An internal error occurred when writing to the cluster database. Retry the operation. If the error persists, stop and restart the cluster daemons; see "Stopping and Restarting Cluster Administration Daemons" on page 412.

If the problem persists, clear the database, reboot, and re-create the database. See "Clearing the Cluster Database" on page 409.

```
Failed to update CDB for node
#cluster#elaine#machines#cxfs2#Cellular#status
```

An internal error occurred when writing to the cluster database. Retry the operation. If the error persists, stop and restart the cluster daemons; see "Stopping and Restarting Cluster Administration Daemons" on page 412.

If the problem persists, clear the database, reboot, and re-create the database. See "Clearing the Cluster Database" on page 409.

```
Machine 101 machine_sync failed with lock_timeout error
```

The `fs2d` daemon was not able to synchronize the cluster database and the `sync` process timed out. This operation will be retried automatically by `fs2d`.

```
ALERT: CXFS Recovery:  Cell 0:  Server Cell 2 Died, Recovering
```

The server (cell 2) died and the system is now recovering a filesystem.

General Messages

```
CI_CONFERR_NOTFOUND, Logging configuration error:  could not
read cluster database /var/cluster/cdb/cdb.db, cdb error = 3.
```

The cluster database has not been initialized. See "Recreating the Cluster Database" on page 413.

```
WARNING: Error receiving messages from cell 2 tcpchannel 1
```

There has been an error on the CXFS membership channel (channel 1; channel 0 is the main message channel for CXFS and XVM data). This may be a result of tearing down the channel or may be an error of the node (node with an ID of 2 in this case). There is no corrective action.

Log File Error Messages

CXFS maintains logs for each of the CXFS daemons. For information about customizing these logs, see "Set Log Configuration with the GUI" on page 190.

Log file messages take the following form:

*daemon*_`log` *timestamp internal_process: message_text*

For example:

```
cad_log:Thu Sep  2 17:25:06.092  cclconf_poll_clconfd: clconf_poll failed with error CI_IPCERR_NOPULSE
```

Table 15-3 on page 402, shows the parts in the preceding message.

Table 15-3 Log File Error Message Format

Content	Part	Meaning
`cad_log`	Daemon identifier	The message pertains to the `cad` daemon
`Sep 2 17:25:06.092`	Time stamp and process ID	September 2 at 5:25 PM, process ID 92.
`cclconf_poll_clconfd`	Internal process information	Internal process information
`clconf_poll failed with error CI_IPCERR_NOPULSE`	Message text	The `clconfd` daemon could not be contacted to get an update on the cluster's status.

cad Messages

The following are examples of messages from `/var/cluster/ha/log/cad_log`:

```
ccacdb_cam_open:  failed to open connection to CAM
server error 4
```

Internal message that can be ignored because the `cad` operation is automatically retried.

```
ccamail_cam_open:  failed to open connection to CAM
server error 4
```

Internal message that can be ignored because the `cad` operation is automatically retried.

```
ccicdb_cam_open:  failed to open connection to CAM
server error 4
```

Internal message that can be ignored because the `cad` operation is automatically retried.

```
cclconf_cam_open:  failed to open connection to CAM
server error 4
```

Internal message that can be ignored because the `cad` operation is automatically retried.

```
cclconf_poll_clconfd:  clconf_poll failed with error
CI_IPCERR_NOCONN
```

The `clconfd` daemon is not running or is not responding to external requests. If the error persists, stop and restart the cluster daemons; see "Stopping and Restarting Cluster Administration Daemons" on page 412.

```
cclconf_poll_clconfd:  clconf_poll failed with error
CI_IPCERR_NOPULSE
```

The `clconfd` daemon could not be contacted to get an update on the cluster's status. If the error persists, stop and restart the cluster daemons; see "Stopping and Restarting Cluster Administration Daemons" on page 412.

```
cclconf_poll_clconfd:  clconf_poll failed with error
CI_CLCONFERR_LONELY
```

The `clconfd` daemon does not have enough information to provide an accurate status of the cluster. It will automatically restart with fresh data and resume its service.

```
csrm_cam_open:  failed to open connection to CAM server error 4
```

Internal message that can be ignored because the `cad` operation is automatically retried.

```
Could not execute notification cmd.  system() failed.  Error:
No child processes
```

No mail message was sent because `cad` could not fork processes. Stop and restart the cluster daemons; see "Stopping and Restarting Cluster Administration Daemons" on page 412.

```
error 3 sending event notification to client [counter:  7 info:
0x000000021010f078]"
```

GUI process exited without cleaning up. (The `counter` and `info` numbers are internal data structures.)

`cli` Messages

The following are examples of messages from `/var/cluster/ha/log/cli_`*hostname*:

`CI_CONFERR_NOTFOUND, No machines found in the CDB.`

The local node is not defined in the cluster database.

`CI_ERR_INVAL, Cluster (bob) not defined`

The cluster called `bob` is not present in the cluster database.

```
CI_ERR_INVAL, CLI private command:  failed (Cluster (bob) not
defined)
```

The cluster called `bob` is not present in the cluster database.

```
CI_IPCERR_NOPULSE, CLI private command:  failed (Cluster state
is UNKNOWN.)
```

The cluster state could not be determined. Check if the `clconfd` daemon is running.

```
CI_IPCERR_NOPULSE, ipcclnt_pulse_internal():  server failed to
pulse
```

The cluster state could not be determined. Check if the `clconfd` daemon is running.

```
CI_IPCERR_NOSERVER, clconf ipc:  ipcclnt_connect() failed, file
/var/cluster/ha/comm/clconfd-ipc_cxfs0
```

The local node (`cxfs0`) is not defined in the cluster database.

```
CI_IPCERR_NOSERVER, Connection file
/var/cluster/ha/comm/clconfd-ipc_cxfs0 not present.
```

The local node (`cxfs0`) is not defined in the cluster database.

`crsd` Errors

The following are examples of messages from `/var/cluster/ha/log/crsd_`*hostname*:

```
CI_CONFERR_INVAL, Nodeid -1 is invalid.
I_CONFERR_INVAL, Error from ci_security_init().
CI_ERR_SYS:125, bind() failed.
CI_ERR_SYS:125, Initialization failed, exiting.
CI_ERR_NOTFOUND, Nodeid does not have a value.
CI_CONFERR_INVAL, Nodeid -1 is invalid.
```

For each of these messages, either the node ID was not provided in the node definition or the cluster processes were not running in that node when node definition was created in the cluster database. This is a warning that optional information is not available when expected.

```
CI_ERR_NOTFOUND, SystemController information for node cxfs2 not
found, requests will be ignored.
```

System controller information (optional information) was not provided for node `cxfs2`. Provide system controller information for node `cxfs2` by modifying node definition. This is a warning that optional information is not available when expected. Without this information, the node will not be reset if it fails, which might prevent the cluster from properly recovering from the failure.

```
CI_ERR_NOTFOUND, SystemController information for node cxfs0 not
found, requests will be ignored.
```

The owner node specified in the node definition for the node with a node ID of 101 has not been defined. You must define the owner node.

```
CI_CRSERR_NOTFOUND, Reset request 0x10087d48 received for node
101, but its owner node does not exist.
```

The owner node specified in the node definition for the node with a node ID of 101 has not been defined. You must define the owner node. `0x10087d48` is a pointer to an internal datastructure that uniquely identifies the request while it is being handled.

`fs2d` Errors

The following are examples of messages from `/var/cluster/ha/log/fs2d_log`:

`Failed to copy global CDB to node cxfs1 (1), error 4`

There are communication problems between the local node and node `cxfs1`. Check the private networks of the two nodes.

`Communication failure send new quorum to machine cxfs2 (102) (error 6003)`

There are communication problems between the local node and node `cxfs2`. Check the private networks of the two nodes.

`Failed to copy CDB transaction to node cxfs2 (1)`

There are communication problems between the local node and node `cxfs2`. Check the private networks of the two nodes.

`Outgoing RPC to` *hostname* `:  NULL`

If you see this message, check your Remote Procedure Call (RPC) setup. For more information, see the `rpcinfo`(8) and `portmap`(8) man pages.

`fs2d - RPC machine register:  rejecting quorum from machine` *hostname* `due to that machine not responding to our poll attempts`

This message might indicate that the NIC for the private network has not been configured or has been configured incorrectly. It also might indicate that the cable has been unplugged.

`cdbreinit` Error Messages

```
Thu Jun 3 16:20:45.431 cxfsopus1.example.com cbe_fs2 - cbe_create_node: cannot create new node (RPC error = 9)
  libcdb       - cdb_create_node: error 9 creating child of node 0x60000000000135c0 with subkey "ifd1"
```

This error means that some nodes have not been created in the cluster database. Error 9 usually means that `fs2d` is has encountered an internal error while creating that node. To fix the problem, make sure that `fs2d` is not running on any server-capable administration node and rerun `cdbreinit`.

`cxfs_admin` Errors

Following are common `cxfs_admin` errors.

```
Connecting to the local CXFS server...
receiving conflicting bootstrap packets from cluster(s) - cannot identify
server to connect to
gave up trying to connect to server
FATAL: exiting on fatal error
```

The `cxfs_admin` command can see multiple clusters. Reconfigure your network so that each cluster's private network subnet is independent of the private network subnet of other clusters. If you have multiple clusters using the same public network as the backup CXFS metadata network, use the `-i` option to identify the cluster name. See "Accessing the Correct Cluster at a Multiple-Cluster Site" on page 234.

```
Connecting to the CXFS server for the "mycluster" cluster...
Error returned from server: authorization error (8)
Inappropriate privileges to connect to the CXFS server
```

The host can see the cluster, but does not have permission to connect to it. Use cxfs_admin on another node that does have permission and use the `access` command to give permission to the first node.

```
Connecting to the CXFS server for the "mycluster" cluster...
Error returned from server: permissions error (9)
Insufficient privileges to acquire the administration lock
```

The host only has monitoring privileges and no administration privileges. Use the `permission=admin` attribute with the `access` command to grant the host administration rights, or use `-r` on the `cxfs_admin` command line.

```
Connecting to the CXFS server for the "mycluster" cluster...
not receiving bootstrap packets from any cluster - cannot identify server to connect to
gave up trying to connect to server
FATAL: exiting on fatal error
```

The host is not on the CXFS metadata private network and has not been granted explicit access to the cluster. Grant the host access by using the `access` command from a server-capable administration node or another host with `admin` access to the cluster.

Mount Errors

The following error indicates that one of the LUNs in this volume is inaccessible. A GPT-labeled LUN in the volume may cause this if GPT labels are not supported on the system:

```
# /sbin/mount -t cxfs -o 'client_timeout=30s,retry=0,server_list=(server1,server2)' \
/dev/cxvm/stripe93 /mnt/stripe93
 cxfs.util get_subvol_stripe: open(/dev/rcxvm/stripe93) returned -1, errno 19 (Operation not supported by device)
 cxfs.util get_subvol_stripe: Some of the volumes needed for /dev/rcxvm/stripe93 may have a main path that
   runs through a controller to which this machine is not connected.
 cxfs.util set_xfs_args: get_subvol_stripe failed
 cxfs.util mount_main: set_xfs_args failed
```

For information about what platforms support GPT labels, see the release notes.

Corrective Actions

This section covers the following corrective actions:

- "Restarting CXFS Services" on page 409
- "Clearing the Cluster Database" on page 409
- "Rebooting" on page 410
- "Recovering a Two-Node Cluster" on page 410
- "Rebooting without Rejoining the Cluster" on page 412
- "Stopping and Restarting Cluster Administration Daemons" on page 412
- "Recreating the Cluster Database" on page 413
- "Verifying Connectivity in a Multicast Environment" on page 413
- "Performing a Power Cycle on a Node with `cmgr`" on page 415
- "Reseting a Node with `cmgr`" on page 415

Restarting CXFS Services

If CXFS services to do not restart after a reboot, it may be that the node was marked as `INACTIVE` in the cluster data base using the **Stop CXFS Services** function of the GUI, or a `disable node:`*nodename* function of `cxfs_admin`. In this case, issuing a `service cxfs_cluster start` or `service cxfs start` will not restart the services.

You must manually start CXFS services. If you use the GUI to restart the services, or enable with `cxfs_admin`, the configuration will be set so that future reboots will also restart CXFS services.

For information, see "Start CXFS Services with the GUI" on page 189 or "Enable a Node with `cxfs_admin`" on page 244.

Clearing the Cluster Database

To clear the cluster database on all of the server-capable administration nodes of the cluster, do the following, completing each step on each server-capable administration node before moving to the next step:

Caution: This procedure deletes all configuration information.

1. Enter the following on all server-capable administration nodes:

   ```
   admin# service cxfs stop
   ```

2. Enter the following on all server-capable administration nodes:

   ```
   admin# service cxfs_cluster stop
   ```

   Caution: Complete steps 1 and 2 on each node before moving to step 3 for any node.

3. Enter the following on all server-capable administration nodes:

   ```
   admin# /usr/cluster/bin/cdbreinit
   ```

   See also "Reboot Before Changing Node ID or Cluster ID" on page 72.

4. Enter the following on all server-capable administration nodes:

   ```
   admin# service cxfs_cluster start
   ```

5. Enter the following on all server-capable administration nodes:

   ```
   admin# service cxfs start
   ```

See "Eliminate a Residual Cluster" on page 364, to get rid of possible stale cluster configuration in the kernel. If needed, reboot the nodes.

Rebooting

Enter the following individually on every node to reboot the cluster (other than Windows, which uses a different reboot mechanism) :

```
node# reboot
```

For information about client-only nodes, see the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

If you want CXFS services to restart whenever the node is rebooted, use the CXFS GUI to start CXFS services or `cxfs_admin` to enable the node. For information, see "Start CXFS Services with the GUI" on page 189 and "Enable a Node with `cxfs_admin`" on page 244.

The following are situations that may require a rebooting:

- If some CXFS clients are unable to unmount a filesystem because of a busy vnode and a reset of the node does not fix the problem, you may need to reboot every node in the cluster
- If there is no recovery activity within 10 minutes, you may need to reboot the node

Recovering a Two-Node Cluster

Suppose the following:

1. You have cluster named `clusterA` that has two server-capable administration nodes and there is no CXFS tiebreaker:

   - `node1`
   - `node2`

2. `node1` goes down and will remain down for a while.
3. `node2` recovers and `clusterA` remains up.

Note: An existing cluster can drop down to 50% of the remaining server-capable administration nodes **after** the initial CXFS kernel membership is formed. For more information, see "CXFS Kernel Membership, Quorum, and Tiebreaker" on page 432.

---

4. `node2` goes down and therefore `clusterA` fails.
5. `node2` comes back up. However, `clusterA` cannot form because the initialization of a cluster requires either:
   - **More than 50%** of the server-capable administration nodes
   - 50% of the server-capable administration nodes, **one of which is the CXFS tiebreaker**

To allow `node2` to form a cluster by itself, you must do the following:

1. Set `node2` to be the CXFS tiebreaker node, using the GUI or `cxfs_admin`:
   - See "Set Tiebreaker Node with the GUI" on page 190.
   - "Create or Modify a Cluster with `cxfs_admin`" on page 249
2. Revoke the CXFS kernel membership of `node2`:
   - See "Revoke Membership of the Local Node with the GUI" on page 192.
   - See "Disable a Node with `cxfs_admin`" on page 244.
3. Allow CXFS kernel membership of `node2`:
   - See "Allow Membership of the Local Node with the GUI" on page 193.
   - "Enable a Node with `cxfs_admin`" on page 244.
4. Unset the CXFS tiebreaker node capability.

Caution: All two-server-capable administration node clusters without a tiebreaker set must have fencing or reset configured. SGI recommends reset.

See:

- "Set Tiebreaker Node with the GUI" on page 190
- "Create or Modify a Node with `cxfs_admin`" on page 235

The cluster will attempt to communicate with the `node1` because it is still configured in the cluster, even though it is down. Therefore, it may take some time for the CXFS kernel membership to form and for filesystems to mount.

Rebooting without Rejoining the Cluster

The `cxfs_cluster` argument to `chkconfig` controls the other cluster administration daemons and the replicated cluster database:

If turned off, the database daemons will not be started at the next reboot and the local copy of the database will not be updated if you make changes to the cluster configuration on the other nodes. This could cause problems later, especially if a majority of nodes are not running the database daemons.

If the cluster daemons are causing serious trouble and prevent the machine from booting, you can recover the node by booting in single-user mode, turning the argument off and booting in multiuser mode. For example, for an Altix 350:

```
ELILO boot: linux single
Uncompressing Linux... done
 ...
Skipped services in runlevel S:                  splash
Give root password for login: *****

(none)# /sbin/chkconfig grio2 off (if running GRIOv2)
(none)# /sbin/chkconfig cxfs off
(none)# /sbin/chkconfig cxfs_cluster off
(none)# reboot
```

For more information, see "`chkconfig` Arguments" on page 288.

Stopping and Restarting Cluster Administration Daemons

The commands to stop and restart cluster administration daemons depends upon the platform. See also "Restarting CXFS Services" on page 409. For general information about the daemons, see "Kernel Threads" on page 419.

To stop and restart cluster administration daemons, enter the following:

- On server-capable administration nodes:

```
admin# service cxfs_cluster stop
admin# service cxfs_cluster start
```

- On client-only nodes:

```
client# killall cxfs_client
client# service cxfs_client start
```

Note: You could also use the `restart` option to stop and start.

These commands affect the cluster administration daemons only.

Caution: When the cluster administration daemons are stopped, the node will not receive database updates and will not update the kernel configuration. This can have very unpleasant side effects. Under most circumstances, the administration daemons should remain running at all times. Use these commands only as directed.

Recreating the Cluster Database

See Chapter 13, "Cluster Database Management" on page 331.

Verifying Connectivity in a Multicast Environment

To verify general connectivity in a multicast environment, you can execute a `ping` command on the `224.0.0.1` IP address; `224.0.0.1` is the address for all systems on this subnet multicast.

To verify the CXFS kernel heartbeat, use the `224.0.0.250` IP address. (SGI uses `224.0.0.250` by default; you may need to explicitly set this if you are having problems with name resolution involving `cluster_mcast`, see below).

Note: A node is capable of responding only when the administration daemons (`fs2d`, `cmond`, `cad`, and `crsd`) or the `cxfs_client` daemon is running.

For example, to see the response for two packets sent from IP address `163.154.17.49` to the multicast address for CXFS kernel heartbeat and ignore loopback, enter the following:

```
nodeA# ping -c 2 -I 163.154.17.49 -L 224.0.0.250
PING 224.0.0.250 (224.0.0.250): 56 data bytes
64 bytes from 163.154.17.140: icmp_seq=0 ttl=64 time=1.146 ms
64 bytes from 163.154.17.55: icmp_seq=0 DUP! ttl=255 time=1.460 ms
64 bytes from 163.154.17.52: icmp_seq=0 DUP! ttl=255 time=4.607 ms
64 bytes from 163.154.17.50: icmp_seq=0 DUP! ttl=255 time=4.942 ms
64 bytes from 163.154.17.140: icmp_seq=1 ttl=64 time=2.692 ms

----224.0.0.250 PING Statistics----
2 packets transmitted, 2 packets received, +3 duplicates, 0.0% packet
loss
round-trip min/avg/max = 1.146/2.969/4.942 ms
```

The above output indicates that there is a response from the following addresses:

```
163.154.17.140
163.154.17.55
163.154.17.52
163.154.17.50
```

To override the default address, you can use the `-c` and `-m` options on client-only nodes; for more information, see the `cxfs_client` man page.

To override or explicitly set the CXFS multicast address, do the following:

1. Set `cluster_mcast` to an IP address in the range `224.0.0.0` through `239.255.255.255` in the `/etc/hosts` file and make it resolvable on all nodes in the cluster. If you want to use DNS or another name service, you should avoid using a registered number. For a list of registered numbers, see:

   http://www.iana.org/assignments/multicast-addresses

2. Verify that the `/etc/nsswitch.conf` file is configured so that local files are accessed before either NIS or DNS. That is, the `hosts` line in `/etc/nsswitch.conf` must list `files` first. For example:

   ```
   hosts:      files nis dns
   ```

   For more information, see "Adding a Private Network" on page 104.

Performing a Power Cycle on a Node with `cmgr`

When CXFS is running, you can perform a powercycle on a node with the following `cmgr` command:

```
admin powerCycle node nodename
```

This command uses the CXFS daemons to shut off power to the node and then restart it.

You can perform a powercycle on a node in a cluster even when the CXFS daemons are not running by using the `standalone` option:

```
admin powerCycle standalone node nodename
```

The above command does not go through the `crsd` daemon.

If the node has not been defined in the cluster database, you can use the following command line (line breaks added here for readability, but it should be all on one line):

```
admin powerCycle dev_name port|IP_address_or_hostname_of_device of dev_type tty|network|ipmi
   with sysctrl_type msc|mmsc|l2|l1|bmc
```

Reseting a Node with `cmgr`

When CXFS is running, you can reset a node with a system controller by using the following `cmgr` command:

```
admin reset node hostname
```

This command uses the CXFS daemons to reset the specified node.

Even when the CXFS daemons are not running, you can reset a node with a system controller by using the `standalone` option of the `admin reset` command:

```
admin reset standalone node hostname
```

If you have defined the node but have not defined system controller information for it, you could use the following `cmgr` commands to connect to the system controller or reset the node:

```
admin ping dev_name port|IP_address_or_hostname_of_device of dev_type tty|network|ipmi with sysctrl_type msc|mmsc|l2|l1|bmc

admin reset dev_name port|IP_address_or_hostname_of_device of dev_type tty|network|ipmi with sysctrl_type msc|mmsc|l2|l1|bmc
```

The above command does not go through the `crsd` daemon.

Reporting Problems to SGI

When reporting a problem about a server-capable administration node to SGI, you should retain the following information::

- The kernel you are running:

  ```
  uname -a
  ```

- The CXFS packages you are running:

```
rpm -q cxfs_client sgi-cxfs-kmp cxfs_utils cxfs-xvm-cmds
```

- The number and types of processors in your machine:

  ```
  cat /proc/cpuinfo
  ```

- The hardware installed on your machine:

  ```
   /sbin/lspci
  ```

- Modules that are loaded on your machine:

  ```
  /sbin/lsmod
  ```

- The `/var/log/cxfs_client` log file
- Any messages that appeared in the system logs immediately before the system exhibited the problem.
- Output about the cluster obtained from the `cxfsdump` utility run on a server-capable administration node.
- The `fs2d` log files if you are experiencing cluster database membership quorum problems.
- After a system kernel panic, the debugger information from the KDB built-in kernel debugger. See "Kernel Status Tools" on page 361.
- Fibre Channel HBA World Wide name mapping:

  ```
  cat /sys/class/fc_transport/bus_ID/node_name
  ```

For example:

```
cat /sys/class/fc_transport/11:0:0:0/node_name
```

The *bus_ID* value is the output of `hwinfo --disk` in the `SysFS BusID` field.

- Output from the following commands:
  - Information from the following files:

    ```
    /var/log/messages
    /var/log/cxfs_client      (for client-only nodes)
    /var/cluster/ha/log/*     (for server-capable administration nodes)
    /etc/failover2.conf       (for XVM failover version 2)
    /etc/hosts
    /proc/discontig
    ```

  - Output from the following commands:

    ```
    /usr/cluster/bin/cdbutil gettree '#'
    /usr/bin/hinv
    /usr/bin/topology
    /sbin/xvm show -v phys
    /sbin/xvm show -top -v vol
    /bin/netstat -ia
    ```

- When a CXFS daemon or command aborts and creates core files, provide the core files and the following associated information:
  - The application that created the core file:

    `file` *core_filename*

  - The binaries listed by the following command:

    `ldd` *application_path*

CXFS Software Architecture

This appendix discusses the following for administration nodes:

- "Kernel Threads"
- "Communication Paths" on page 421
- "Flow of Metadata for Reads and Writes" on page 425

Also see the following:.

- "Cluster Administration Daemons" on page 40
- "CXFS Control Daemons" on page 41
- *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*

Kernel Threads

Table A-1 on page 420, discusses kernel threads. CXFS shares with XFS the Linux `xfsbufd` and SGI ProPack `xfsdatad` kernel threads to push buffered writes to disk.

Note: The thread names begin with a `*` (such as `[*mtcp_notify]`).

Table A-1 Kernel Threads

Kernel Thread	Software Product	Description
`cmsd`	`sgi-cxfs-server-kmp`	Manages CXFS kernel membership and CXFS kernel heartbeating.
`Recovery`	`sgi-cxfs-server-kmp`	Manages recovery protocol for node.
`corpse leader`	`sgi-cxfs-server-kmp`	Coordinates recovery between nodes.
`dcshake`	`sgi-cxfs-server-kmp`	Purges idle CXFS vnodes on the CXFS client.
`cxfsd`	`sgi-cxfs-server-kmp`	Manages sending extent and size updates from the client to the server. This daemon (which runs on the CXFS client) takes modified inodes on the client and ships back any size and unwritten extent changes to the server.
`mtcp_recv`	`sgi-cxfs-server-kmp`	Reads messages (one per open message channel).
`mtcp_notify`	`sgi-cxfs-server-kmp`	Accepts new connections.
`mtcp_discovery`	`sgi-cxfs-server-kmp`	Monitors and discovers other nodes.
`mtcp_xmit`	`sgi-cxfs-server-kmp`	Supplies CXFS kernel heartbeat.

The `fs2d`, `clconfd`, and `crsd` daemons run at real-time priority. However, the `mount` and `umount` commands and scripts executed by `clconfd` are run at normal, time-shared priority.

Communication Paths

The following figures show communication paths in CXFS.

Note: The following figures do not represent the `cmond` cluster manager daemon. The purpose of this daemon is to keep the other daemons running.

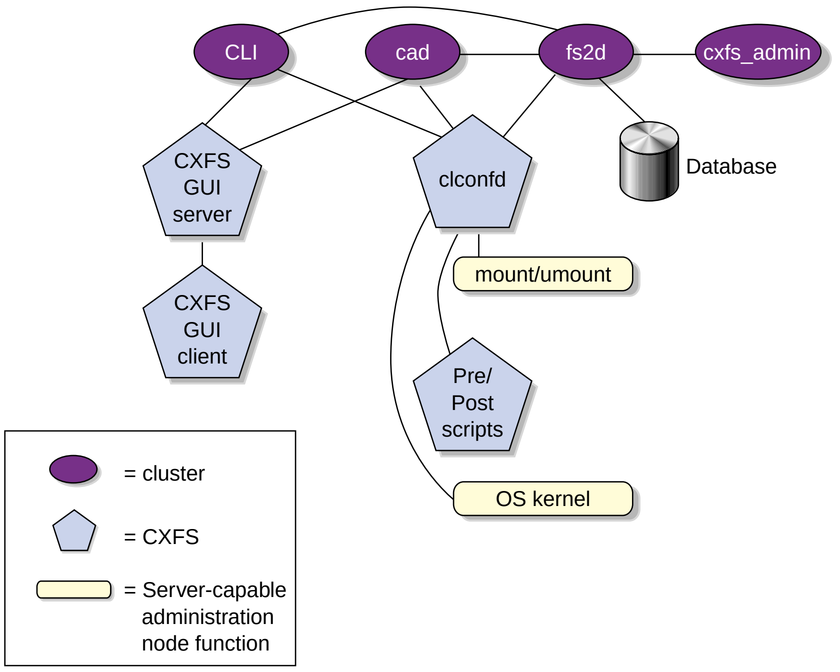


Figure A-1 Communication within One Server-Capable Administration Node

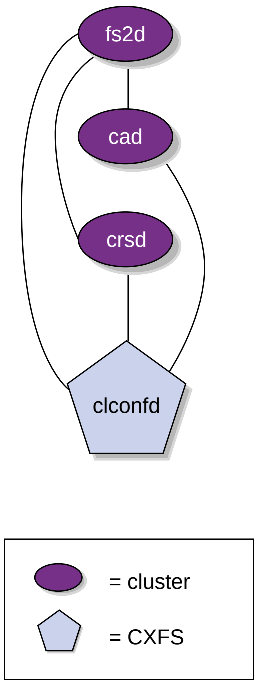


Figure A-2 Daemon Communication within One Server-Capable Administration Node

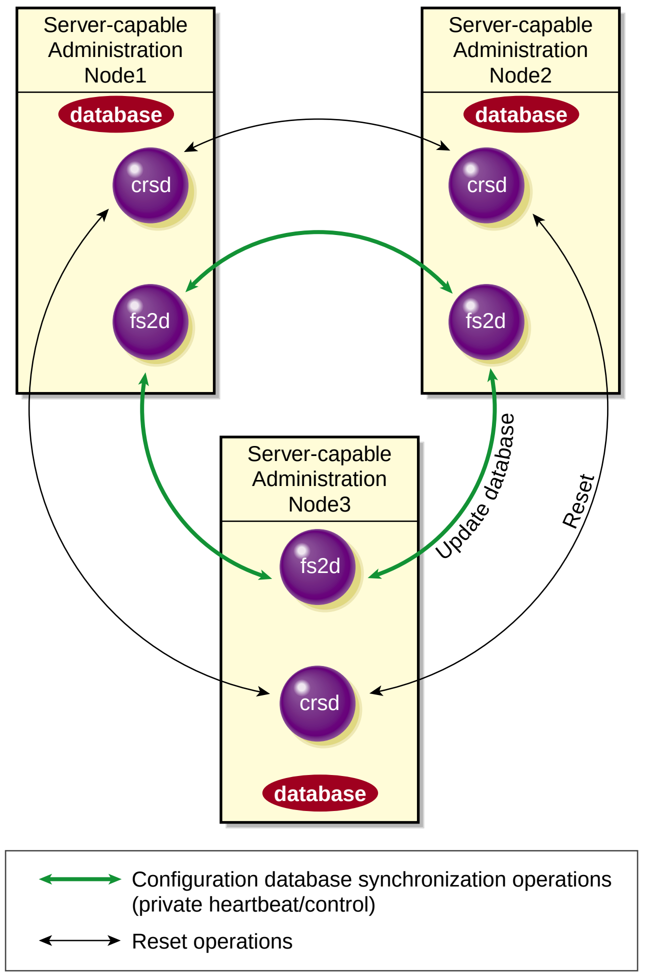

Server-capable Administration Node1
database
crsd
fs2d
Server-capable Administration Node2
database
crsd
fs2d
Server-capable Administration Node3
fs2d
crsd
database
Update database
Reset
Configuration database synchronization operations (private heartbeat/control)
Reset operations

Figure A-3 Communication between Nodes in the Pool

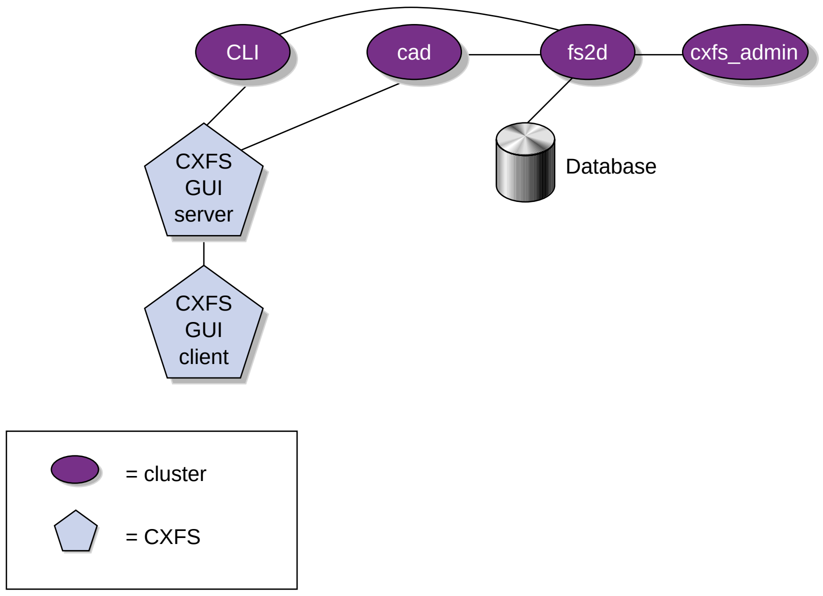


Figure A-4 Communication for a Server-Capable Administration Node Not in a Cluster

One of the server-capable administration nodes running the `fs2d` daemon is chosen to periodically multicasts its IP address and the generation number of the cluster database to each of the client-only nodes. Each time the database is changed, a new generation number is formed and multicast. The following figure describes the communication among nodes, using a Solaris client-only node as an example.

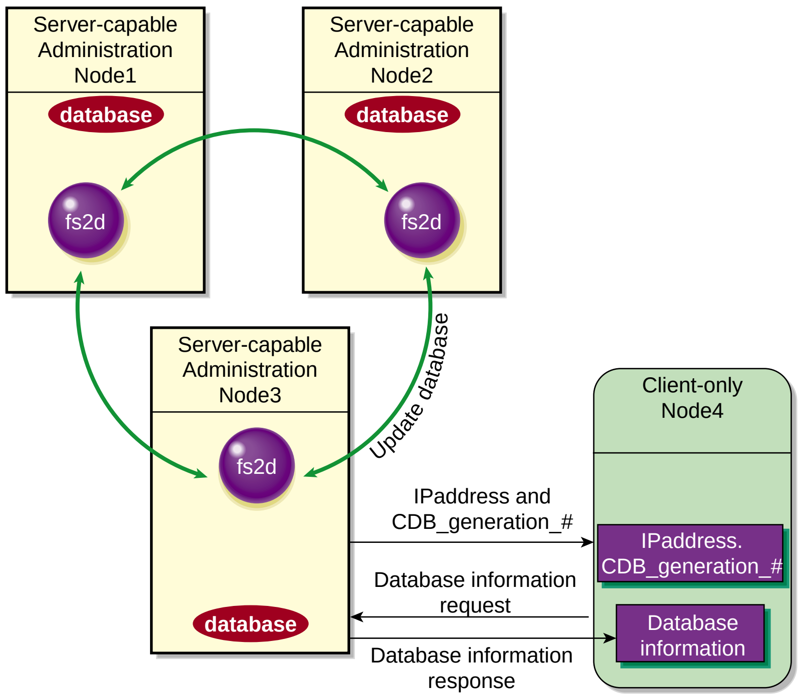

Server-capable Administration Node1
database
fs2d
Server-capable Administration Node2
database
fs2d
Server-capable Administration Node3
fs2d
database
Update database
Client-only Node4
IPaddress and CDB_generation_#
IPaddress. CDB_generation_#
Database information request
Database information
Database information response

Figure A-5 Communication Among Nodes

Flow of Metadata for Reads and Writes

The following figures show examples of metadata flow.

Note: A token protects a file. There can be multiple read tokens for a file at any given time, but only one write token.

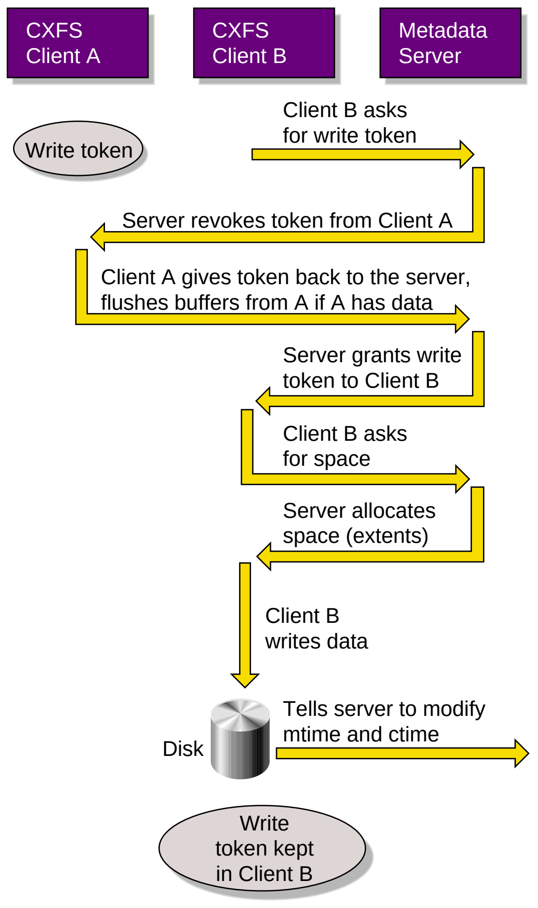

CXFS Client A
CXFS Client B
Metadata Server
Write token
Client B asks for write token
Server revokes token from Client A
Client A gives token back to the server, flushes buffers from A if A has data
Server grants write token to Client B
Client B asks for space
Server allocates space (extents)
Client B writes data
Disk
Tells server to modify mtime and ctime
Write token kept in Client B

Figure A-6 Metadata Flow on a Write

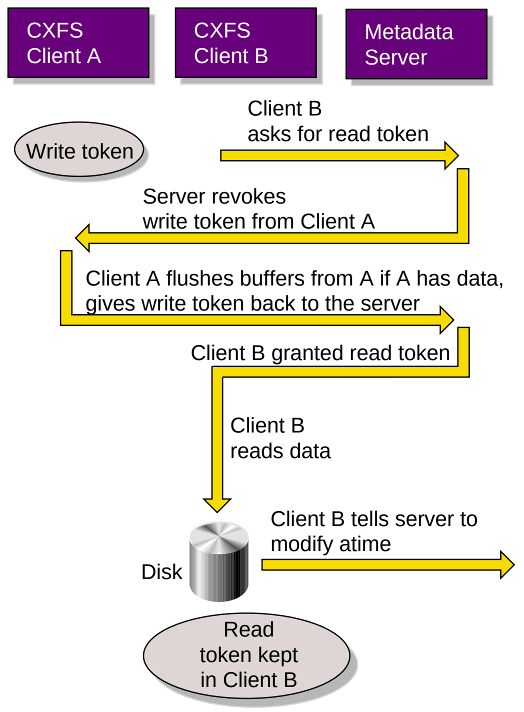

CXFS Client A
CXFS Client B
Metadata Server
Write token
Client B asks for read token
Server revokes write token from Client A
Client A flushes buffers from A if A has data, gives write token back to the server
Client B granted read token
Client B reads data
Client B tells server to modify atime
Disk
Read token kept in Client B

Figure A-7 Metadata Flow on a Read on Client B Following a Write on Client A

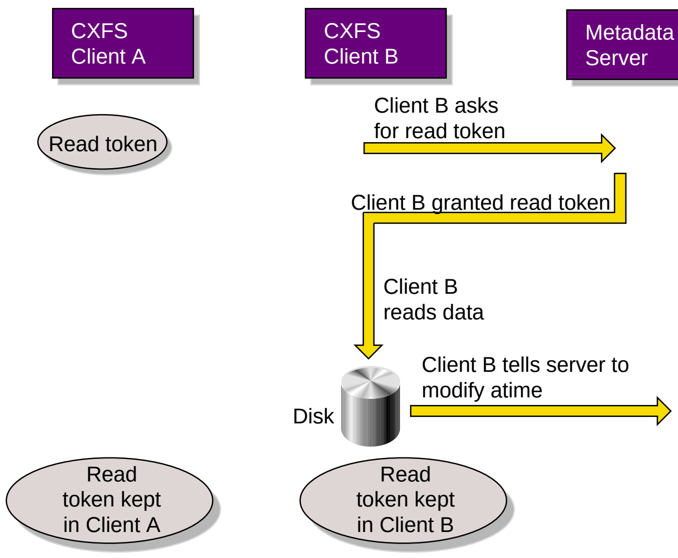

CXFS Client A
CXFS Client B
Metadata Server
Read token
Client B asks for read token
Client B granted read token
Client B reads data
Client B tells server to modify atime
Disk
Read token kept in Client A
Read token kept in Client B

Figure A-8 Metadata Flow on a Read on Client B Following a Read on Client A

Appendix B

Memberships and Quorums

The nodes in a CXFS cluster must act together to provide a service. To act in a coordinated fashion, each node must know about all the other nodes currently active and providing the service. The set of nodes that are currently working together to provide a service is called a *membership*. Cluster activity is coordinated by a configuration database that is replicated or at least accessible on all nodes in the cluster. The software sends CXFS kernel heartbeat messages between the nodes to indicate that a node is up and running. CXFS kernel heartbeat messages and cluster database heartbeat messages are exchanged via a private network so that each node can verify each membership.

Nodes within the cluster must have the correct memberships in order to provide services. This appendix discusses the different types of membership and the effect they have on the operation of your cluster.

Nodes might not be able to communicate for reasons such as the following:

- They are down
- The communication daemons have failed or have been turned off
- Software has not been configured, or has been misconfigured
- The network is misconfigured (in this case, some CXFS kernel heartbeat or cluster database heartbeat messages may fail while others succeed)
- The network router or cable fails (in this case, all CXFS kernel heartbeat and cluster database heartbeat messages will fail)

Nodes that cannot communicate must be excluded from the membership because the other nodes will not be able to verify their status.

It is critical that only one membership of each type exist at any one time, as confusion and corruption will result if two sets of nodes operate simultaneously but independently. There is a risk of this happening whenever a segmentation of the private network occurs, or any other network problem occurs that causes the nodes eligible for membership to be divided into two or more sets, where the nodes in each set can communicate with themselves, but not with nodes outside of the set. Thus, in order to form a membership, the nodes must have a quorum, the minimum number of nodes required to form a membership. The quorum is typically set at half the total eligible members.

For example, consider the case of six nodes eligible for a membership:

- If all six nodes can communicate with each other, they will form a membership of six and begin offering the membership's services.
- If a network segmentation occurs that causes four nodes to be in one set and two in another set, the two-node set will try to form its own membership but will be unable to do so because it does not have enough nodes to form a quorum; these nodes will therefore stop offering services. The four-node set will be able to form a new membership of four nodes and will continue to offer the membership's services.
- If a network segmentation occurs that divides the nodes into three sets of two nodes each, no set will be able to form a membership because none contains enough nodes to form a quorum. In this case, the membership services will be unavailable; this situation is unavoidable, as each set of two nodes thinks that the four other nodes may have formed a quorum, and so no set may safely offer the membership's services.
- If a network segmentation occurs that divides the nodes into two sets of three, then both could have a quorum, which could cause problems. To prevent this situation from occurring, some memberships may require a majority (>50%) of nodes or a tiebreaker node to form or maintain a membership. Tiebreaker nodes are used when exactly half of the server-capable administration nodes can communicate with each other.

The following sections provide more information about the specific requirements for membership.

Note: Because the nodes are unable to distinguish between a network segmentation and the failure of one or more nodes, the quorum must always be met, regardless of whether a partition has actually occurred or not.

Membership Types

There are the following types of membership:

- "Cluster Database Membership and Quorum" on page 431
- "CXFS Kernel Membership, Quorum, and Tiebreaker" on page 432

Each provides a different service using a different type of heartbeat. Nodes are usually part of more than one membership.

Cluster Database Membership and Quorum

The nodes that are part of the the cluster database membership (also known as `fs2d` *membership*) work together to coordinate configuration changes to the cluster database:

- The *potential* cluster database membership is all of the server-capable administration nodes (installed with `cluster_admin` and running `fs2d`) that are defined using the GUI or `cxfs_admin` as nodes. (CXFS client-only nodes are not eligible for cluster database membership.)
- The *actual* membership is the subset of eligible nodes that are up and running and accessible to each other, as determined by cluster database heartbeats on the private network. If the primary private network is unavailable, the cluster database heartbeat will failover to the next available heartbeat network defined for the node, if any.

The cluster database heartbeat messages use remote procedure calls (RPCs). Heartbeats are performed among all nodes in the pool. You cannot change the heartbeat timeout or interval.

If a node loses its cluster database membership, the cluster database write-operations from the node will fail; therefore, CXFS configuration changes cannot be made from that node.

The *cluster database membership quorum* ensures atomic write-operations to the cluster database that `fs2d` replicates in all server-capable administration nodes in the pool.

The cluster database membership quorum allows an initial membership to be formed when at least half (>=50%) of the eligible members are present. If there is a difference in the membership log between members, the cluster database tiebreaker node is used to determine which database is replicated. (See "Cluster Database Membership Logs" on page 433.) The cluster database tiebreaker node is always the server-capable administration node in the membership with the lowest node ID; you cannot reconfigure the tiebreaker for cluster database membership.

When the quorum is lost, the cluster database cannot be updated. This means that CXFS configuration changes cannot be made; although CXFS may continue to run, the loss of the cluster database quorum usually results in the loss of quorum for CXFS, because the nodes that drop from the cluster database membership will probably also drop from other memberships.

CXFS Kernel Membership, Quorum, and Tiebreaker

The nodes that are part of the CXFS kernel membership can share CXFS filesystems:

- The *potential* CXFS kernel membership is the group of all CXFS nodes defined in the cluster and on which CXFS services have been enabled. Nodes are enabled when CXFS services are started. The enabled status is stored in the cluster database; if an enabled node goes down, its status will remain enabled to indicate that it is supposed to be in the membership.
- The *actual* membership consists of the eligible nodes on which CXFS services have been enabled and that are communicating with other nodes using the CXFS private network. If the primary private network is unavailable, the cluster database heartbeat will failover to the next available heartbeat network defined for the node, if any.

  Note: CXFS metadata also uses the private network. The multiple heartbeats on the private network therefore reduce the bandwidth available for CXFS metadata.

  During the boot process, a CXFS node applies for CXFS kernel membership. Once accepted, the node can actively share the filesystems in the cluster.

The CXFS kernel heartbeat uses multicast. Heartbeats are performed among all CXFS-enabled nodes in the cluster.

If a node loses its CXFS kernel membership, it can no longer share CXFS filesystems.

The *CXFS kernel membership quorum* ensures that only one metadata server is writing the metadata portion of the CXFS filesystem over the storage area network:

- For the *initial* CXFS kernel membership quorum, a majority (>50%) of the server-capable administration nodes with CXFS services enabled must be available to form a membership. (*Server-capable administration* nodes are those that are installed with the `cluster_admin` product and are also defined with the CXFS GUI or `cxfs_admin` as capable of serving metadata.)

  Note: Client-only nodes can be part of the CXFS kernel membership, but they are not considered when forming a CXFS kernel membership quorum. Only server-capable administration nodes are counted when forming the quorum.

- To *maintain* the existing CXFS kernel membership quorum requires at least half (50%) of the server-capable administration nodes that are eligible for membership. If CXFS kernel quorum is lost, the shared CXFS filesystems are no longer available.

No matter what the cluster components are, SGI recommends a system reset configuration on potential metadata servers to protect data integrity and improve server reliability. I/O fencing (or system reset when available) must be used on client-only nodes. See "Use a Client-Only Tiebreaker" on page 49. In clusters with an even number of potential metadata servers, a tiebreaker should be used in addition to I/O fencing or system reset; see "Data Integrity Protection" on page 24.

You can set the CXFS tiebreaker node by using the GUI's **Set Tiebreaker Node** task or by using the `modify` command in `cxfs_admin`. See "Set Tiebreaker Node with the GUI" on page 190 and "Create a Tiebreaker with `cxfs_admin`" on page 250.

Note: If one of the server-capable administration nodes is the CXFS tiebreaker in a two server-capable cluster, failure of that node or stopping the CXFS services on that node will result in a cluster-wide forced shutdown. SGI recommends making a client-only node the tiebreaker to avoid losing the cluster if the tiebreaker node fails.

If I/O fencing or reset is used, the quorum is maintained by whichever side wins the reset/fence race.

If a tiebreaker node is set and the network being used for CXFS kernel heartbeat/control is divided in half, only the group that has the CXFS tiebreaker node will remain in the CXFS kernel membership. Nodes on any portion of the private network that are not in the group with the tiebreaker node will exit from the membership. Therefore, if the private network is cut in half, you will not have an active metadata server on each half of the private network trying to access the same CXFS metadata over the storage area network at the same time.

Cluster Database Membership Logs

Each `fs2d` daemon keeps a *membership log* that contains a history of each database change (write transaction), along with a list of nodes that were part of the membership when the write transaction was performed. All nodes that are part of the cluster database membership will have identical membership logs.

When a node is defined in the database, it must obtain a current copy of the cluster database and the membership log from a node that is already in the cluster database membership. The method used to choose which node's database is replicated follows a hierarchy:

1. If the membership logs in the pool share a common transaction history, but one log does not have the most recent transactions and is therefore incomplete, the database from a node that has the complete log will be chosen to be replicated.
2. If there are two different sets of membership logs, the database from the set with the most number of nodes will be chosen.
3. If there are two different sets of membership logs, and each set has an equal number of nodes, then the set containing the node with the lowest node ID will be chosen.

To ensure that the complete transaction history is maintained, do not make configuration changes on two different server-capable administration nodes in the pool simultaneously. You should connect the CXFS GUI to a single server-capable administration node in the pool when making changes. However, you can use any node in the pool when requesting status or configuration information.

The following figures describe potential scenarios using the hierarchies.

Figure B-1 on page 435, shows:

- Time 1: An established pool of three server-capable administration nodes sharing cluster database heartbeats, with node IDs 1-3, represented by the node names N1-N3. The `fs2d` database tiebreaker node is the node in the membership with the lowest node ID. Each successive database write is identified by a letter in the membership log.
- Time 2: A new node, N4, is defined using `cxfs_admin` or the GUI connected to node N1. Node N4 (node ID = 4) joins the pool. Its membership log is empty.
- Time 3: Because N1/N2/N3 have identical membership logs, the database is replicated from one of them. In this case, N2 is randomly chosen.
- Time 4: All nodes in the pool have identical membership logs.
- Time 5: A network partition occurs that isolates N1. Therefore, N1 can no longer receive database updates. Configuration changes are made by connecting the GUI to N2 (or running `cxfs_admin` on node N2); this results in updates to the membership logs in N2, N3, and N4, but not to N1 because it is isolated.

- Time 6: The partition is resolved and N1 is no longer isolated. Because N2/N3/N4 have identical membership logs, and share the beginning history with N1, the database is replicated from one of them. N4 is chosen at random.
- Time 7: All nodes in the pool have identical membership logs.

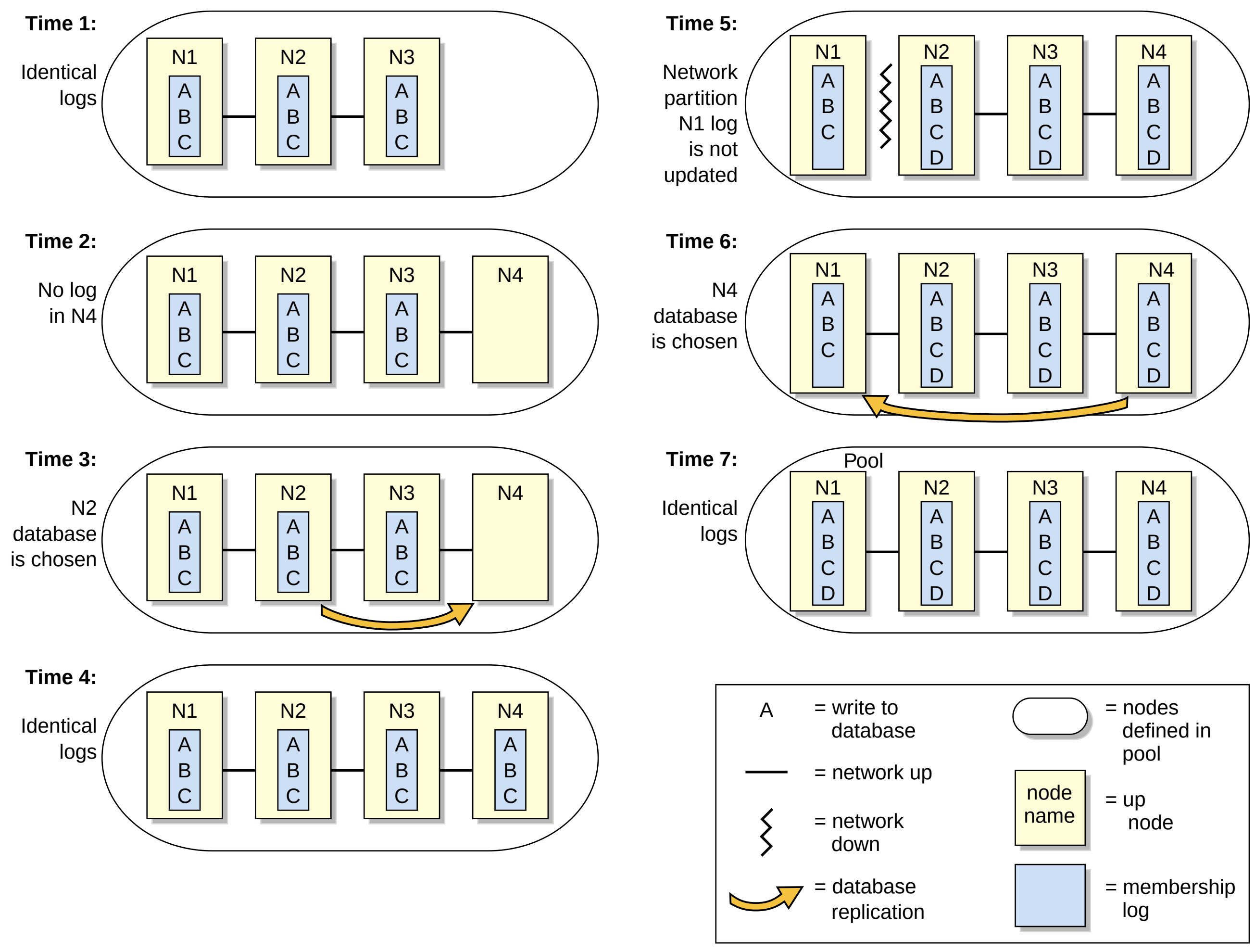


Figure B-1 One Node is Out of Date: Most Recent Log is Replicated

Recall that a node can be in only one pool at a time. If there are two separate pools, and from a node in one pool you define one or more nodes that are already in the other pool, the result will be that nodes from one of the pools will move into the other pool. **This operation is not recommended**, and determining which nodes will move into which other pool can be difficult. Figure B-2 on page 437 illustrates what to expect in this situation.

- Time 1: There are two pools that do not share membership log contents. One pool has two nodes (N1/N2), the other has three (N3/N4/N5).
- Time 2: N1 and N2 are defined as part of the second pool by running `cxfs_admin` or connecting the GUI to node N3, N4, or N5. This results in a new pool with five nodes with different membership logs.
- Time 3: The database from the larger set of nodes is the one that must be replicated. N3 is chosen at random from the N3/N4/N5 set.
- Time 4: All nodes in the pool have identical membership logs.

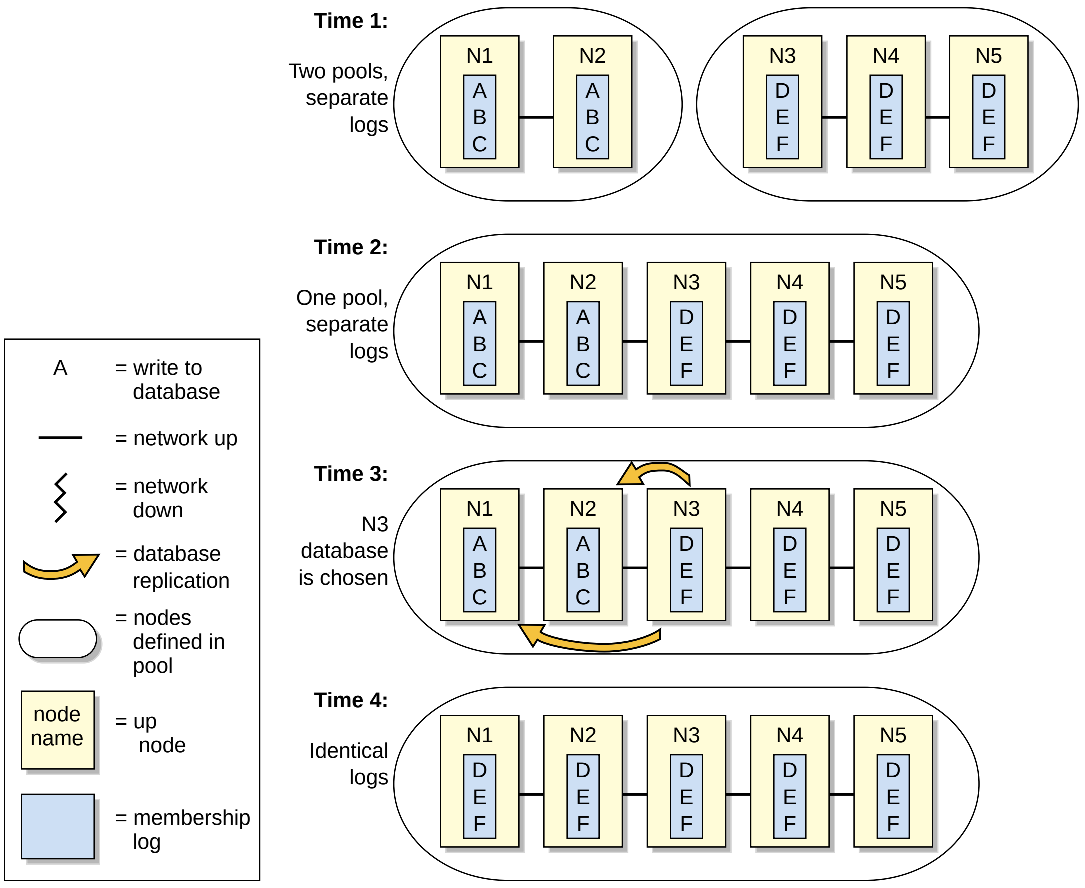


Figure B-2 Unequally Sized Pools are Joined: Log from Larger Pool is Replicated

Figure B-3 on page 438, shows a similar situation in which two nodes are defined in two pools, but the pools are of equal size:

- Time 1: There are two pools that do not share membership log contents. Each pool has two nodes (N1/N2 in pool 1, and N3/N4 in pool 2).
- Time 2: N1 and N2 are defined as part of the second pool by connecting the GUI or running `cxfs_admin` on node N3 or N4. This results in a new pool with four nodes with different membership logs.

- Time 3: Because each set has the same number of nodes, the tiebreaker node (the node with the lowest node ID in the membership) must be used to determine whose database will be chosen. Because node N1 is the lowest node ID (node ID=1), the database from N1 is chosen.
- Time 4: All nodes in the pool have identical membership logs.

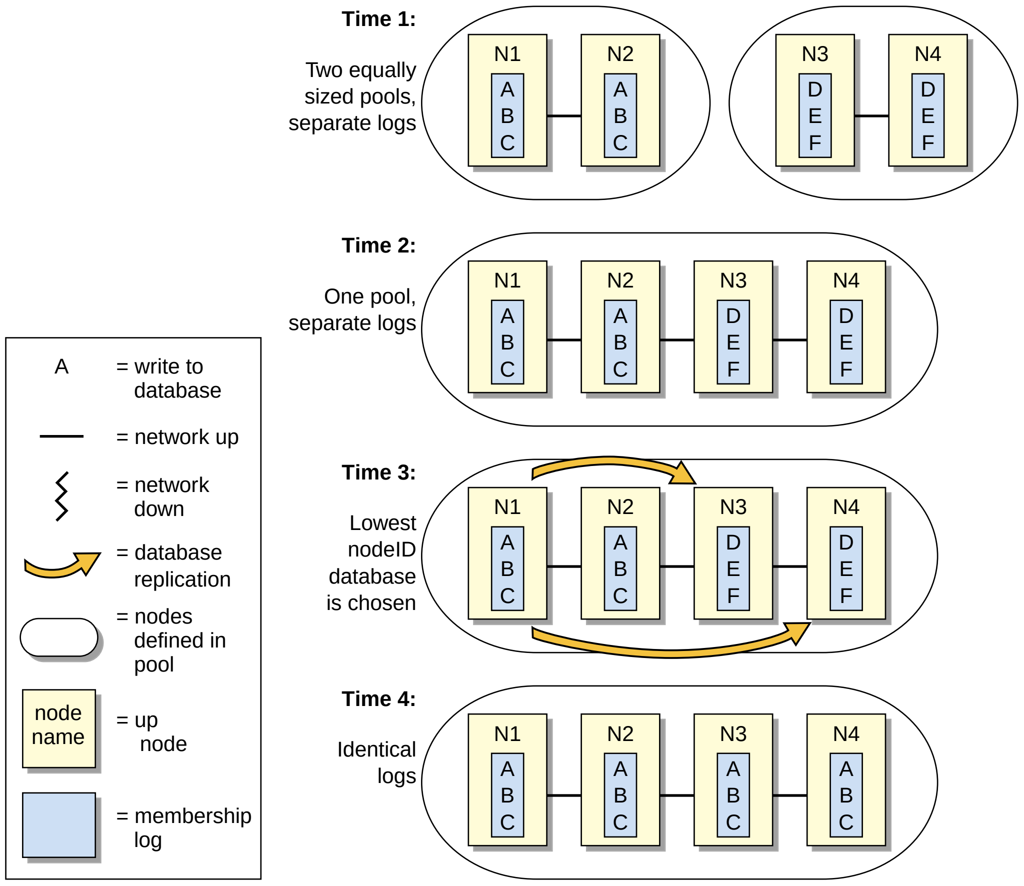


Figure B-3 Equally Sized Pools are Joined: Log from Node with Lowest Node ID is Replicated

Quorum and Tiebreaker Examples

Changing CXFS Kernel Membership Quorum Example

Figure B-4 on page 440, shows an example of a changing CXFS kernel membership quorum. It shows a pool of:

- Five CXFS server-capable administration nodes (A, B, C, D, and E)
- Two client-only nodes (F and G)
- One client admin node (H)

All nodes except E are defined as part of the cluster. Assume that CXFS services have been enabled on A, B, C, D, F, G, and H.

Of the seven nodes eligible for CXFS kernel membership, four are server-capable administration nodes (A, B, C, and D). Therefore, at least three of these four nodes must be able to communicate with each other to form an initial CXFS kernel quorum (>50% of the eligible server-capable administration nodes). Once the quorum has been reached, a membership will form with the nodes in the quorum plus all other eligible nodes that can communicate with the nodes in the quorum.

Figure B-4 on page 440, shows the following:

- Time 1: The CXFS kernel membership quorum is formed with three server-capable administration nodes, A, B, and C. The membership is A, B, C, F, G, and H.
- Time 2: Node B shuts down and leaves the membership. The remaining nodes in the quorum are A and C. The membership is still be available in this case because it satisfies the quorum requirement to maintain 50% of the eligible server-capable administration nodes (that is, two of the four server-capable administration nodes). The membership is A, C, F, G, and H.
- Time 3: Node A also shuts down and leaves the membership. Therefore, the quorum requirement is no longer met because quorum cannot be maintained with fewer than 50% of the eligible server-capable administration nodes. Without a quorum, the membership cannot continue, and so the CXFS filesystems in the cluster would not be available.

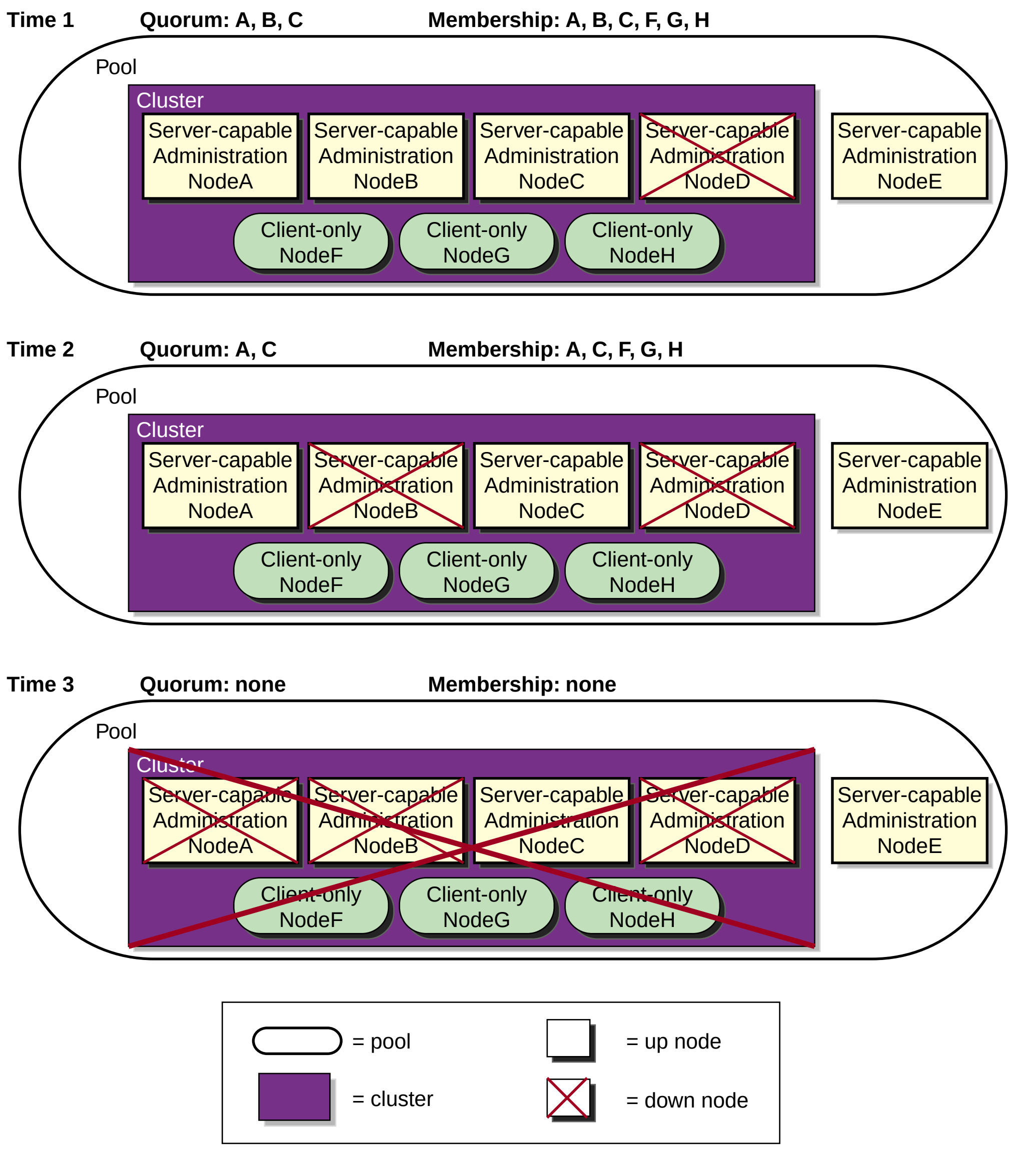

Time 1
Quorum: A, B, C
Membership: A, B, C, F, G, H
Pool
Cluster
Server-capable Administration NodeA
Server-capable Administration NodeB
Server-capable Administration NodeC
Server-capable Administration NodeD
Server-capable Administration NodeE
Client-only NodeF
Client-only NodeG
Client-only NodeH
Time 2
Quorum: A, C
Membership: A, C, F, G, H
Time 3
Quorum: none
Membership: none
= pool
= cluster
= up node
= down node

Figure B-4 Changing Quorum for CXFS Kernel Membership

CXFS Tiebreaker Node Example

Figure B-5 on page 442, displays a situation in which a private-network router dies and the CXFS kernel network is effectively split in two. The potential CXFS kernel membership is defined to be nodes A, B, C, and D. The nodes on network segment 2 (nodes C and D) will leave the CXFS kernel membership because they do not contain the CXFS tiebreaker node, and therefore do not have a quorum. On network segment 1, one of the other two potential metadata servers will become active and the membership will only include the systems on network segment 1. The nodes that were on network segment 2 will remain out of the membership until after CXFS services are restarted on them and the router is repaired, as shown in Time 3.

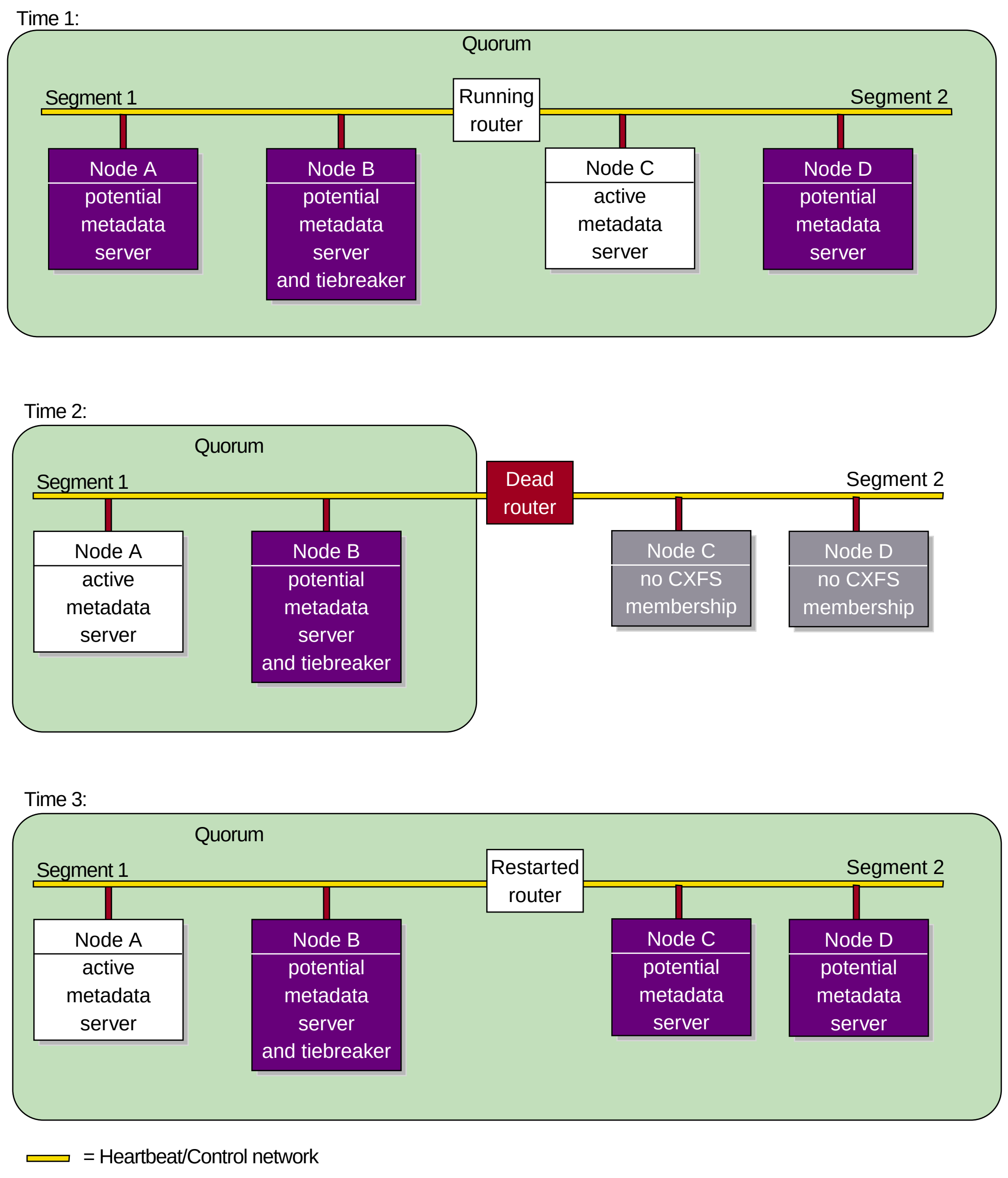

Time 1:
Quorum
Segment 1
Running router
Segment 2
Node A potential metadata server
Node B potential metadata server and tiebreaker
Node C active metadata server
Node D potential metadata server
Time 2:
Quorum
Segment 1
Dead router
Segment 2
Node A active metadata server
Node B potential metadata server and tiebreaker
Node C no CXFS membership
Node D no CXFS membership
Time 3:
Quorum
Segment 1
Restarted router
Segment 2
Node A active metadata server
Node B potential metadata server and tiebreaker
Node C potential metadata server
Node D potential metadata server
= Heartbeat/Control network

Figure B-5 CXFS Tiebreaker Node

CXFS Kernel Heartbeat and Cluster Database Heartbeat Considerations

There are different heartbeats for each membership type, and each uses a different networking method. Therefore, certain network misconfiguration can cause one heartbeat to fail while another succeeds.

If the highest priority network fails, the cluster database and CXFS kernel memberships will continue using the next priority network. (However, if private network failover has not been defined for CXFS, then the CXFS kernel membership will fail.)

CXFS Recovery Issues in a Cluster with Only Two Server-Capable Administration Nodes

A cluster with an odd number of server-capable administration nodes is recommended for a production environment. However, if you use a production cluster with an even number of server-capable administration nodes (especially only two server-capable administration nodes), you must do one of the following:

- Use system reset configuration on potential metadata servers to protect data integrity and improve server reliability. I/O fencing (or system reset when available) must be used on client-only nodes. Clusters should have an odd number of server-capable administration nodes or use a client-only tiebreaker node.
- Set a CXFS tiebreaker node. If the tiebreaker node is a server-capable administration node, there will be a loss of CXFS kernel membership, and therefore CXFS filesystems, if the tiebreaker node goes down. If the tiebreaker is a server-capable administration node, the cluster database membership may also be lost.

However, even with these methods, there are recovery and relocation issues inherent to a cluster with only two server-capable administration nodes.

IP Filtering for the CXFS Private Network

The method used for IP filtering for the CXFS private network varies by OS platform:

- IRIX can use `ipfilter` or `ipfilterd`
- SGI ProPack uses `SuSEfirewall2` or `iptables`

Note: The `SuSEfirewall2` utility silently makes changes to the `/proc/sys/net` system tunable parameters.

If you use I/O fencing, the configuration must allow communication between the node and the `telnet` port on the switch. For details about port use, see "CXFS Port Usage" on page 288.

For more information about these utilities, see their documentation.

Following is an example `/etc/ipfilterd.conf` file for configuring CXFS. Note the following:

- There must be an `/etc/ipfilterd.conf` file configured on each node on which you want to filter IP traffic. The files will be similar except for the first set of lines, which are node-dependent; that is, the lines in the file for `NodeA` must match the networking interfaces on which the network traffic may pass for `NodeA`.
- The `systune` variable `ipfilterd_inactive_behavior` must be set to 0, which means that the filter will be disabled as soon as `ipfilterd` is terminated using the `killall` command.
- The `ipfilterd` argument to `chkconfig` must be turned on for each node where `ipfilterd` will run. For example:

  ```
  NodeA# chkconfig ipfilterd on
  ```

- If any network interface name is changed on a system, you must update the `/etc/ipfilterd.conf` file to include the change in the appropriate `accept` line. That is:

  ```
  accept -i changed_or_new_interface
  ```

- For debugging purposes, each dropped packet will log a message similar to the following in the `syslog` file:

```
May 24 16:44:44 5A:rodin unix: NOTICE: ipfilter(cache) - packet dropped:
10.1.1.5 SPT=137 DPT=137 UDP
```

If you want to disable the filtering, such as in the case where it is blocking wanted traffic, do the following:

1. Kill the `ipfilterd` daemon:

   ```
   NodeA# killall ipfilterd
   ```

2. Turn off the `ipfilterflag` argument:

   ```
   NodeA# chkconfig ipfilterd off
   ```

Following is a sample file for `NodeA`:

```
NodeA# cat ipfilterd.conf
#
# ipfilterd.conf for NodeA
#
#
# Filters follow:
#
# Do not restrict traffic on any of the interfaces for NodeA,
# except from ef1 (CXFS private network)
#
accept -i lo0
accept -i ef0
accept -i eg0
accept -i eg1
accept -i lb0

#
# Restrict access over the CXFS private network
# Interface ef1
#

# Accept any fragment, reassembly won’t work if first fragment filtered out.
accept -i ef1 ip.off>0

# CXFS is using RPC, need portmapper.
```

```
accept -i ef1 udp.port 111
accept -i ef1 tcp.port 111


# fs2d daemon is dynamically assigning ports in range 600-1023.
# We need port definition (sport + dport for both directions).
accept -i ef1 tcp.sport>=600 and tcp.sport<=1023
accept -i ef1 tcp.dport>=600 and tcp.dport<=1023


# sgi-cad defaults to 9000/tcp
accept -i ef1 tcp.port 9000

# sgi-crsd
# Each node opens 7500/udp, both directions needed
accept -i ef1 udp.port 7500

# Uncomment the line below for CXFS client-only node.
# accept -i ef1 udp.port 5449


# CXFS kernel ports 5450-5453
# Connections in both directions so open dport and sport.
accept -i ef1 tcp.port 5450
accept -i ef1 tcp.port 5451
accept -i ef1 udp.port 5452
accept -i ef1 udp.port 5453

# fs2d client are using ports in range 7000-8500
accept -i ef1 tcp.dport>7000
accept -i ef1 udp.dport>7000

# Uncomment the line below for IO fencing only if switches are on CXFS private network
#  (ip.src is the switch address)
# accept -i ef1 tcp.sport=23 and ip.src=10.1.1.6

# Let icmp traffic pass, especially ’PORT UNREACHABLE ICMP packet’
accept -i ef1 icmp
# Reject the rest (-l will log any rejected packet to the SYSLOG)
reject -i ef1 -l
```

Path Summary

This appendix lists the locations for commonly used commands on server-capable administration nodes.

Table D-1 Paths for Server-Capable Administration Nodes

Command/File	SGI ProPack
`cdbreinit`	`/usr/cluster/bin/cdbreinit`
`chkconfig`	`/bin/chkconfig`
`cxfs_admin`	`/usr/cluster/bin/cxfs_admin`
`df`	`/bin/df`
`grioadmin`	`/usr/sbin/grioadmin`
`grioqos`	`/usr/sbin/grioqos`
`hinv`	`/usr/bin/hinv`
`hostname`	`/bin/hostname`
`mount`	`/bin/mount`
`netstat`	`/bin/netstat`
`ping`	`/bin/ping`
`ps`	`/bin/ps`
`xvm`	`/sbin/xvm`
Cluster daemon configuration files	`/etc/cluster/config/`
System log	`/var/log/messages`
CXFS/cluster daemon initialization	`/etc/init.d/cxfs_cluster`
CXFS license verification command:	`/usr/cluster/bin/cxfslicense`

System Reset Configuration

This appendix discusses system controllers that can be used in CXFS system reset configurations:

- "BMC System Controller" on page 451
- "L2 System Controller" on page 457
- "L1 System Controller" on page 464
- "MSC System Controller" on page 466
- "MMSC System Controller" on page 469

Note: Serial cables are provided with SAN server configurations for TTY ports. Other configurations that use TTY ports require that you purchase serial cables.

BMC System Controller

SGI recommends that you use the baseboard management controller (BMC) system controller when it is available.

The BMC must not be on the primary CXFS private network. Ideally, the BMC should be on a different private network that is reachable by all server-capable administration nodes in the cluster. A public network is not ideal for security reasons, but is acceptable.

Altix XE systems contain an integrated BMC. CXFS uses Intelligent Platform Management Interface (IPMI) to communicate with the BMC.

To use the BMC, you must create an `admin` user ID and assign the BMC a static IP address. This can be done using `ipmitool`(1) on the system containing the BMC. Do the following:

1. Verify that you have `ipmitool`(1) version 1.8.9 or later:

   ```
   # ipmitool -V
   ipmitool version 1.8.9
   ```

2. Load the following IPMI modules:

```
# modprobe ipmi_msghandler
# modprobe ipmi_devintf
# modprobe ipmi_si
```

3. Create a user named `admin` with a password and `ADMINISTRATOR` privileges:

   a. Find the next available user ID:

   ```
   ipmitool -d /dev/ipmi0 user list 1|2
   ```

   b. Assign the user name `admin` to the next available user ID:

```
ipmitool -d /dev/ipmi0 user set name userID admin
```

   c. Set the password for user `admin`:

```
ipmitool -d /dev/ipmi0 user set password userID admin_password
```

   d. Enable the access modes and set the privilege level to `ADMINISTRATOR`:

```
ipmitool -d /dev/ipmi0 channel setaccess 1|2 userID callin=on ipmi=on link=on privilege=4
```

   Note: You must apply the privilege change separately for channel 1 and for channel 2.

   e. Verify that the correct settings were applied:

   ```
   ipmitool -d /dev/ipmi0 user list 1|2
   ipmitool -d /dev/ipmi0 channel getaccess 1|2 userID
   ```

   For example (line breaks shown here for readability):

```
# ipmitool -d /dev/ipmi0 user list 1
ID  Name             Callin  Link Auth  IPMI Msg   Channel Priv Limit
1                    true    false      true       ADMINISTRATOR

# ipmitool -d /dev/ipmi0 user list 2
ID  Name             Callin  Link Auth  IPMI Msg   Channel Priv Limit
1                    true    false      true       ADMINISTRATOR
```

```
# ipmitool -d /dev/ipmi0 user set name 2 admin
# ipmitool -d /dev/ipmi0 user set password 2 password
# ipmitool -d /dev/ipmi0 channel setaccess 1 2 callin=on \
ipmi=on link=on privilege=4
[root@linux root]# ipmitool -d /dev/ipmi0 channel setaccess 2 2 callin=on \
ipmi=on link=on privilege=4

# ipmitool -d /dev/ipmi0 user list 1
ID  Name             Callin  Link Auth  IPMI Msg   Channel Priv Limit
1                    true    false      true       ADMINISTRATOR
2   admin            true    true       true       ADMINISTRATOR

# ipmitool -d /dev/ipmi0 user list 2
ID  Name             Callin  Link Auth  IPMI Msg   Channel Priv Limit
1                    true    false      true       ADMINISTRATOR
2   admin            true    true       true       ADMINISTRATOR

# ipmitool -d /dev/ipmi0 channel getaccess 1 2
Maximum User IDs     : 15
Enabled User IDs     : 2
User ID              : 2
User Name            : admin
Fixed Name           : No
Access Available     : call-in / callback
Link Authentication  : enabled
IPMI Messaging       : enabled
Privilege Level      : ADMINISTRATOR

# ipmitool -d /dev/ipmi0 channel getaccess 2 2
Maximum User IDs     : 15
Enabled User IDs     : 2
User ID              : 2
User Name            : admin
Fixed Name           : No
Access Available     : call-in / callback
Link Authentication  : enabled
IPMI Messaging       : enabled
Privilege Level      : ADMINISTRATOR
```

4. Apply the following local area network (LAN) settings for the BMC on the Altix XE system, for which the IPMI device is `/dev/ipmi0`. The BMC LAN settings apply to LAN channels 1 and 2.

   Note: You must apply each change separately for channel 1 and for channel 2.

   - Set the IP Address (use the same IP address for both channels):

     ```
     ipmitool -d /dev/ipmi0 lan set 1|2 ipaddr IP_address
     ```

   - Set the subnet mask (use the same value for both channels):

     ```
     ipmitool -d /dev/ipmi0 lan set 1|2 netmask netmask
     ```

   - Enable address resolution protocol (ARP) responses:

     ```
     ipmitool -d /dev/ipmi0 lan set 1|2 arp respond on
     ```

   - Enable *gratuitous ARP*, which broadcasts the MAC address to IP address mappings on a specified interface:

     ```
     ipmitool -d /dev/ipmi0 lan set 1|2 arp generate on
     ```

   - Set the gratuitous ARP interval (in seconds):

   Note: An interval of 5 seconds is supported for CXFS.

     ```
     ipmitool -d /dev/ipmi0 lan set 1|2 arp interval 5
     ```

For example:

```
# ipmitool -d /dev/ipmi0 lan set 1 ipaddr nodename-bmc.company.com
Setting LAN IP Address to nodename-bmc.company.com
# ipmitool -d /dev/ipmi0 lan set 2 ipaddr nodename-bmc.company.com
Setting LAN IP Address to nodename-bmc.company.com
# ipmitool -d /dev/ipmi0 lan set 1 netmask 255.255.0.0
Setting LAN Subnet Mask to 255.255.0.0
# ipmitool -d /dev/ipmi0 lan set 2 netmask 255.255.0.0
Setting LAN Subnet Mask to 255.255.0.0
# ipmitool -d /dev/ipmi0 lan set 1 arp respond on
Enabling BMC-generated ARP responses
# ipmitool -d /dev/ipmi0 lan set 2 arp respond on
Enabling BMC-generated ARP responses
# ipmitool -d /dev/ipmi0 lan set 1 arp generate on
Enabling BMC-generated Gratuitous ARPs
# ipmitool -d /dev/ipmi0 lan set 2 arp generate on
Enabling BMC-generated Gratuitous ARPs
# ipmitool -d /dev/ipmi0 lan set 1 arp interval 5
BMC-generated Gratuitous ARP interval:  5.0 seconds
# ipmitool -d /dev/ipmi0 lan set 2 arp interval 5
BMC-generated Gratuitous ARP interval:  5.0 seconds
```

5. Verify your changes by using the following command:

   ```
   ipmitool -d /dev/ipmi0 lan print 1|2
   ```

   For example:

```
# ipmitool -d /dev/ipmi0 lan print 1
Set in Progress         : Set Complete
Auth Type Support       : NONE MD5 PASSWORD
Auth Type Enable        : Callback :
                        : User     :
                        : Operator :
                        : Admin    : MD5 PASSWORD
                        : OEM      :
IP Address Source       : Static Address
IP Address              : nodename-bmc.company.com
Subnet Mask             : 255.255.0.0
MAC Address             : 00:04:23:d5:af:3c
SNMP Community String   :
```

```
IP Header               : TTL=0x40 Flags=0x40 Precedence=0x00 TOS=0x10
BMC ARP Control         : ARP Responses Enabled, Gratuitous ARP Enabled
Gratituous ARP Intrvl   : 5.0 seconds
Default Gateway IP      : 0.0.0.0
Default Gateway MAC     : 00:00:00:00:00:00
Backup Gateway IP       : 0.0.0.0
Backup Gateway MAC      : 00:00:00:00:00:00
RMCP+ Cipher Suites     : 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14
Cipher Suite Priv Max   : XXXXXXXXXXXXXXX
                        :     X=Cipher Suite Unused
                        :     c=CALLBACK
                        :     u=USER
                        :     o=OPERATOR
                        :     a=ADMIN
                        :     O=OEM
```

6. Verify the BMC configuration and connectivity from a remote node by issuing `ipmitool`(1) commands remotely :

```
ping IP_address_or_hostname
ipmitool -H IP_address_or_hostname -U admin -P admin_passwd lan print 1|2
```

For example (line breaks shown here for readability):

```
# ping nodename-bmc.company.com

# ipmitool -H nodename-bmc.company.com -U admin \
-P mypassword lan print 1
Set in Progress         : Set Complete
Auth Type Support       : NONE MD5 PASSWORD
Auth Type Enable        : Callback :
                        : User     :
                        : Operator :
                        : Admin    : MD5 PASSWORD
                        : OEM      :
IP Address Source       : Static Address
IP Address              : nodename-bmc.company.com
Subnet Mask             : 255.255.0.0
MAC Address             : 00:04:23:d5:af:3c
SNMP Community String   :
IP Header               : TTL=0x40 Flags=0x40 Precedence=0x00 TOS=0x10
BMC ARP Control         : ARP Responses Enabled, Gratuitous ARP Enabled
```

```
Gratuitous ARP Intrvl   : 5.0 seconds
Default Gateway IP      : 0.0.0.0
Default Gateway MAC     : 00:00:00:00:00:00
Backup Gateway IP       : 0.0.0.0
Backup Gateway MAC      : 00:00:00:00:00:00
RMCP+ Cipher Suites     : 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14
Cipher Suite Priv Max   : XXXXXXXXXXXXXXX
                        :     X=Cipher Suite Unused
                        :     c=CALLBACK
                        :     u=USER
                        :     o=OPERATOR
                        :     a=ADMIN
                        :     O=OEM
```

For more information, see the `ipmitool`(1) man page, *Guide to Programming Environments and Tools Available on SGI Altix XE System*, and the user guide or quick start guide for your system.

L2 System Controller

The L2 system controller and the required USB cables are optional equipment available for purchase. The L2 method is recommended when available.

Note: You can use network reset if you have an L2 on a network. For details, see the information about `reset_comms` in "Create or Modify a Node with `cxfs_admin`" on page 235.

The L2 controller must not be on the primary CXFS private network. Ideally, the L2 controller should be on a different private network and must be reachable by all server-capable administration nodes in the cluster. A public network is not ideal for security reasons, but is acceptable. The number of network connections allowed depends upon the L2 version; contact your SGI support person for assistance.

Altix systems with an integrated L2 (such as a NUMAlink 4 R-brick), Altix 3000 Bx2 systems, Altix 450 systems and Altix 4700 systems use the L2 over Ethernet. See Figure E-1.

In Altix 350, use IO10 and a *multiport serial adapter cable*, which is a device that provides four DB9 serial ports from a 36-pin connector; see Figure E-2.

Use the modem port on the L2 system controller as shown in Figure E-3. Use DB9 serial ports on an IX-brick on Altix 3000 and Origin 3000. Connect the serial cable to the modem port on one end and the serial port on the IX-brick (for example, serial port connector 0), as shown in Figure E-4.

Figure E-5, Figure E-6, and Figure E-7 show serial connections for two machines with an L2 system controller. (These figure shows direct attached storage. Serial connections for other storage configurations will be the same.)

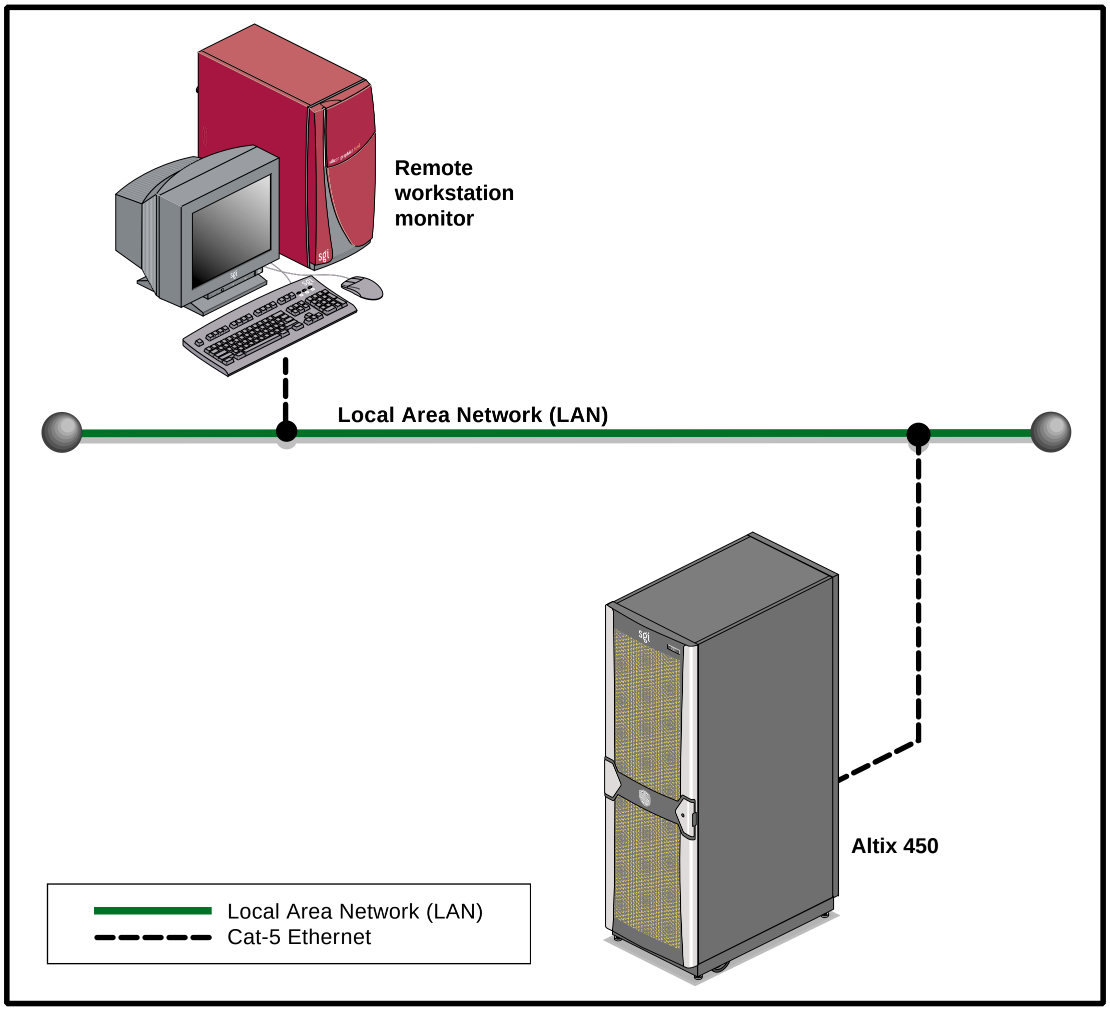


Figure E-1 SGI Altix 450 System Control Network

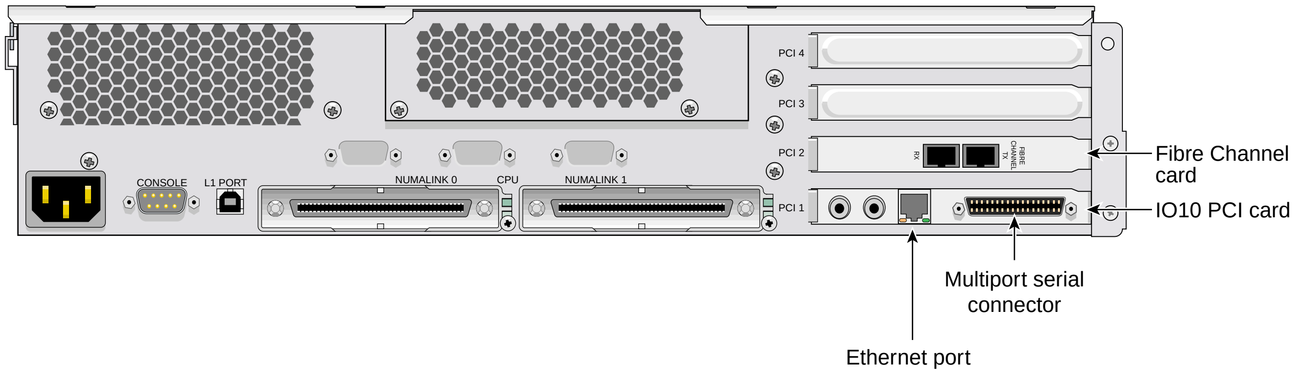


Figure E-2 Altix 350 Rear Panel

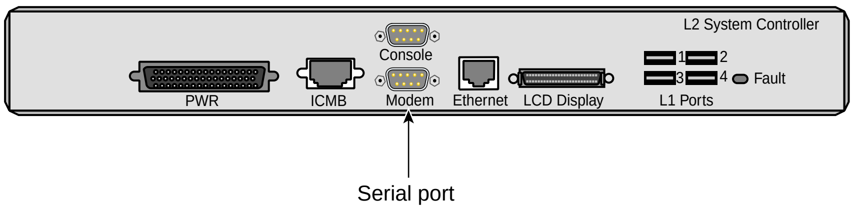


Figure E-3 L2 Rear Panel

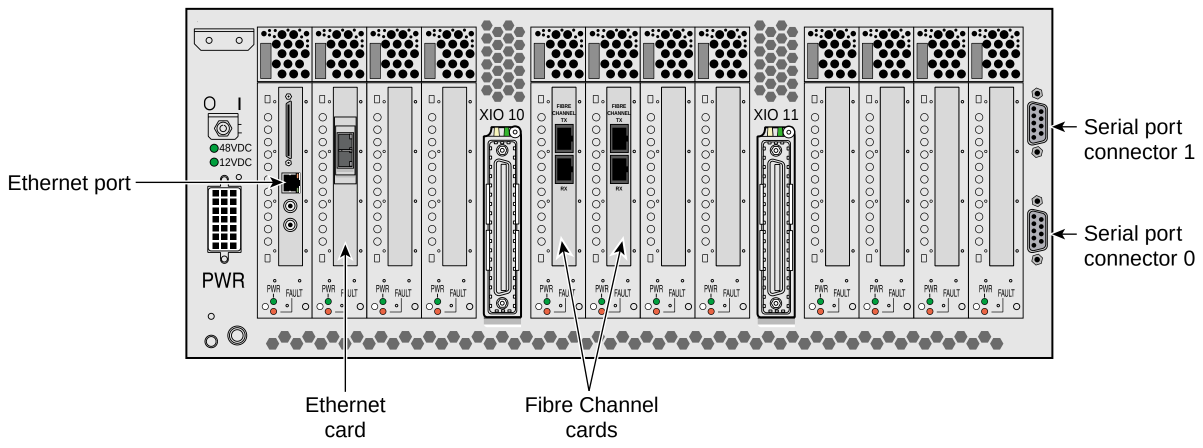


Figure E-4 IX-brick Rear Panel

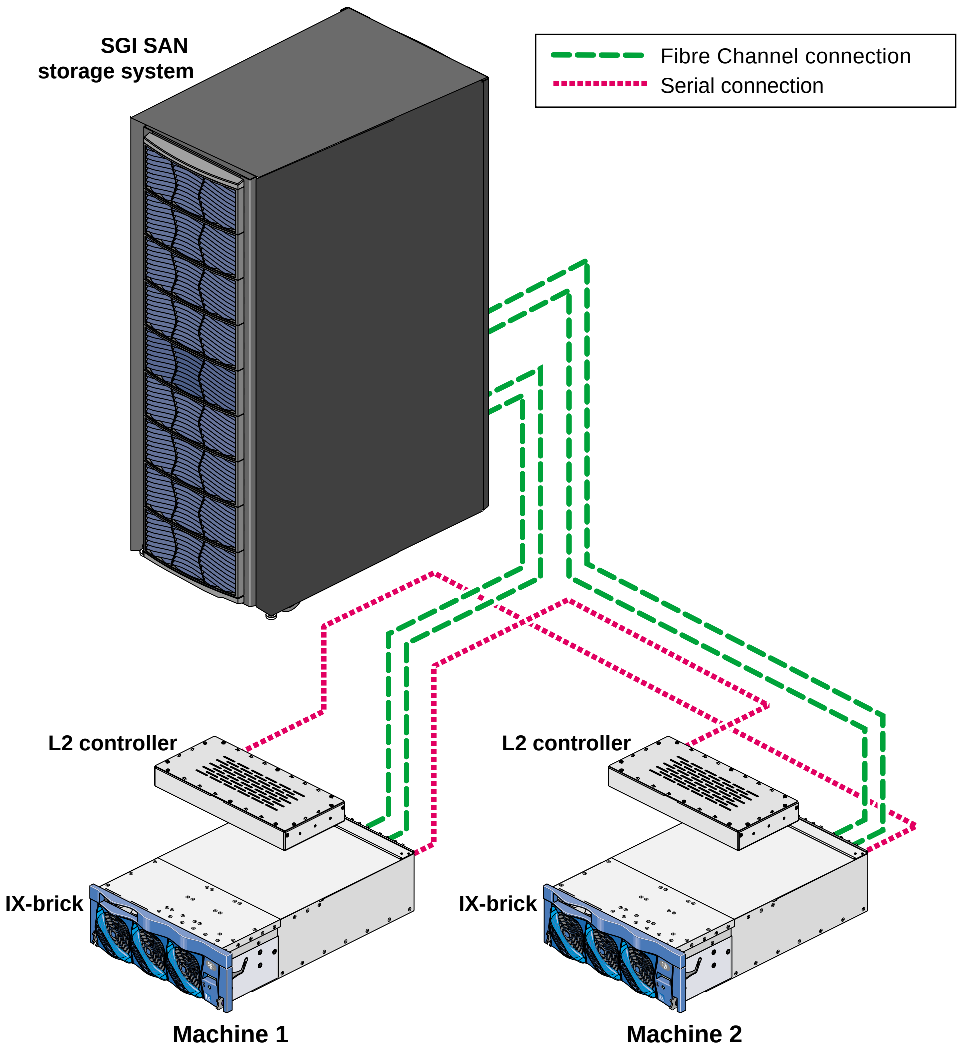


Figure E-5 Altix 3000 and Origin 3000 Serial Connections

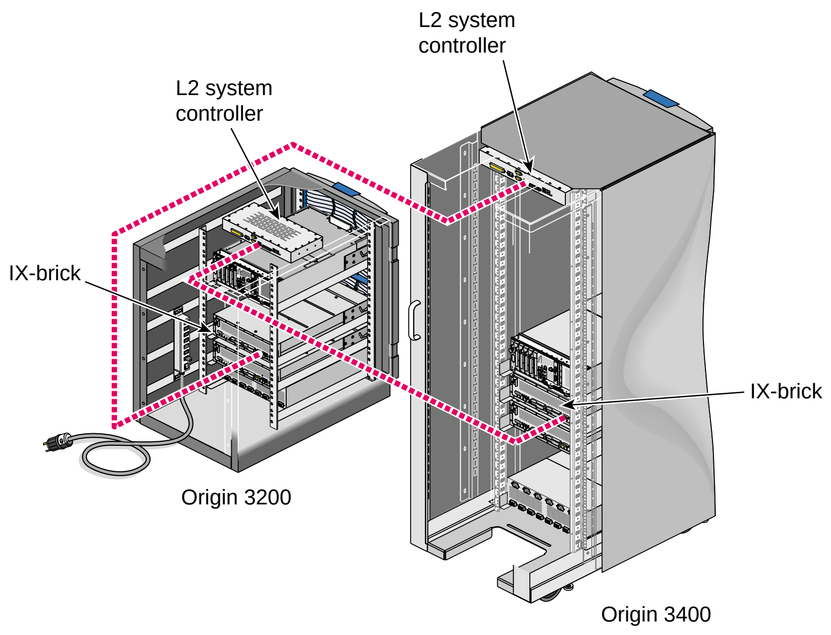


Figure E-6 Serial Connection Between SGI Origin 3200 and Origin 3400/3800 Servers

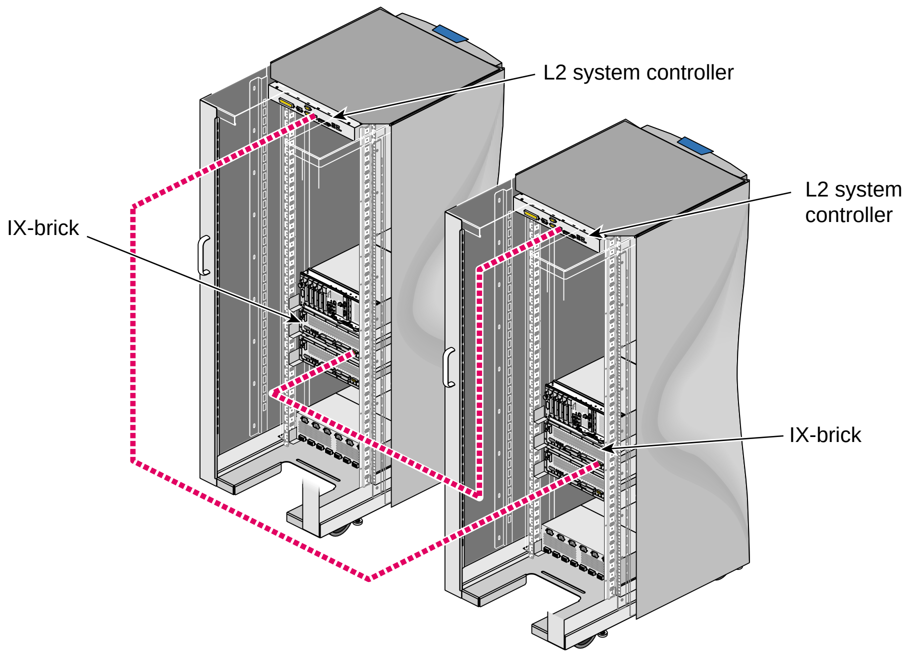


Figure E-7 Serial Connection Between Two SGI Origin 3400 or SGI Origin 3800 Servers

L1 System Controller

L1 system controller can be used for reset, however, SGI recommends the use of L2 when available.

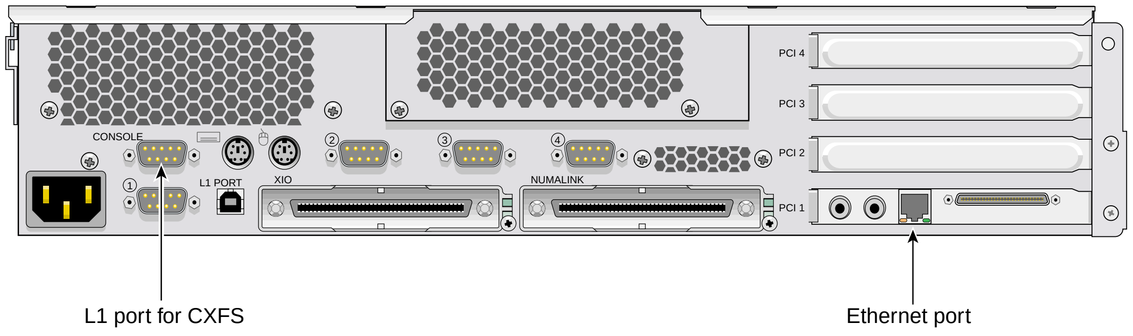


Figure E-8 Origin 350 Rear Panel

Connect the serial cable to the console port (port labeled **CONSOLE**) on one end and the serial port of other node on the other end. The serial ports on Origin 350 are labeled as **1**, **2**, **3**, and **4**; see Figure E-8.

Note: The USB port on the Origin 350 is labeled **L1 PORT**. Do not use this port for system reset. Use the port labeled **CONSOLE** as shown in Figure E-8.

Redirecting the Console for Origin 300, Origin 350, Origin 3200C, Onyx 300, Onyx 350, and Onyx 3200C

On Origin 300, Origin 350, Origin 3200C, Onyx 300, Onyx 350, and Onyx 3200C systems, there is only one serial/USB port that provides both L1 system controller and console support for the machine. In a CXFS configuration, this port (the DB9 connector) is used for system reset. It is connected to a serial port in another node or to the Ethernet multiplexer.

To get access to console input and output, you must redirect the console to another serial port in the machine.

Use the following procedure to redirect the console:

1. Edit the `/etc/inittab` file to use an alternate serial port.
2. Either issue an `init q` command or reboot.

For example, suppose you had the following in the `/etc/inittab` file (line breaks added for readability):

```
# on-board ports or on Challenge/Onyx MP machines, first IO4 board ports
t1:23:respawn:/sbin/suattr -C CAP_FOWNER,CAP_DEVICE_MGT,CAP_DAC_WRITE+ip
-c "exec /sbin/getty ttyd1 console"    # alt console
t2:23:off:/sbin/suattr -C CAP_FOWNER,CAP_DEVICE_MGT,CAP_DAC_WRITE+ip
-c "exec /sbin/getty -N ttyd2 co_9600"     # port 2
```

You could change it to the following:

```
# on-board ports or on Challenge/Onyx MP machines, first IO4 board ports
t1:23:off:/sbin/suattr -C CAP_FOWNER,CAP_DEVICE_MGT,CAP_DAC_WRITE+ip
-c "exec /sbin/getty ttyd1 co_9600"       # port 1
t2:23:respawn:/sbin/suattr -C CAP_FOWNER,CAP_DEVICE_MGT,CAP_DAC_WRITE+ip
-c "exec /sbin/getty -N ttyd2 console" # alt console
```

Caution: Redirecting the console by using the above method works only when the IRIX operating system is running. To access the console when the operating system is not running (miniroot) , you must physically reconnect the machine: unplug the serial hardware reset cable from the console/L1 port and then connect the console cable.

MSC System Controller

Figure E-9 and Figure E-10 show the serial connection between two deskside servers.

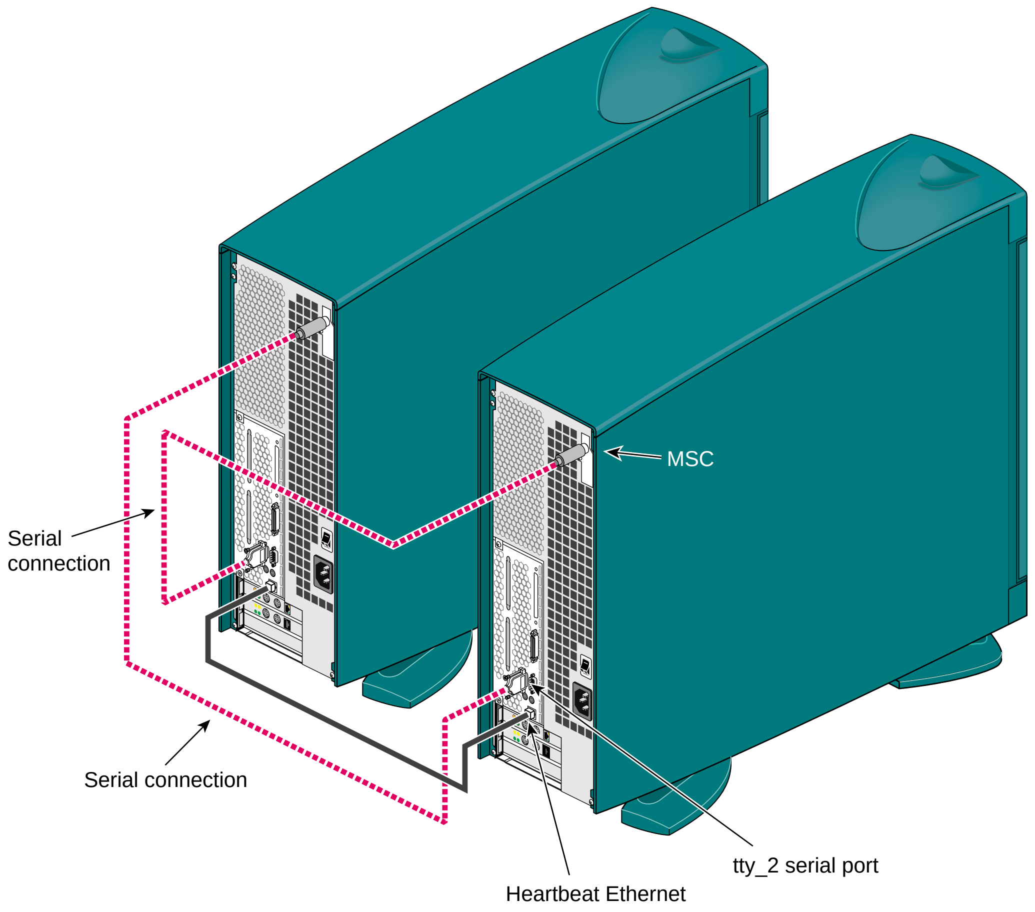


Figure E-9 Serial Connection Between Two Origin 200 Deskside Servers

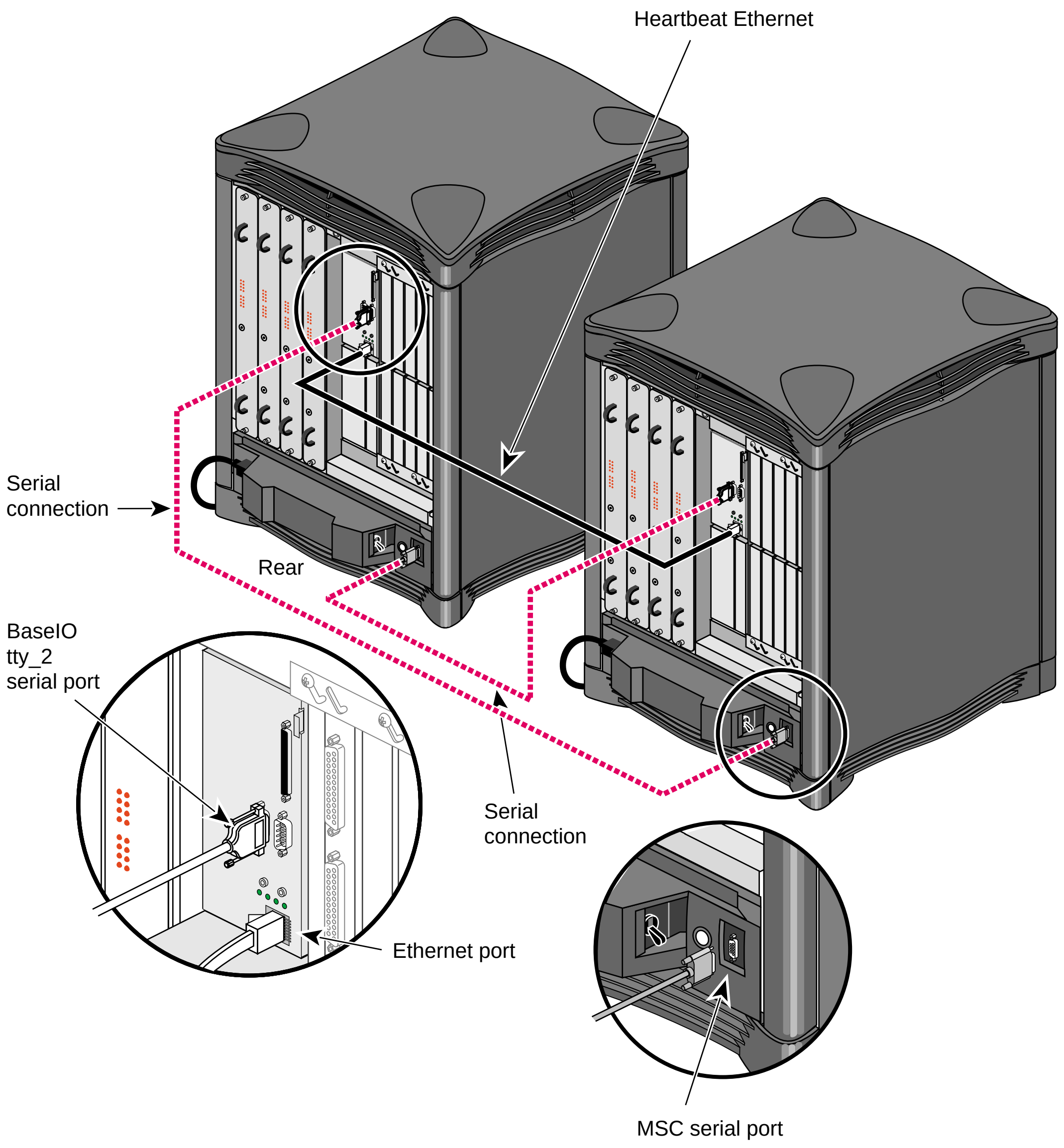


Figure E-10 Serial Connection Between Two SGI 2200 Deskside Servers

MMSC System Controller

Figure E-11 shows the MMSC. The alternate console port should be connected to the serial port on another machine using a serial cable.

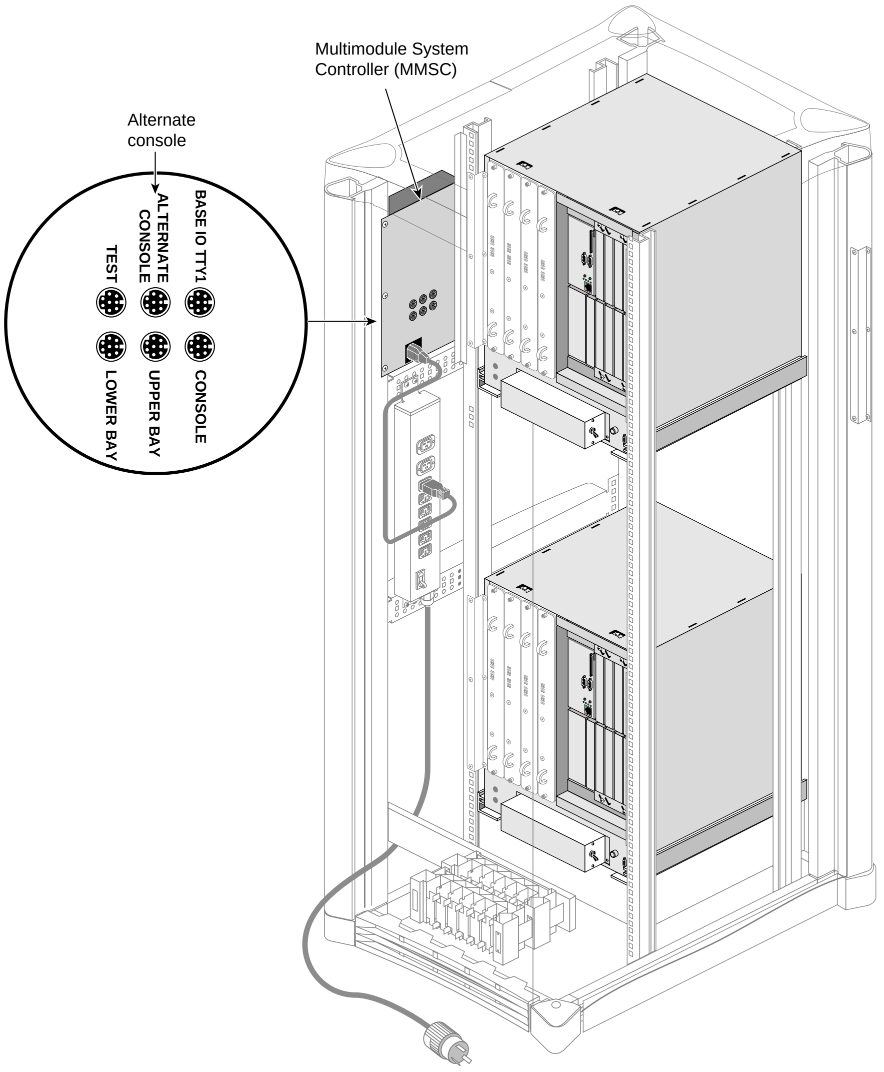


Figure E-11 MMSC Serial Port

System Tunable Parameters

This appendix discusses the following:

- "Site-Changeable System Tunable Parameters" on page 471
- "Restricted System Tunable Parameters" on page 479

Also see "Set System Tunable Parameters Appropriately" on page 75.

Site-Changeable System Tunable Parameters

This section lists the CXFS system tunable parameters that you can change. SGI recommends that you use the same settings on all applicable nodes in the cluster.

Note: Before changing any parameter, you should understand the ramifications of doing so on your system. Contact your SGI support person for guidance.

To manipulate these parameters on a running system, you can use the Linux `sysctl` command or the IRIX `systune` command. For more information, see the `sysctl`(1M), `systune`(1M), and `modules.conf`(5) man pages.

Linux organizes the tunables in a hierarchy, therefore you must specify the entire "path" to the tunable. The first part of that path is given under the "Location" entry in the following sections. For example, the full path to the tunable `cxfsd_sync_force` is `fs.cxfs.cxfsd_sync_force`.

Example of a query using `sysctl`:

```
admin# sysctl fs.cxfs.cxfsd_sync_force
fs.cxfs.cxfsd_sync_force = 8372224
```

Example of setting a value using `sysctl`:

```
admin# sysctl fs.cxfs.cxfsd_sync_force=0
fs.cxfs.cxfsd_sync_force = 0
```

Note: There cannot be spaces around the = character when setting a value.

See also "Set System Tunable Parameters Appropriately" on page 75.

Site-Changeable Static Parameters

Static parameters require a reboot to take affect. On IRIX, you must build and boot new kernels, which happens automatically during a normal boot process. On any of the Linux flavors supported in this CXFS release, you must specify the parameter in `/etc/modprobe.conf.local`.

`mtcp_hb_watchdog`

Controls the behavior of the CXFS kernel heartbeat monitor watchdog. This facility monitors the generation of heartbeats in the kernel.

Range of values:

- `0` species that there is no use of watchdog (default)
- `1` specifies that watchdog expiration causes CXFS shutdown
- `2` specifies that watchdog expiration causes panic

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

`mtcp_nodelay`

Specifies whether to enable or disable `TCP_NODELAY` on CXFS message channels.

Range of values:

- `0` disables
- `1` enables (default)

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

`mtcp_rpc_thread`

Specifies whether metadata messages are sent from a separate thread in order to save stack space.

Range of values:

- `0` disables (default for most nodes)
- `1` enables

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

`rhelpd_max`

Specifies the maximum number of `rhelpd` threads to run. The `rhelpd` threads help out recovery and relocation tasks. They are used for asynchronous inode reconstruction, parallel recoveries, and so on. The `rhelpd` thread pool is global in nature and gets created during module load time.

Range of values:

- Default: `0`, which specifies an automatically calculated value that will be 4 times the number of CPUS, as long as it is in the range `0` through `128`. To disable automatic `rhelpd_max` calculation, set `rhelpd_max` to a non-zero value.
- Minimum: `0`
- Maximum: `128`

Location:

- IRIX: `/var/sysgen/mtune/cxfs`
- Linux: `fs.cxfs` (`sgi-cxfs` module)

`rhelpd_min`

Specifies the minimum number of `rhelpd` threads to run.

Range of values:

- Default: `0`, which specifies an automatically calculated value that is 4 times the number of CPUs or `128`, whichever is smaller. To disable automatic `rhelpd_min` calculation, set `rhelpd_min` to a non-zero value. When the value is set explicitly, the maximum is `8`.
- Minimum: `0`
- Maximum: `8`

Location:

- IRIX: `/var/sysgen/mtune/cxfs`
- Linux: `fs.cxfs` (`sgi-cxfs` module)

Site-Changeable Dynamic Parameters

Dynamic parameters take affect as soon as they are changed.

`cms_local_fail_action`

Specifies the action to take when a local node detects that it has failed:

Range of values:

- `0` withdraws from the cluster (default)
- `1` halts
- `2` reboots

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

`cxfs_client_push_period`

Specifies (in hundredths of a second) how long that a client may delay telling the metadata server that it has updated the `atime` timestamp of a file. The default for both `cxfs_client_push_period` and `cxfs_server_push_period` is 1/4 of a second, so `atime` updates are delayed by up to 1/2 second by default. See also "`cxfs_server_push_period`" on page 477.

Range of values:

- Default: `25`
- Minimum: `0`
- Maximum: `1000`

Location:

- IRIX: `/var/sysgen/mtune/cxfs`
- Linux: `fs.cxfs` (`sgi-cxfs` module)

`cxfs_dcvn_timeout`

Specifies the time-out (in seconds) of the `dcvn` idle period before returning tokens to the server.

Range of values:

- Default: `60`
- Minimum: `5`
- Maximum: `3600`

Location:

- IRIX: `/var/sysgen/mtune/cxfs`
- Linux: `fs.cxfs` (`sgi-cxfs` module)

`cxfs_extents_delta`

Specifies whether or not to optimize the way extent lists are sent across the private network by sending a delta when possible.

Range of values:

- `0` does not optimize
- `1` optimizes (default)

Location:

- IRIX: `/var/sysgen/mtune/cxfs`
- Linux: `fs.cxfs` (`sgi-cxfs` module)

`cxfs_punch_hole_restrict`

Specifies whether or not to allow exported files to have their extents freed by DMAPI via `dm_punch_hole()`.

Range of values:

- `0` allows extents to be freed (default)
- `1` does not allow extents to be freed

Location:

- IRIX: `/var/sysgen/mtune/cxfs`
- Linux: `fs.cxfs` (`sgi-cxfs` module)

`cxfs_relocation_ok`

Specifies whether relocation is disabled or enabled (must be specified on the active metadata server):

Range of values:

- `0` disables relocation (default)
- `1` enables relocation

Note: Relocation is disabled by default and is only supported on standby nodes. See:

- "Node Types, Node Functions, and the Cluster Database" on page 12
- "Relocation" on page 25

Location:

- Linux: `fs.cxfs` (`sgi-cxfs` module)

`cxfs_server_push_period`

Specifies (in hundredths of a second) how long that a metadata server may delay broadcasting to the clients that it has updated the `atime` timestamp. The default for both `cxfs_client_push_period` and `cxfs_server_push_period` is 1/4 of a second, so `atime` updates are delayed by up to 1/2 second by default. See also "`cxfs_client_push_period`" on page 474.

Range of values:

- Default: `25`
- Minimum: `0`
- Maximum: `1000`

Location:

- Linux: `fs.cxfs` (`sgi-cxfs` module)

`cxfsd_max`

Specifies the maximum number of `cxfsd` threads to run per CXFS filesystem. (The `cxfsd` threads do the disk block allocation for delayed allocation buffers in CXFS and the flushing of buffered data for files that are being removed from the local cache by the metadata server.) The threads are allocated at filesystem mount time. The value of the `cxfsd_max` parameter at mount time remains in effect for a filesystem until it is unmounted.

Range of values:

- Default: `0`, which specifies the value of `cxfsd_min` + 2. (The value for `cxfsd_max` is always at least `cxfsd_min` + 2, even if that forces the kernel to increase the value beyond 2048.) To disable automatic `cxfsd_max` calculation, set `cxfsd_max` to a non-zero value.
- Minimum: `16`
- Maximum: `2048`

Note: The value for `cxfsd_max` cannot be less than the value specified for `cxfsd_min`.

Location:

- IRIX: `/var/sysgen/mtune/cxfs`
- Linux: `fs.cxfs` (`sgi-cxfs` module)

`cxfsd_min`

Specifies the minimum number of `cxfsd` threads to run per CXFS filesystem. The value of the `cxfsd_min` parameter at mount time remains in effect for a filesystem until it is unmounted.

Range of values:

- Default: `0`, which specifies an automatically calculated value that will be 2 times the number of CPUS (the number of actual running `cxfsd` threads is dynamic), as long as it is in the range `16` through `2048`. To disable automatic `cxfsd_min` calculation, set `cxfsd_min` to a non-zero value.
- Minimum: `16`
- Maximum: `2048`

Location:

- IRIX: `/var/sysgen/mtune/cxfs`
- Linux: `fs.cxfs` (`sgi-cxfs` module)

`mtcp_mesg_validate`

Enables checksumming. Normally, this is not needed and is only used if TCP data corruption is suspected.

Range of values:

- `0` performs no validation (default)
- `1` generates checksums, but does not perform validation
- `2` generates and validates checksums, warns (via a `SYSLOG` message) on validation failure
- `3` generates and validates checksums, warns and returns an error message on validation failure

- `4` generates and validates checksums, warns and panics on validation error

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

Restricted System Tunable Parameters

This section lists the CXFS system tunable parameters that are provided for debugging purposes.

Caution: You must not modify any of these parameters unless directed to do so by SGI support.

Restricted Static Parameters

Static parameters require a reboot to take affect. On IRIX, you must build and boot new kernels, which happens automatically during a normal boot process. On Linux, you must specify the parameter in `/etc/modprobe.conf.local`. For more information, see "Site-Changeable System Tunable Parameters" on page 471.

`cxfs_extents_block_size`

Specifies the size in kilobytes of the units to use for memory allocations for extent lists on CXFS filesystems. You should only change this parameter for debugging purposes.

Range of values:

- Default: `0` (page size of the platform)
- Minimum: `0`
- Maximum: `256`

Location:

- IRIX: `/var/sysgen/mtune/cxfs`
- Linux: `fs.cxfs` (`sgi-cxfs` module)

`cxfs_extents_delta_depth`

Specifies the number of changes to the extent list kept by the CXFS metadata server for generating extents deltas. You should only change it for for debugging purposes. This parameter applies to only server-capable administration nodes.

Range of values:

- Default: `5`
- Minimum: `0`
- Maximum: `32`

Location:

- Linux: `fs.cxfs` (`sgi-cxfs` module)

`cxfs_shutdown_time`

Specifies the time (in seconds) that other nodes will wait for the node to take media offline after they have recognized that it has lost quorum, if the node has neither fencing nor reset configured. SGI recommends a value of 50 (0.5 seconds).

Range of values:

- Default: `50`
- Minimum: `0`
- Maximum: `6000`

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

`mesg_delay_time`

Specifies the amount of time to delay messages, in nsecs.

Range of values:

- Default: `0`

- Minimum: `0`
- Maximum: `1000000`

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

`mtcp_hb_local_options`

Specifies how CXFS kernel heartbeat is generated for a Linux node. You should only change this value at the recommendation of SGI support.

Legal values:

- `0x0` uses the standard heartbeat generation routine (default).
- `0x1` uses the interrupt timer list instead of a kernel thread.
- `0x3` uses a heartbeat generation routine that avoids some memory allocation problems that may occur on nodes with large CPU counts that run massively parallel jobs.

Location:

- Linux: `kernel.cell` (`sgi-cell` module)

`mtcp_hb_period`

Specifies (in hundredths of a second) the length of time that CXFS waits for CXFS kernel heartbeat from other nodes before declaring node failure. SGI recommends a value of 500 (5 seconds). You should only change this value at the recommendation of SGI support. The same value must be used on all nodes in the cluster.

Range of values:

- Default: `500`
- Minimum: `100`
- Maximum: `12000`

Note: If your cluster includes large Altix systems (greater than 64 processors) , you may want to use a larger value, such as `6000` (60 seconds) or `12000` (120 seconds). However, the larger the timeout, the longer it takes the cluster to recognize a failed node and start recovery of the shared resources granted to that node. See "Avoid CXFS Kernel Heartbeat Issues on Large SGI Altix Systems" on page 60.

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

`mtcp_reserve_size`

Sets the size of the TCP window in bytes.

Range of values:

- Default: `61440`
- Minimum: `2048`
- Maximum: `1073741824`

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

Restricted Dynamic Parameters

Dynamic parameters take affect as soon as they are changed.

`cell_tkm_feature_disable`

Disables selected features of the token module by setting a flag bit to one of the following:

- `0x1` disables speculative token acquisition
- `0x2` disables token prefetching

- `0x4` uses multiple RPCs to obtain a token set if rank and class conflict

Range of values:

- Default: `0`
- Minimum: `0`
- Maximum: `0x7fff`

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

Note: This parameter supersedes the following parameters:

- `cxfs_prefetch`, which enabled/disabled token obtain optimization
- `cxfs_speculative_token`, which enabled/disabled speculative vnode token fetching

---

`cms_fence_timeout`

Specifies the number of seconds to wait for `clconfd` to acknowledge a fence request. If a non-zero value is set and the time-out expires, CXFS takes the action specified by the `cms_fence_timeout_action` parameter. This parameter applies to only server-capable administration nodes.

Range of values:

- Default: `0` (infinite wait)
- Minimum: `0`
- Maximum: `10000`

Location:

- Linux: `kernel.cell` (`sgi-cell` module)

`cms_fence_timeout_action`

Specifies the action to be taken when `clconfd` does not acknowledge a fence request (determined by `cms_fence_timeout`). This parameters applies to only server-capable administration nodes.

Legal values:

- `0` proceeds as if the fence returned an error (default). This causes the node waiting for the fence acknowledgement to forcibly withdraw from the cluster, equivalent to a forced CXFS shutdown that occurs when a node loses quorum. If `clconfd` is still present and functioning properly, it will then restart the kernel `cms` daemon and the node will attempt to rejoin the cluster.
- `1` proceeds as if the fence succeeded. This clears all pending fence requests and continues (that is, fakes acknowledgment).

  Caution: Setting this value is potentially dangerous.

- `2` panics the local node.

Location:

- Linux: `kernel.cell` (`sgi-cell` module)

`cms_reset_error_override`

Specifies whether or not to ignore reset errors.

Legal values:

- `0` does not ignore reset errors (default)
- `1` ignores reset errors

  Caution: You should only set this value to `1` for testing purposes, and **never** on a production system.

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

`cms_reset_timeout`

Specifies the number of seconds to wait for `clconfd` to acknowledge a reset request. If you specify a non-zero value and the time-out expires, CXFS takes the action specified by the `cms_reset_timeout_action` parameter. This applies to only server-capable administration nodes.

Range of values:

- Default: `0` (infinite wait)
- Minimum: `0`
- Maximum: `10000`

Location:

- Linux: `kernel.cell` (`sgi-cell` module)

`cms_reset_timeout_action`

Specifies the action to be taken when `clconfd` does not acknowledge a reset request (determined by `cms_reset_timeout`).

Legal values:

- `0` proceeds as if the reset returned an error. This causes the node waiting for the reset acknowledgement to forcibly withdraw from the cluster, equivalent to a forced CXFS shutdown that occurs when a node loses quorum (default). If `clconfd` is still present and functioning properly, it will then restart the kernel `cms` daemon and the node will attempt to rejoin the cluster.
- `1` proceeds as if the reset succeeded. This clears all pending resets and continues (that is, fakes acknowledgment).

Caution: Setting this value is potentially dangerous.

- `2` panics the local node

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

`cms_trace_enable`

Enables or disables `cms` tracing on a non-`DEBUG` kernel and determines the number of trace entries allocated.

Range of values:

- Default: `0` (disables)
- Minimum: `0`
- Maximum: `1048576`

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

`cxfs_recovery_slowdown`

Slows down recovery by inserting delays (measured in ms).

Range of values:

- Default: `0`
- Minimum: `0`
- Maximum: `60000`

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

`cxfs_recovery_timeout_panic`

Specifies the action taken when a node with stalled recovery is discovered.

Legal values:

- `0` shuts down a node with stalled recovery (default)
- `1` panics a node with stalled recovery

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

`cxfs_recovery_timeout_period`

Specifies the time in seconds between recovery time-out polls.

Range of values:

- Default: `60`
- Minimum: `0` (disables recovery polls)
- Maximum: `3600`

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

`cxfs_recovery_timeout_stalled`

Specifies the time in seconds after which a node whose status is not changing is considered to have a stalled recovery.

Range of values:

- Default: `600`
- Minimum: `0` (disables time-out)
- Maximum: `3600`

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

`cxfs_recovery_timeout_start`

Specifies the time in seconds following a recovery before the recovery time-out monitoring begins.

Range of values:

- Default: `60`
- Minimum: `0`
- Maximum: `3600`

Location:

- IRIX: `/var/sysgen/mtune/cell`
- Linux: `kernel.cell` (`sgi-cell` module)

`cxfs_token_fault_tolerant`

Specifies whether to tolerate certain recoverable errors in the token subsystem. The least significant 4 bits are used in non-`DEBUG` kernels, the next 4 bits are used in `DEBUG` kernels (SGI internal only). In each group of 4 bits, the most significant bit determines whether the system will panic if an error condition is detected. The next bit determines whether part of the code path doing error detection and/or handling is enabled or disabled. The last 2 bits are interpreted as a debug level:

- 0 = No messages are printed
- 1 = Debug level 1
- 2 = Debug level 2
- 3 = Debug level 3

Figure F-1 displays the interpretation of the bits.

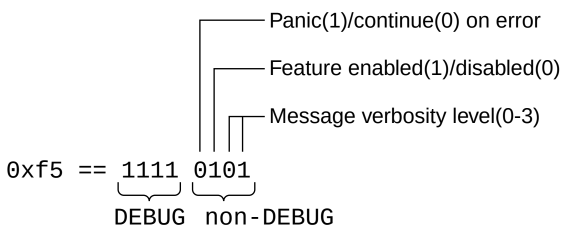

Panic(1)/continue(0) on error
Feature enabled(1)/disabled(0)
Message verbosity level(0-3)
0xf5 == 1111 0101
DEBUG
non-DEBUG

Figure F-1 Value Bit Interpretation

Range of values:

- Default: `0xf5` (debug level 1: prints only some messages in the non-`DEBUG` case, and prints all messages and panics in the `DEBUG` case)
- Minimum: `0` (debug level 0: disables all messages and diagnostics, `DEBUG` and non-`DEBUG`)
- Maximum: `0xff` (debug level 3: enables panics on error detection, maximum verbosity for diagnostic messages, `DEBUG` and non-`DEBUG`)

Location:

- IRIX: `/var/sysgen/mtune/cxfs`
- Linux: `fs.cxfs` (`sgi-cxfs` module)

`cxfs_token_track`

Specifies whether to verify that a client complies with the token locking hierarchy.

Range of values (for more information, see "`cxfs_token_fault_tolerant`" on page 488):

- Default: `0x0` (prints only some messages in the non-`DEBUG` case, and prints all messages and panics in the `DEBUG` case)
- Minimum: `0` (disables all messages and diagnostics, `DEBUG` and non-`DEBUG`)
- Maximum: `0xff` (enables panics on error detection, maximum verbosity for diagnostic messages, `DEBUG` and non-`DEBUG`)

Location:

- IRIX: `/var/sysgen/mtune/cxfs`
- Linux: `fs.cxfs` (`sgi-cxfs` module)

`cxfs_validate_objid`

Specifies the following:

- Server: specifies whether to check that an `objid` received from a client corresponds to an object of the expected type
- Client: specifies the level of reporting upon receipt of an `EBADOBJID` error from the server.

Range of values (for more information, see "`cxfs_token_fault_tolerant`" on page 488):

- Default: `0xf5` (prints only some messages in the non-`DEBUG` case, and prints all messages and panics in the `DEBUG` case)
- Minimum: `0` (disables all messages and diagnostics, `DEBUG` and non-`DEBUG`)
- Maximum: `0xff` (enables panics on error detection, maximum verbosity for diagnostic messages, `DEBUG` and non-`DEBUG`)

Location:

- IRIX: `/var/sysgen/mtune/cxfs`
- Linux: `fs.cxfs` (`sgi-cxfs` module)

`cxfs_verify_existence_token`

Specifies whether or not to verify that a client has the existence token before trying to obtain additional tokens.

Range of values (for more information, see "`cxfs_token_fault_tolerant`" on page 488):

- Default: `0xf5` (prints only some messages in the non-`DEBUG` case, and prints all messages and panics in the `DEBUG` case)
- Maximum: `0` (disables all messages and diagnostics, `DEBUG` and non-`DEBUG`)

- Minimum: `0xff` (enables panics on error detection, maximum verbosity for diagnostic messages, `DEBUG` and non-`DEBUG`)

Location:

- IRIX: `/var/sysgen/mtune/cxfs`
- Linux: `fs.cxfs` (`sgi-cxfs` module)

`cxfsd_sync_force`

Specifies a bitmask indicating `cxfsd` tasks that must be run synchronously, as opposed to the normal case where they are run asynchronously by threads from a `cxfsd` thread pool. The bits correspond to the opcodes in `cxfsd.h` (an SGI internal source file).

Range of values:

- Default: `0`
- Minimum: `0`
- Maximum: `0x7fffffff`

Location:

- IRIX: `/var/sysgen/mtune/cxfs`
- Linux: `fs.cxfs` (`sgi-cxfs` module)

Reference to `cmgr` Tasks

Caution: This appendix is included for convenience, but has not been updated to support CXFS 5.0. With the exception of a few administrative `cmgr`commands, the preferred CXFS configuration tools are `cxfs_admin` and the CXFS graphical user interface (GUI). See:

- Chapter 10, "Reference to GUI Tasks" on page 147
- Chapter 11, "Reference to `cxfs_admin` Tasks" on page 217

The following `cmgr` commands are still useful:

```
admin fence
admin nmi
admin ping
admin powerCycle
admin reset
start/stop cx_services
test connectivity
test serial
```

This appendix discusses the following:

- "`cmgr` Overview" on page 494
- "Initial Setup with the `cmgr` Command" on page 500
- "Set Configuration Defaults with `cmgr`" on page 512
- "Node Tasks with `cmgr`" on page 513
- "Cluster Tasks with `cmgr`" on page 533
- "Cluster Services Tasks with `cmgr`" on page 541
- "CXFS Filesystem Tasks with `cmgr`" on page 547
- "Switches and I/O Fencing Tasks with `cmgr`" on page 560

- "Script Example" on page 565
- "Creating a `cmgr` Script Automatically" on page 568
- "Troubleshooting `cmgr`" on page 572
- "Additional `cmgr` Examples" on page 572

For an overview of the tasks that must be performed to configure a cluster, see "Initial Setup with the `cmgr` Command" on page 500.

Tasks must be performed using a certain hierarchy. For example, to modify a partition ID, you must first identify the node name.

You can also use the `clconf_info` tool to view status. See Chapter 14, "Monitoring Status" on page 341.

Note: CXFS requires a license key to be installed on each server-capable node. If you install the software without properly installing the license key, you cannot use the `cmgr` command. For more information about licensing, see Chapter 5, "CXFS License Keys" on page 89.

For information about using the preferred `cxfs_admin` command rather than `cmgr`, see Chapter 11, "Reference to `cxfs_admin` Tasks" on page 217 and Appendix H, "Migration from `cmgr` to `cxfs_admin`" on page 595

cmgr Overview

To use the `cmgr` command, you must be logged in as `root` on a CXFS administration node. Then enter either of the following:

```
# /usr/cluster/bin/cmgr
```

or

```
# /usr/cluster/bin/cluster_mgr
```

After you have entered this command, you will see the following message and the command prompt (`cmgr>`):

```
Welcome to SGI Cluster Manager Command-Line Interface

cmgr>
```

For more information, see the `cmgr`(1M) man page][1]

Making Changes Safely

Do not make configuration changes on two different administration nodes in the pool simultaneously, or use the CXFS GUI, `cmgr`, and `xvm` commands simultaneously to make changes. You should run one instance of the `cmgr` command or the CXFS GUI on a single administration node in the pool when making changes at any given time. However, you can use any node in the pool when requesting status or configuration information.

Getting Help

After the command prompt displays, you can enter subcommands. At any time, you can enter `?` or `help` to bring up the `cmgr` help display.

Using Prompt Mode

The `-p` option to `cmgr` displays prompts for the required inputs of administration commands that define and modify CXFS components. You can run in prompt mode in either of the following ways:

- Specify a `-p` option on the command line:

  ```
  # cmgr -p
  ```

- Execute a `set prompting on` command after you have brought up `cmgr`, as in the following example:

  ```
  cmgr> set prompting on
  ```

  This method allows you to toggle in and out of prompt mode as you execute individual subcommands. To get out of prompt mode, enter the following:

  ```
  cmgr> set prompting off
  ```

The following shows an example of the questions that may be asked in prompting mode (the actual questions asked will vary depending upon your answers to previous questions):

[1] This man page is also accessible by `man cluster_mgr` for historical purposes.

```
cmgr> define node nodename
Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional] ?
Is this a FailSafe node <true|false> ?
Is this a CXFS node <true|false> ?
Operating System <IRIX|Linux32|Linux64|AIX|HPUX|Solaris|MacOSX|Windows> ?
Node Function <server_admin|client_admin|client_only> ?
Node ID ?[optional]
Partition ID ?[optional] (0)
Do you wish to define failure hierarchy[y/n]:
Reset type <powerCycle|reset|nmi> ? (powerCycle)
Do you wish to define system controller info[y/n]:
Sysctrl Type <msc|mmsc|l2|l1|bmc>? (msc)
Sysctrl Password[optional] ? ( )
Sysctrl Status <enabled|disabled> ?
Sysctrl Owner ?
Sysctrl Device ?
Sysctrl Owner Type <tty|network|ipmi> ? (tty)
Number of Network Interfaces ? (1)
NIC 1 - IP Address ?
```

For details about this task, see "Define a Node with cmgr" on page 513.

Completing Actions and Cancelling

When you are creating or modifying a component of a cluster, you can enter either of the following commands:

- `cancel`, which aborts the current mode and discards any changes you have made
- `done`, which executes the current definitions or modifications and returns to the `cmgr>` prompt

Using Script Files

You can execute a series of `cmgr` commands by using the `-f` option and specifying an input file:

`cmgr -f` *input_file*

Or, you could include the following as the first line of the file and then execute it as a `bash` script:

```
#!/usr/cluster/bin/cmgr -f
```

Each line of the file must be a valid `cmgr` command line, comment line (starting with #), or a blank line.

Note: You must include a `done` command line to finish a multilevel command and end the file with a `quit` command line.

If any line of the input file fails, `cmgr` will exit. You can choose to ignore the failure and continue the process by using the `-i` option with the `-f` option, as follows:

`cmgr -if` *input_file*

Or include it in the first line for a script:

```
#!/usr/cluster/bin/cmgr -if
```

Note: If you include `-i` when using a `cmgr` command line as the first line of the script, you must use this exact syntax (that is, `-if`).

For example, suppose the file /tmp/showme contains the following:

```
cxfs6# more /tmp/showme
show clusters
show nodes in cluster cxfs6-8
quit
```

You can execute the following command, which will yield the indicated output:

```
cxfs6# /usr/cluster/bin/cmgr -if /tmp/showme

1 Cluster(s) defined
        cxfs6-8


Cluster cxfs6-8 has following 3 machine(s)
        cxfs6
        cxfs7
        cxfs8
```

Or you could include the cmgr command line as the first line of the script, give it execute permission, and execute showme itself:

```
cxfs6# more /tmp/showme
#!/usr/cluster/bin/cmgr -if
#
show clusters
show nodes in cluster cxfs6-8
quit

cxfs6# /tmp/showme

1 Cluster(s) defined
        cxfs6-8



Cluster cxfs6-8 has following 3 machine(s)
        cxfs6
        cxfs7
        cxfs8
```

For an example of defining a complete cluster, see "Script Example" on page 565.

Invoking a Shell from within `cmgr`

To invoke a shell from within `cmgr`, enter the following:

```
cmgr> sh
cxfs6#
```

To exit the shell and to return to the `cmgr>` prompt, enter the following:

```
cxfs6# exit
cmgr>
```

Entering Subcommands on the Command Line

You can enter some `cmgr` subcommands directly from the command line using the following format:

`cmgr -c "`*subcommand*`"`

where *subcommand* can be any of the following with the appropriate operands:

- `admin`, which allows you to perform certain actions such as resetting a node
- `delete`, which deletes a cluster or a node
- `help`, which displays help information
- `show`, which displays information about the cluster or nodes
- `start`, which starts CXFS services and sets the configuration so that CXFS services will be automatically restarted upon reboot
- `stop`, which stops CXFS services and sets the configuration so that CXFS services are not restarted upon reboot
- `test`, which tests connectivity

For example, to display information about the cluster, enter the following:

```
# cmgr -c "show clusters"
1 Cluster(s) defined
      eagan
```

See the `cmgr` man page for more information.

Template Scripts

The /var/cluster/cmgr-templates directory contains template cmgr scripts that you can modify to configure the different components of your system.

Each template file contains lists of cmgr commands required to create a particular object, as well as comments describing each field. The template also provides default values for optional fields.

The /var/cluster/cmgr-templates directory contains the following templates to create a cluster and nodes:

- cmgr-create-cluster
- cmgr-create-node

To create a CXFS configuration, you can concatenate multiple templates into one file and execute the resulting script.

Note: If you concatenate information from multiple template scripts to prepare your cluster configuration, you must remove the quit at the end of each template script, except for the final quit. A cmgr script must have only one quit line.

For example, for a three-node configuration, you would concatenate three copies of the cmgr-create-node file and one copy of the cmgr-create-cluster file.

Initial Setup with the cmgr Command

Note: For the initial installation, SGI highly recommends that you use the GUI guided configuration tasks. See "Initial Setup with the CXFS GUI" on page 131.

To initially configure the cluster with the cmgr command, do the following:

1. Follow the directions in "Preliminary Cluster Configuration Steps" on page 128.
2. Define the nodes that are eligible to be part of the cluster. The hostname/IP-address pairings and priorities of the networks must be the same for each node in the cluster. See "Define a Node with cmgr" on page 513.

For large clusters, SGI recommends that you define only the first three CXFS administration nodes and then continue on to the next step; add the remaining nodes after you have a successful small cluster.

The following example sequence defines three nodes. (To use the default value for a prompt, press the `Enter` key. The `Enter` key is not shown in the examples in this guide.)

To define the first node, named `cxfs6`, enter the following:

Caution: It is critical that you enter the primary name for the first node defined in the pool.

```
cxfs6 # /usr/cluster/bin/cmgr -p
Welcome to SGI Cluster Manager Command-Line Interface

cmgr> define node cxfs6
Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional] ?
Is this a FailSafe node <true|false> ? false
Is this a CXFS node <true|false> ? true
Operating System  <IRIX|Linux32|Linux64|AIX|HPUX|Solaris|MacOSX|Windows> ? irix
Node Function <server_admin|client_admin|client_only> ? server_admin
Node ID[optional]?
Partition ID[optional] ? (0)
Do you wish to define failure hierarchy[y/n]:n
Reset type <powerCycle|reset|nmi>  ? (powerCycle)
Do you wish to define system controller info[y/n]:y
Sysctrl Type <msc|mmsc|l2|l1|bmc> ? (msc)
Sysctrl Password[optional] ? ( )
Sysctrl Status <enabled|disabled> ? enabled
Sysctrl Owner ? cxfs8
Sysctrl Device ? /dev/ttyd2
Sysctrl Owner Type <tty|network|ipmi> ? (tty)
Number of Network Interfaces ? (1)
NIC 1 - IP Address ? cxfs6
NIC 1 - Heartbeat HB (use network for heartbeats) <true|false> ? true
NIC 1 - (use network for control messages) <true|false> ? true

NIC 1 - Priority <1,2,...> 1
```

```
Successfully defined node cxfs6
```

To define the second node, named `cxfs7`, enter the following:

```
cmgr> define node cxfs7
Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional] ?
Is this a FailSafe node <true|false> ? false
Is this a CXFS node <true|false> ? true
Node Function <server_admin|client_admin|client_only> ? server_admin
Operating System <IRIX|Linux32|Linux64|AIX|HPUX|Solaris|MacOSX|Windows> ? irix
Node ID[optional] ?
Partition ID[optional] ? (0)
Do you wish to define failure hierarchy[y/n]:n
Reset type <powerCycle|reset|nmi>  ? (powerCycle)
Do you wish to define system controller info[y/n]:y
Sysctrl Type <msc|mmsc|l2|l1|bmc> ? (msc)
Sysctrl Password[optional] ? ( )
Sysctrl Status <enabled|disabled> ? enabled
Sysctrl Owner ? cxfs6
Sysctrl Device ? /dev/ttyd2
Sysctrl Owner Type <tty|network|ipmi> ? (tty)
Number of Network Interfaces ? (1)
NIC 1 - IP Address ? cxfs7
NIC 1 - Heartbeat HB (use network for heartbeats) <true|false> ? true
NIC 1 - (use network for control messages) <true|false> ? true

NIC 1 - Priority <1,2,...> 1

Successfully defined node cxfs7
```

To define the third node, named `cxfs8`, enter the following:

```
cmgr> define node cxfs8
Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional] ?
Is this a FailSafe node <true|false> ? false
Is this a CXFS node <true|false> ? true
```

```
Node Function <server_admin|client_admin|client_only> ? server_admin
Operating System <IRIX|Linux32|Linux64|AIX|HPUX|Solaris|MacOSX|Windows> ? irix
Node ID[optional] ?
Partition ID[optional] ? (0)
Do you wish to define failure hierarchy[y/n]:n
Reset type <powerCycle|reset|nmi>  ? (powerCycle)
Do you wish to define system controller info[y/n]:y
Sysctrl Type <msc|mmsc|l2|l1|bmc> ? (msc)
Sysctrl Password[optional] ? ( )
Sysctrl Status <enabled|disabled> ? enabled
Sysctrl Owner ? cxfs7
Sysctrl Device ? /dev/ttyd2
Sysctrl Owner Type <tty|network|ipmi> ? (tty)
Number of Network Interfaces ? (1)
NIC 1 - IP Address ? cxfs8
NIC 1 - Heartbeat HB (use network for heartbeats) <true|false> ? true
NIC 1 - (use network for control messages) <true|false> ? true

NIC 1 - Priority <1,2,...> 1

Successfully defined node cxfs8
```

You now have three nodes defined in the pool. To verify this, enter the following:

```
cmgr> show nodes in pool

3 Machine(s) defined
        cxfs6
        cxfs7
        cxfs8
```

To show the contents of node `cxfs6`, enter the following:

```
cmgr> show node cxfs6

Logical Machine Name: cxfs6
Hostname: cxfs6.example.com
Operating System: irix
Node Is FailSafe: false
Node Is CXFS: true
Node Function: server_admin
Nodeid: 13203
```

```
Partition id: 0
Reset type: powerCycle
System Controller: msc
System Controller status: enabled
System Controller owner: cxfs8
System Controller owner device: /dev/ttyd2
System Controller owner type: tty
ControlNet Ipaddr: cxfs6
ControlNet HB: true
ControlNet Control: true
ControlNet Priority: 1
```

3. Define the cluster and add the nodes to it. See "Define a Cluster with `cmgr`" on page 533.

   For example, to define a cluster named `cxfs6-8` and add the nodes that are already defined, enter the following:

```
cmgr> define cluster cxfs6-8
Enter commands, you may enter "done" or "cancel" at any time to exit

Is this a FailSafe cluster <true|false> false ?
Is this a CXFS cluster  <true|false> true ?
Cluster Notify Cmd [optional] ?
Cluster Notify Address [optional] ?
Cluster mode <normal|experimental>[optional]
Cluster ID ? 22

No nodes in cluster cxfs6-8

Add nodes to or remove nodes from cluster cxfs6-8
Enter "done" when completed or "cancel" to abort

cxfs6-8 ? add node cxfs6
cxfs6-8 ? add node cxfs7
cxfs6-8 ? add node cxfs8
cxfs6-8 ? done
Successfully defined cluster cxfs6-8

Added node <cxfs6> to cluster <cxfs6-8>
Added node <cxfs7> to cluster <cxfs6-8>
Added node <cxfs8> to cluster <cxfs6-8>
```

The fail action hierarchy is the set of instructions that determines which method is used in case of failure. If you set a hierarchy including fencing, you could define the switch at this point. For more information, see "Switches and I/O Fencing Tasks with `cmgr`" on page 560.

To define a list of private networks that can be used in case the highest priority network (consisting by default of the priority `1` NICs) fails, use the `add net` command; see "Define a Node with `cmgr`" on page 513.

For more information, see "Define a Cluster with `cmgr`" on page 533.

To verify the cluster and its contents, enter the following:

```
cmgr> show clusters

1 Cluster(s) defined
        cxfs6-8

cmgr> show cluster cxfs6-8
Cluster Name: cxfs6-8
Cluster Is FailSafe: false
Cluster Is CXFS: true
Cluster ID: 22
Cluster CX mode: normal

Cluster cxfs6-8 has following 3 machine(s)
        cxfs6
        cxfs7
        cxfs8

CXFS Failover Networks:
     default network 0.0.0.0, mask 0.0.0.0
```

For an example of this step using a script, see "Script Example" on page 565.

4. Start CXFS services for each node in the cluster by entering the following:

   ```
   start cx_services for cluster clustername
   ```

   For example:

   ```
   cmgr> start cx_services for cluster cxfs6-8
   ```

```
CXFS services have been activated in cluster cxfs6-8
```

This action starts CXFS services and sets the configuration so that CXFS services will be restarted automatically whenever a node reboots.

Note: If you stop CXFS services using either the GUI or `cmgr`, the automatic restart capability is turned off. You must start CXFS services again to reinstate the automatic restart capability.

To verify that CXFS services have been started in the cluster and there is a membership formed, you can use the following `cmgr` command:

`show status of cluster` *clustername*

For example:

```
cmgr> show status of cluster cxfs6-8

Cluster (cxfs6-8) is not configured for FailSafe


CXFS cluster state is ACTIVE.
```

You can also use the `clconf_info` command. For example:

```
cxfs6 # /usr/cluster/bin/clconf_info

Event at [2004-04-16 09:20:59]

Membership since Fri Apr 16 09:20:56 2004

____________ ______ ________ ______ ______
Node         NodeID Status   Age    CellID
____________ ______ ________ ______ ______
cxfs7         12812 up        0          1
cxfs6         13203 up        0          0
cxfs8         14033 up        0          2
____________ ______ ________ ______ ______
0 CXFS FileSystems
```

For more information, see "Display a Cluster with `cmgr`" on page 540.

5. Obtain a shell window for one of the CXFS administration nodes in the cluster and use the IRIX `fx`(1M) command or the Linux `parted`(8) command to create a volume header on the disk drive. For information, see the man pages, *IRIX Admin: Disks and Filesystems*, and *Linux Configuration and Operations Guide*.

6. Create the XVM logical volumes. In the shell window, use the `xvm` command line interface. For information, see the *XVM Volume Manager Administrator's Guide*.

7. Make the XFS filesystems. In the shell window, use the `mkfs` command. For information, see the *XVM Volume Manager Administrator's Guide* and *IRIX Admin: Disks and Filesystems*.

8. Define the filesystems by using the `define cxfs_filesystem` subcommand to `cmgr`. See "CXFS Filesystem Tasks with `cmgr`" on page 547.

   The following example shows two potential metadata servers for the `fs1` filesystem; if `cxfs6` (the preferred server, with rank 0) is not up when the cluster starts or later fails or is removed from the cluster, then `cxfs7` (rank 1) will be used. It also shows the filesystem being mounted by default on all nodes in the cluster (`Default Local Status enabled`) but explicitly not mounted on `cxfs8`.

   Note: Although the list of metadata servers for a given filesystem is ordered, it is impossible to predict which server will become the server during the boot-up cycle because of network latencies and other unpredictable delays.

   Do the following:

```
cmgr> define cxfs_filesystem fs1 in cluster cxfs6-8

(Enter "cancel" at any time to abort)

Device ? /dev/cxvm/d76lun0s0
Mount Point ? /mnts/fs1
Mount Options[optional] ?
Use Forced Unmount ? <true|false> ? false
Default Local Status <enabled|disabled> ? (enabled)

DEFINE CXFS FILESYSTEM OPTIONS

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:1

No current servers

Server Node ? cxfs6
Server Rank ? 0

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:1
Server Node ? cxfs7
Server Rank ? 1


        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:**5**

No disabled clients

Disabled Node ? **cxfs8**

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:**7**

Current settings for filesystem (fs1)

CXFS servers:
        Rank 0          Node cxfs6
        Rank 1          Node cxfs7

Default local status: enabled

No explicitly enabled clients

Explicitly disabled clients:
        Disabled Node: cxfs8

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:9
Successfully defined cxfs_filesystem fs1

cmgr> define cxfs_filesystem fs2 in cluster cxfs6-8

(Enter "cancel" at any time to abort)

Device ? /dev/cxvm/d77lun0s0
Mount Point ? /mnts/fs2
Mount Options[optional] ?
Use Forced Unmount ? <true|false> ? false
Default Local Status <enabled|disabled> ? (enabled)

DEFINE CXFS FILESYSTEM OPTIONS

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:1

Server Node ? cxfs8
Server Rank ? 0

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:7

Current settings for filesystem (fs2)

CXFS servers:
        Rank 0          Node cxfs8

Default local status: enabled

No explicitly enabled clients

No explicitly disabled clients

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:9
Successfully defined cxfs_filesystem fs2
```

To see the modified contents of cluster `cxfs6-8`, enter the following:

```
cmgr> show cxfs_filesystems in cluster cxfs6-8

fs1
fs2
```

9. Mount the filesystems on all nodes in the cluster by using the `admin cxfs_mount cxfs_filesystem` subcommand to `cmgr`. See "Mount a CXFS Filesystem with `cmgr`" on page 553. For example:

```
cmgr> admin cxfs_mount cxfs_filesystem fs1 in cluster cxfs6-8
cxfs_mount operation successful

cmgr> admin cxfs_mount cxfs_filesystem fs2 in cluster cxfs6-8
cxfs_mount operation successful
```

10. To quit out of `cmgr`, enter the following:

    ```
    cmgr> quit
    ```

Set Configuration Defaults with cmgr

You can set a default cluster and node to simplify the configuration process for the current session of `cmgr`. The default will then be used unless you explicitly specify a name. You can use the following commands to specify default values:

```
set cluster  clustername
set node hostname
```

clustername and *hostname* are logical names. Logical names cannot begin with an underscore (_) or include any whitespace characters, and can be at most 255 characters.

To view the current defaults, use the following:

```
show set defaults
```

For example:

```
cmgr> set cluster cxfs6-8
cmgr> set node cxfs6
cmgr> show set defaults
Default cluster set to: cxfs6-8

Default node set to: cxfs6
Default cdb set to: /var/cluster/cdb/cdb.db
Default resource_type is not set
Extra prompting is set off
```

Node Tasks with `cmgr`

This section tells you how to define, modify, delete, display, and reset a node using `cmgr`.

Note: The entire cluster status information is sent to each CXFS administration node each time a change is made to the cluster database; therefore, the more CXFS administration nodes in a configuration, the longer it will take.

Define a Node with `cmgr`

To define a node, use the following commands:

```
define node logical_hostname
    set hostname to hostname
    set is_failsafe to true|false
    set is_cxfs to true|false
    set operating_system to irix|linux32|linux64|aix|solaris|macosx|windows
    set node_function to server_admin|client_admin|client_only
    set nodeid to nodeID
    set partition_id to partitionID
    set hierarchy to [system][fence][reset][fencereset][shutdown]
    set reset_type to powerCycle|reset|nmi (for SGI hardware)
    set sysctrl_type to msc|mmsc|l2|l1|bmc (based on node hardware)
    set sysctrl_password to password
    set sysctrl_status to enabled|disabled
    set sysctrl_owner to node_sending_reset_command
    set sysctrl_device to port|IP_address_or_hostname_of_device
    set sysctrl_owner_type to tty|network|ipmi
    add nic IP_address_or_hostname (if DNS)
            set heartbeat to true|false
            set ctrl_msgs to true|false
            set priority to integer
    remove nic IP_address_or_hostname (if DNS)
    set weight to 0|1 (no longer needed)
```

Usage notes:

- *logical_hostname* is a simple hostname (such as `lilly`) or a fully qualified domain name (such as `lilly.example.com`) or an entirely different name (such as nodeA). Logical names cannot begin with an underscore (_) or include any whitespace characters, and can be at most 255 characters.

- hostname is the fully qualified hostname unless the simple hostname is resolved on all nodes. Use the ping to display the fully qualified hostname. Do not enter an IP address. The default for *hostname* is the value for *logical_hostname*; therefore, you must supply a value for this command if you use a value other than the hostname or an abbreviation of it for *logical_hostname*.
- If you are running just CXFS on this node, set is_cxfs to true and is_failsafe to false. If you are running both CXFS and FailSafe on this node in a coexecution cluster, set both values to true.
- operating_system can be set to irix, linux32, linux64, aix, solaris, macosx, or windows. Choose windows for Windows 2000, Windows 2003, or Windows XP. Choose linux64 when defining an x86_64 or ia64 architecture node. (Use the uname -i command to determine the architecture type.)

  Note: For support details, see the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

  If you specify aix, solaris, macosx or windows, the weight is assumed to be 0. If you try to specify incompatible values for operating_system and is_failsafe or weight, the define command will fail.
- node_function specifies the function of the node. Enter one of the following:
  - server_admin is an IRIX or SGI ProPack node on which you will execute cluster administration commands and that you also want to be a CXFS metadata server. (You will use the **Define a CXFS Filesystem** task to define the specific filesystem for which this node can be a metadata server.) Use this node function only if the node will be a metadata servers. You must have installed the cluster_admin product on this node.
  - client_admin is an IRIX node on which you will execute cluster administration commands but that you do not want to use as a CXFS metadata server. Use this node function only if the node will run FailSafe but you do not want it to be a metadata server. You must install the cluster_admin product on this node.
  - client_only, is a node that shares CXFS filesystems but on which you will not execute cluster administration commands and that will not be a CXFS

metadata server. Use this node function for all nodes other than those that will be metadata servers, or those that will run FailSafe without being a metadata server. You must install the `cxfs_client` product on this node. This node can run IRIX, SGI ProPack, Linux third-party, AIX, Solaris, Mac OS X, or Windows. (Nodes other than IRIX and SGI ProPack are **required** to be client-only nodes.)

AIX, Solaris, Mac OS X, and Windows nodes are automatically specified as `client-only`. You should specify `client-only` with `linux32`.

- `nodeid` is an integer in the range 1 through 32767 that is unique among the nodes in the pool. You must not change the node ID number after the node has been defined.
  - For administration nodes, this value is optional. If you do not specify a number for an administration node, CXFS will calculate an ID for you. The default ID is a 5-digit number based on the machine's serial number and other machine-specific information; it is not sequential.
  - For client-only nodes, you must specify a unique value.
- `partition_id` uniquely defines a partition in a partitioned Origin 3000 system, Altix 3000 series system, or Altix 4700 system. The `set partition_id` command is optional; if you do not have a partitioned Origin 3000 system, you can skip this command or enter 0.

Note: In an Origin 3000 system, use the `mkpart` command to determine this value:

- The `-n` option lists the partition ID (which is 0 if the system is not partitioned).
- The `-l` option lists the bricks in the various partitions (use *rack#.slot#* format in `cmgr`)

  For example (output truncated here for readability):

```
# mkpart -n
Partition id = 1
# mkpart -l
partition: 3 = brick: 003c10 003c13 003c16 003c21 003c24 003c29 ...
partition: 1 = brick: 001c10 001c13 001c16 001c21 001c24 001c29 ...
```

  You could enter one of the following for the **Partition ID** field:

```
1
001.10
```

To unset the partition ID, use a value of 0 or none.

On an Altix 3000, you can find the partition ID by reading the proc file. For example:

```
[root@linux root]# cat /proc/sgi_sn/partition_id
0
```

The 0 indicates that the system is not partitioned. If the system is partitioned, the number of partitions (such as 1, 2, etc.) is displayed.

- hierarchy defines the failpolicy hierarchy, which determines what happens to a failed node. You can specify up to three options. The second option will be completed only if the first option fails; the third option will be completed only if both the first and second options fail. Options must be separated by commas and no whitespace.

  The option choices are as follows:

  – system deletes all hierarchy information about the node from the database, causing the system defaults to be used. The system defaults are the same as entering reset,shutdown. This means that a reset will be performed on a node with a system controller; if the reset fails or if the node does not have a system controller, CXFS services will be forcibly shut down.

  – fence disables access to the SAN from the problem node. Fencing provides faster recovery of the CXFS kernel membership than reset.

  – fencereset performs a fence and then, if the node is successfully fenced, **also** performs an asynchronous reset of the node via a system controller; recovery begins without waiting for reset acknowledgement.

  Note: SGI recommends that a server-capable node include reset in its hierarchy (unless it is the only server-capable node in the cluster). See "Data Integrity Protection" on page 24.

  – reset performs a system reset via a system controller.

  – shutdown tells the other nodes in the cluster to wait for a period of time (long enough for the node to shut itself down) before reforming the CXFS kernel membership. (However, there is no notification that the node's shutdown has actually taken place.)

Caution: Because there is no notification that a shutdown has occurred, if you have a cluster with no tiebreaker, you must not use the `shutdown` setting for any server-capable node in order to avoid multiple clusters being formed. See "Shutdown" on page 55.

You cannot use `shutdown` on client nodes if you choose `dynamic` monitoring.

Note: If the failure hierarchy contains `reset` or `fencereset`, the reset might be performed before the system kernel core-dump can complete, resulting in an incomplete core-dump.

For a list of valid fail action sets, see "Data Integrity Protection" on page 24.

To perform a reset only if a fencing action fails, specify the following:

```
set hierarchy fence,reset
```

Note: If `shutdown` is not specified and the other actions fail, the node attempting to deliver the CXFS kernel membership will stall delivering the membership until either the failed node attempts to re-enter the cluster or the system administrator intervenes using `cms_intervene`. Objects held by the failed nodes stall until membership finally transitions and initiates recovery.

To perform a fence and an asynchronous reset, specify the following:

```
set hierarchy fencereset
```

To return to system defaults (`reset,shutdown`), specify the following:

```
set hierarchy system
```

- `reset_type` applies to SGI hardware and can be one of the following:
  - `powerCycle` shuts off power to the node and then restarts it
  - `reset` simulates the pressing of the reset button on the front of the machine
  - `nmi` (nonmaskable interrupt) performs a core-dump of the operating system kernel, which may be useful when debugging a faulty machine

- `sysctrl_type` is the system controller type based on the node hardware, as show in Table 11-1 on page 243.
- `sysctrl_password` is the password for the system controller port, not the node's `root` password or PROM password. On some nodes, the system administrator may not have set this password. If you wish to set or change the system controller password, consult the hardware manual for your node.
- `sysctrl_status` allows you to provide information about the system controller but temporarily disable reset by setting this value to `disabled` (meaning that CXFS cannot reset the node). To allow CXFS to reset the node, enter `enabled`. For nodes without system controllers, set this to `disabled`; see "Hardware and Software Requirements for Server-Capable Administration Nodes" on page 34.
- `sysctrl_device` is one of the following:
  - For systems with serial ports (`reset_comms=tty`), this is the name of the terminal port (TTY) on the owner node (the node issuing the reset). A serial cable connects the terminal port on the owner node to the system controller of the node being reset. `/dev/ttyd2` is the most commonly used port, except on Origin 300 and Origin 350 systems (where `/dev/ttyd4` is commonly used) and on Altix 350 systems (where `/dev/ttyIOC0` is commonly used).

    Note: Check the owner node's specific hardware configuration to verify which device to use.

  - For systems with network-attached L2 system controllers (`reset_comms=network`), this is the IP address or hostname of the L2 controller on the node being reset. For example, `reset_device=nodename-l2.company.com`. For systems with network-attached baseboard management controller (BMC) system controllers (`reset_comms=ipmi`), this is the IP address or hostname of the BMC controller on the node being reset. For example:

    ```
    reset_device=nodename-bmc.company.com
    ```

- `sysctrl_owner` is the name of the node that sends the reset command. If you use `tty` for `sysctrl_owner_type`, serial cables must physically connect the node being defined and the owner node through the system controller port. At run time, the node must be defined in the CXFS pool.
- `sysctrl_owner_type` is either `tty` for TTY serial devices or `network` for network reset (available for systems with L2 system controllers).

Note: If you are running in coexecution with FailSafe, the `network`and `ipmi` selections are not supported.

For example:

– For an Origin 3000 system:

```
Sysctrl Device? /dev/ttyd2
Sysctrl Owner Type? <tty|network|ipmi> tty
```

– For an Altix system with an integrated L2, such as a NUMAlink 4 R-brick, or SGI Altix 3000 Bx2 systems:

```
Sysctrl Device? nodename-l2.company.com
Sysctrl Owner Type? <tty|network|ipmi> network
```

– For an Altix 350:

```
Sysctrl Device? /dev/ttyIOC0
Sysctrl Owner Type? <tty|network|ipmi> tty
```

– For an Altix XE system with an integrated baseboard management controller (BMC) using the Intelligent Platform Management Interface (IPMI):

```
Sysctrl Device? nodename-bmc.company.com
 Sysctrl Owner Type?  ipmi
```

- `nic` is the IP address or hostname of the private network. (The hostname must be resolved in the `/etc/hosts` file.)

  There can be up to 8 network interfaces. The NIC number is not significant. Priority `1` is the highest priority. By default, only the priority `1` NICs will be used as the CXFS private network and they must be on the same subnet. However, you can use the `add net` command to configure the NICs of a given priority into a network; each network takes its priority from the set of NICs it contains. In this case, if the highest priority network fails, the second will be used, and so on; see "Define a Cluster with `cmgr`" on page 533.

Note: You cannot add a NIC or a network grouping while CXFS services are active (that is, when `start cx_services` has been executed); doing so can lead to cluster malfunction. If services have been started, they should be stopped with `stop cx_services`.

If you do not use the `add net` command to group the NICs into a set of networks, all NICs other than priority `1` are ignored.

SGI requires that this network be private; see "Private Network" on page 22.

For more information about using the hostname, see "Hostname Resolution and Network Configuration Rules" on page 103.

- `weight`, which is automatically set internally to either 0 or 1 to specify how many votes a particular CXFS administration node has in CXFS kernel membership decisions. This information is now set by the `Node Function` field and this command is no longer needed.

  Note: Although it is possible to use the `set weight` command to set a weight other than 0 or 1, SGI recommends that you do not do so. There is no need for additional weight.

For more information, see "CXFS Kernel Membership, Quorum, and Tiebreaker" on page 432, and "Define a Node with the GUI" on page 171.

In prompting mode, press the `Enter` key to use default information. (The `Enter` key is not shown in the examples.) For general information, see "Define a Node with the GUI" on page 171. Following is a summary of the prompts.

```
cmgr> define node logical_hostname
Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional] ? hostname
Is this a FailSafe node <true|false> ? true|false
Is this a CXFS node <true|false> ? truet
Operating System <IRIX|Linux32|Linux64|AIX|HPUX|Solaris|MacOSX|Windows> ?OS_type
Node Function <server_admin|client_admin|client_only> ? node_function
Node ID ?[optional] node_ID
Partition ID ?[optional] (0)partition_ID
Do you wish to define failure hierarchy[y/n]:y|n
Do you wish to define system controller info[y/n]:y|n
Reset type <powerCycle|reset|nmi> ? (powerCycle)
Do you wish to define system controller info[y/n]:y|n
Sysctrl Type <msc|mmsc|l2|l1|bmc>? (msc) type (based on node hardware)
Sysctrl Password[optional] ? ( )password
Sysctrl Status <enabled|disabled> ? enabled|disabled
Sysctrl Owner ? node_sending_reset_command
Sysctrl Device ? *port|IP_address_or_hostname_of_device*
Sysctrl Owner Type <tty|network|ipmi> ? (tty) **tty|network|ipmi**
Number of Network Interfaces ? (1) *number*
NIC 1 - IP Address ? *IP_address_or_hostname (if DNS)*
```

For example, in normal mode:

```
# /usr/cluster/bin/cmgr
Welcome to SGI Cluster Manager Command-Line Interface

cmgr> define node foo
Enter commands, you may enter "done" or "cancel" at any time to exit

? set is_failsafe to false
? set is_cxfs to true
? set operating_system to irix
? set node_function to server_admin
? set hierarchy to fencereset,reset
? add nic 111.11.11.111
Enter network interface commands, when finished enter "done" or "cancel"

NIC 1 - set heartbeat to true
NIC 1 - set ctrl_msgs to true
NIC 1 - set priority to 1
NIC 1 - done
? done
```

For example, in prompting mode:

```
# /usr/cluster/bin/cmgr -p
Welcome to SGI Cluster Manager Command-Line Interface

cmgr> define node foo
Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional] ?
Is this a FailSafe node <true|false> ? false
Is this a CXFS node <true|false> ? true
Operating System <IRIX|Linux32|Linux64|AIX|HPUX|Solaris|MacOSX|Windows> ? irix
Node Function <server_admin|client_admin|client_only> server_admin
Node ID[optional]?
```

```
Partition ID ? [optional] (0)
Do you wish to define failure hierarchy[y|n]:y
Hierarchy option 0 <System|FenceReset|Fence|Reset|Shutdown>[optional] ? fencereset
Hierarchy option 1 <System|FenceReset|Fence|Reset|Shutdown>[optional] ? reset
Hierarchy option 2 <System|FenceReset|Fence|Reset|Shutdown>[optional] ?
Reset type <powerCycle|reset|nmi>  ? (powerCycle)
Do you wish to define system controller info[y/n]:n
Number of Network Interfaces ? (1)
NIC 1 - IP Address ? 111.11.11.111
NIC 1 - Heartbeat HB (use network for heartbeats) <true|false> ? true
NIC 1 - (use network for control messages) <true|false> ? true

NIC 1 - Priority <1,2,...> 1
```

Following is an example of defining a Solaris node in prompting mode (because it is a Solaris node, no default ID is provided, and you are not asked to specify the node function because it must be `client_only`).

```
cmgr> define node solaris1
Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional] ?
Is this a FailSafe node <true|false> ? false
Is this a CXFS node <true|false> ? true
Operating System <IRIX|Linux32|Linux64|AIX|HPUX|Solaris|MacOSX|Windows> ? solaris
Node ID ? 7
Do you wish to define failure hierarchy[y/n]:y
Hierarchy option 0 <System|FenceReset|Fence|Reset|Shutdown>[optional] ? fence
Hierarchy option 1 <System|FenceReset|Fence|Reset|Shutdown>[optional] ?
Number of Network Interfaces ? (1)
NIC 1 - IP Address ? 163.154.18.172
```

Modify a Node with cmgr

To modify an existing node, use the following commands:

```
modify node logical_hostname
    set hostname to hostname
    set partition_id to partitionID
    set reset_type to powerCycle|reset|nmi
    set sysctrl_type to msc|mmsc|l2|l1|bmc (based on node hardware)
    set sysctrl_password to password
    set sysctrl_status to enabled|disabled
    set sysctrl_owner to node_sending_reset_command
    set sysctrl_device to port|IP_address_or_hostname_of_device
    set sysctrl_owner_type to tty|network|ipmi
    set is_failsafe to true|false
    set is_cxfs to true|false
    set weight to 0|1
    add nic IP_address_or_hostname (if DNS)
            set heartbeat to true|false
            set ctrl_msgs to true|false
            set priority to integer
    remove nic IP_address_or_hostname (if DNS)
    set hierarchy to [system] [fence][reset][fencereset][shutdown]
```

The commands are the same as those used to define a node. You can change any of the information you specified when defining a node except the node ID. For details about the commands, see "Define a Node with `cmgr`" on page 513.

Caution: Do not change the node ID number after the node has been defined.

You cannot add a NIC or a network grouping while CXFS services are active (that is, when `start cx_services` has been executed); doing so can lead to cluster malfunction. If services have been started, they should be stopped with `stop cx_services`.

You cannot modify the `operating_system` setting for a node; trying to do so will cause an error. If you have mistakenly specified the incorrect operating system, you must delete the node and define it again.

You cannot modify the node function. To change the node function, you must delete the node and redefine it (and reinstall software products, as needed); the node function for a Solaris or Windows node is always `client_only`.

Example of Partitioning

The following shows an example of partitioning an Origin 3000 system:

```
# cmgr
Welcome to SGI Cluster Manager Command-Line Interface
```

```
cmgr> modify node n_preston
Enter commands, when finished enter either "done" or "cancel"

n_preston ? set partition_id to 1
n_preston ? done

Successfully modified node n_preston
```

To perform this function with prompting, enter the following:

```
# cmgr -p
Welcome to SGI Cluster Manager Command-Line Interface

cmgr> modify node n_preston
Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional] ? (preston.example.com)
Is this a FailSafe node <true|false> ? (true)
Is this a CXFS node <true|false> ? (true)
Node ID[optional] ? (606)
Partition ID[optional] ? (0) 1
Do you wish to modify failure hierarchy[y/n]:n
Reset type <powerCycle|reset|nmi> ? (powerCycle)
Do you wish to modify system controller info[y/n]:n
Number of Network Interfaces? (1)
NIC 1 - IP Address ? (preston)
NIC 1 - Heartbeat HB (use network for heartbeats)  ? (true)
NIC 1 - (use network for control messages)  ? (true)
NIC 1 - Priority <1,2,...> ? (1)

Successfully modified node n_preston

cmgr> show node n_preston
Logical Machine Name: n_preston
Hostname: preston.example.com
Operating System: IRIX
Node Is FailSafe: true
Node Is CXFS: true
Node Function: client_admin
Nodeid: 606
Partition id: 1
```

```
Reset type: powerCycle
ControlNet Ipaddr: preston
ControlNet HB: true
ControlNet Control: true
ControlNet Priority: 1
```

To unset the partition ID, use a value of `0` or `none`.

Changing Failure Hierarchy

The following shows an example of changing the failure hierarchy for the node `perceval` from the system defaults to `fencereset,reset,shutdown` and back to the system defaults.

Caution: If you have a cluster with **an even number of server-capable nodes** and **no tiebreaker**: to avoid a split-brain scenario, you should not use the `shutdown` setting for any server-capable node. For a more detailed explanation, see "Shutdown" on page 55.

```
cmgr> modify node perceval
Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional] ? (perceval.example.com)
Is this a FailSafe node <true|false> ? (false)
Is this a CXFS node <true|false> ? (true)
Node ID[optional] ? (803)
Partition ID[optional] ? (0)
Do you wish to modify failure hierarchy[y/n]:y
Hierarchy option 0 <System|FenceReset|Fence|Reset|Shutdown>[optional] ?fencereset
Hierarchy option 1 <System|FenceReset|Fence|Reset|Shutdown>[optional] ?reset
Hierarchy option 2 <System|FenceReset|Fence|Reset|Shutdown>[optional] ?shutdown
Reset type <powerCycle|reset|nmi> ? (powerCycle)
Do you wish to modify system controller info[y/n]:n
Number of Network Interfaces ? (1)
NIC 1 - IP Address ? (163.154.18.173)

Successfully modified node perceval

cmgr> show node perceval
```

```
Logical Machine Name: perceval
Hostname: perceval.example.com
Operating System: IRIX
Node Is FailSafe: false
Node Is CXFS: true
Node Function: client_admin
Nodeid: 803
Node Failure Hierarchy is: FenceReset Reset Shutdown
Reset type: powerCycle
ControlNet Ipaddr: 163.154.18.173
ControlNet HB: true
ControlNet Control: true
ControlNet Priority: 1
```

To return to system defaults:

```
cmgr> modify node perceval

Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional] ? (perceval.example.com)
Is this a FailSafe node <true|false> ? (false)
Is this a CXFS node <true|false> ? (true)
Node ID[optional] ? (803)
Partition ID[optional] ? (0)
Do you wish to modify failure hierarchy[y/n]:y
Hierarchy option 0 <System|FenceReset|Fence|Reset|Shutdown>[optional] ?
(FenceReset) system
Reset type <powerCycle|reset|nmi> ? (powerCycle)
Do you wish to modify system controller info[y/n]:n
Number of Network Interfaces ? (1)
NIC 1 - IP Address ? (163.154.18.173)
NIC 1 - Heartbeat HB (use network for heartbeats) <true|false> ? (true)
NIC 1 - (use network for control messages) <true|false> ? (true)
NIC 1 - Priority <1,2,...> ? (1)

cmgr> show node perceval
Logical Machine Name: perceval
Hostname: perceval.example.com
Operating System: IRIX
Node Is FailSafe: false
Node Is CXFS: true
Node Function: client_admin
Nodeid: 803
Reset type: powerCycle|reset|nmi
ControlNet Ipaddr: 163.154.18.173
ControlNet HB: true
ControlNet Control: true
ControlNet Priority: 1
```

Note: When the system defaults are in place for failure hierarchy, no status is displayed with the `show` command.

Perform an NMI on a Node with `cmgr`

When CXFS daemons are running, you can perform a nonmaskable interrupt (NMI) on a node with the following command:

`admin nmi node` *nodename*

Note: The `nmi` option is not available for a `dev_type` of `ipmi` and `sysctrl_type` of `bmc`.

This command uses the CXFS daemons to perform an NMI on the specified node.

You can perform an NMI on a node in a cluster even when the CXFS daemons are not running by using the `standalone` option:

`admin nmi standalone node` *nodename*

This command does not go through the CXFS daemons.

If the node has not been defined in the cluster database, you can use the following command line:

`admin nmi dev_name` *port|IP_address_or_hostname_of_device* `of dev_type` *tty|network|ipmi* `with sysctrl_type` *msc|mmsc|l2|l1|*

For example:

```
admin nmi dev_name /dev/ttyIOC0 of dev_type tty  with sysctrl_type l2
```

If crsd does not see the response it expects within a certain time, it will issue another NMI, which invalidates the dump. This is determined by the setting of the following values in the cluster database (defaults shown):

```
CrsResetInterval = "20000"
CrsRetryInterval = "1000"
CrsResendTimeout = "10000"
CrsResendRetries = "2"
```

Note: These values can only be changed using advanced tools. If you feel that these values need to be changed, please contact your local SGI support provider.

Convert a Node to CXFS or FailSafe with cmgr

To convert an existing FailSafe node so that it also applies to CXFS, use the modify command to change the setting.

Note: You cannot turn off FailSafe or CXFS for a node if the respective high availability (HA) or CXFS services are active. You must first stop the services for the node.

For example, in normal mode:

```
cmgr> modify node cxfs6
Enter commands, when finished enter either "done" or "cancel"

cxfs6 ? set is_FailSafe to true
cxfs6 ? done

Successfully modified node cxfs6
```

For example, in prompting mode:

```
cmgr> modify node cxfs6
Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional] ? (cxfs6.example.com)
Is this a FailSafe node <true|false> ? (false) true
Is this a CXFS node <true|false> ? (true)
Node ID[optional] ? (13203)
```

```
Partition ID[optional] ? (0)
Do you wish to modify failure hierarchy[y/n]:n
Reset type <powerCycle|reset|nmi> ? (powerCycle)
Do you wish to modify system controller info[y/n]:n
Number of Network Interfaces ? (1)
NIC 1 - IP Address ? (163.154.18.172)
NIC 1 - Heartbeat HB (use network for heartbeats) <true|false> ? (true)
NIC 1 - (use network for control messages) <true|false> ? (true)
NIC 1 - Priority <1,2,...> ? (1)

Successfully modified node cxfs6
```

Delete a Node with `cmgr`

To delete a node, use the following command:

`delete node` *hostname*

You can delete a node only if the node is not currently part of a cluster. If a cluster currently contains the node, you must first modify that cluster to remove the node from it.

For example, suppose you had a cluster named `cxfs6-8` with the following configuration:

```
cmgr> show cluster cxfs6-8
Cluster Name: cxfs6-8
Cluster Is FailSafe: true
Cluster Is CXFS: true
Cluster ID: 20
Cluster HA mode: normal
Cluster CX mode: normal


Cluster cxfs6-8 has following 3 machine(s)
        cxfs6
        cxfs7
        cxfs8
```

To delete node `cxfs8`, you would do the following in prompting mode (assuming that CXFS services have been stopped on the node):

```
cmgr> modify cluster cxfs6-8
Enter commands, when finished enter either "done" or "cancel"

Is this a FailSafe cluster <true|false> ? (false)
Is this a CXFS cluster <true|false> ? (true)
Cluster Notify Cmd [optional] ?
Cluster Notify Address [optional] ?
Cluster CXFS mode <normal|experimental>[optional] ? (normal)
Cluster ID ? (20)

Current nodes in cluster cxfs6-8:
Node - 1: cxfs6
Node - 2: cxfs7
Node - 3: cxfs8

Add nodes to or remove nodes/networks from cluster
Enter "done" when completed or "cancel" to abort


cxfs6-8 ? remove node cxfs8
cxfs6-8 ? done
Successfully modified cluster cxfs6-8

cmgr> show cluster cxfs6-8
Cluster Name: cxfs6-8
Cluster Is FailSafe: false
Cluster Is CXFS: true
Cluster ID: 20
Cluster CX mode: normal


Cluster cxfs6-8 has following 2 machine(s)
        cxfs6
        cxfs7
```

To delete `cxfs8` from the pool, enter the following:

```
cmgr> delete node cxfs8
```

```
Deleted machine (cxfs6).
```

Display a Node with `cmgr`

After you have defined a node, you can display the node's parameters with the following command:

`show node` *hostname*

For example:

```
cmgr> show node cxfs6
Logical Machine Name: cxfs6
Hostname: cxfs6.example.com
Operating System: IRIX
Node Is FailSafe: false
Node Is CXFS: true
Node Function: server_admin
Nodeid: 13203
Reset type: powerCycle
ControlNet Ipaddr: 163.154.18.172
ControlNet HB: true
ControlNet Control: true
ControlNet Priority: 1
```

You can see a list of all of the nodes that have been defined with the following command:

```
show nodes in pool
```

For example:

```
cmgr> show nodes in pool

3 Machine(s) defined
        cxfs8
        cxfs6
        cxfs7
```

You can see a list of all of the nodes that have been defined for a specified cluster with the following command:

```
show nodes [in cluster clustername]
```

For example:

```
cmgr> show nodes in cluster cxfs6-8

Cluster cxfs6-8 has following 3 machine(s)
        cxfs6
        cxfs7
        cxfs8
```

If you have specified a default cluster, you do not need to specify a cluster when you use this command. For example:

```
cmgr> set cluster cxfs6-8
cmgr> show nodes

Cluster cxfs6-8 has following 3 machine(s)
        cxfs6
        cxfs7
        cxfs8
```

Test Node Connectivity with cmgr

You can use `cmgr` to test the network connectivity in a cluster. This test checks if the specified nodes can communicate with each other through each configured interface in the nodes. This test will not run if CXFS is running. This test requires that the `/etc/.rhosts` file be configured properly; see "SGI ProPack Modifications for CXFS Connectivity Diagnostics" on page 117.

Use the following command to test the network connectivity for the nodes in a cluster:

```
test connectivity in cluster clustername [on node nodename1 node nodename2 ...]
```

For example:

```
cmgr> test connectivity in cluster cxfs6-8 on node cxfs7
Status: Testing connectivity...
Status: Checking that the control IP_addresses are on the same networks
Status: Pinging address cxfs7 interface ef0 from node cxfs7 [cxfs7]
Notice: overall exit status:success, tests failed:0, total tests executed:1
```

This test yields an error message when it encounters its first error, indicating the node that did not respond. If you receive an error message after executing this test, verify that the network interface has been configured up, using the `ifconfig` command. For example (line breaks added here for readability):

```
# /usr/etc/ifconfig ef0
ef0: flags=405c43 <UP,BROADCAST,RUNNING,FILTMULTI,MULTICAST,CKSUM,DRVRLOCK,IPALIAS>
inet 128.162.89.39 netmask 0xffff0000 broadcast 128.162.255.255
```

The `UP` in the first line of output indicates that the interface is configured up.

If the network interface is configured up, verify that the network cables are connected properly and run the test again.

Test the Serial Connections with `cmgr`

See "System Reset Connection for Server-Capable Administration Nodes" on page 142.

Cluster Tasks with `cmgr`

This section tells you how to define, modify, delete, and display a cluster using `cmgr`. It also tells you how to start and stop CXFS services.

Define a Cluster with `cmgr`

When you define a cluster with `cmgr`, you define a cluster and add nodes to the cluster with the same command. For general information, see "Define a Cluster with the GUI" on page 186.

Use the following commands to define a cluster:

```
define cluster clustername
    set is_failsafe to true|false
    set is_cxfs to true|false
    set clusterid to clusterID
    set notify_cmd to notify_command
    set notify_addr to email_address
    set ha_mode to normal|experimental
    set cx_mode to normal|experimental
```

```
add node node1name
add node node2name
add net network network_address mask netmask
```

Usage notes:

- `cluster` is the logical name of the cluster. Logical names cannot begin with an underscore (_) or include any whitespace characters, and can be at most 255 characters. Clusters must have unique names
- If you are running just CXFS, set `is_cxfs` to `true` and `is_failsafe` to `false`. If you are running a coexecution cluster, set both values to `true`.
- `clusterid` is a unique number within your network in the range 1 through 255. The cluster ID is used by the operating system kernel to make sure that it does not accept cluster information from any other cluster that may be on the network. The kernel does not use the database for communication, so it requires the cluster ID in order to verify cluster communications. This information in the kernel cannot be changed after it has been initialized; therefore, you must not change a cluster ID after the cluster has been defined. Clusters must have unique IDs.
- `notify_cmd` is the command to be run whenever the status changes for a node or cluster.
- `notify_addr` is the address to be notified of cluster and node status changes. To specify multiple addresses, separate them with commas. CXFS will send e-mail to the addresses whenever the status changes for a node or cluster. If you do not specify an address, notification will not be sent. If you use the `notify_addr` command, you must specify the e-mail program (such as `/usr/sbin/Mail`) as the *notify_command*.
- The `set ha_mode` and `set cx_mode` commands should usually be set to `normal`. The `set cx_mode` command applies only to CXFS, and the `set ha_mode` command applies only to IRIS FailSafe.
- `net` defines a set of NICs into a network. If the highest priority network (beginning with NIC priority `1`) fails, the next highest will be used. All NICs within one network must be at the same priority. NICs of a given priority (such as priority `2`) cannot be in two separate `net` networks. Although the primary network must be private, the backup network may be public.

  If you do not specify a `net` list, the set of priority `1` NICs are used by default as the CXFS heartbeat network and there will be no failover to any other set of NICs.

The `network` parameter specifies an IP network address (such as `1.2.3.0`) and the `mask` parameter specifies the subnet mask (such as `255.255.255.0`) in decimal notation. The order in which you specify `network` or `mask` is not important.

Note: You cannot add a NIC or a network grouping while CXFS services are active (that is, when `start cx_services` has been executed); doing so can lead to cluster malfunction. If services have been started, they should be stopped with `stop cx_services`.

The following shows the commands with prompting:

```
cmgr> define cluster clustername
Enter commands, you may enter "done" or "cancel" at any time to exit

Is this a FailSafe cluster <true|false> ? true|false
Is this a CXFS cluster  <true|false> ? true|false
Cluster Notify Cmd [optional] ?
Cluster Notify Address [optional] ?
Cluster CXFS mode <normal|experimental>[optional] use_default_of_normal
Cluster ID ? cluster_ID
No nodes in cluster clustername

No networks in cluster clustername

Add nodes to or remove nodes/networks from cluster clustername
Enter "done" when completed or "cancel" to abort

clustername ? add node node1name
clustername ? add node node2name
...
clustername ? done
Successfully defined cluster clustername

Added node <node1name> to cluster <clustername>
Added node <node2name> to cluster <clustername>

...
```

You should set the cluster to the default `normal` mode. Setting the mode to `experimental` turns off heartbeating in the CXFS kernel membership code so that you can debug the cluster without causing node failures. For example, this can be useful if you just want to disconnect the network for a short time (provided that there is no other cluster networking activity, which will also detect a failure even if there is no heartbeating). However, you should never use `experimental` mode on a production cluster and should only use it if directed to by SGI customer support. SGI does not support the use of `experimental` by customers.

For example:

```
cmgr> define cluster cxfs6-8
Enter commands, you may enter "done" or "cancel" at any time to exit

Is this a FailSafe cluster <true|false> ? false
Is this a CXFS cluster  <true|false> ? true
Cluster Notify Cmd [optional] ?
Cluster Notify Address [optional] ?
Cluster CXFS mode <normal|experimental>[optional]
Cluster ID ? 20

No nodes in cluster cxfs6-8

No networks in cluster cxfs6-8

Add nodes to or remove nodes/networks from cluster cxfs6-8
Enter "done" when completed or "cancel" to abort

cxfs6-8 ? add node cxfs6
cxfs6-8 ? add node cxfs7
cxfs6-8 ? add node cxfs8
cxfs6-8 ? done
Successfully defined cluster cxfs6-8

Added node <cxfs6> to cluster <cxfs6-8>
Added node <cxfs7> to cluster <cxfs6-8>
Added node <cxfs8> to cluster <cxfs6-8>
```

To do this without prompting, enter the following:

```
cmgr> define cluster cxfs6-8
Enter commands, you may enter "done" or "cancel" at any time to exit

cluster cxfs6-8? set is_cxfs to true
cluster cxfs6-8? set clusterid to 20
cluster cxfs6-8? add node cxfs6
cluster cxfs6-8? add node cxfs7
cluster cxfs6-8? add node cxfs8
cluster cxfs6-8? done
Successfully defined cluster cxfs6-8
```

Modify a Cluster with `cmgr`

The commands are as follows:

```
modify cluster clustername
    set is_failsafe to true
    set is_cxfs to true
    set clusterid to clusterID
    set notify_cmd to command
    set notify_addr to email_address
    set ha_mode to normal|experimental
    set cx_mode to normal|experimental
    add node node1name
    add node node2name
    remove node node1name
    remove node node2name
    add net network network_address mask netmask
    remove net network network_address
```

These commands are the same as the `define cluster` commands. For more information, see "Define a Cluster with `cmgr`" on page 533, and "Define a Cluster with the GUI" on page 186.

Note: If you want to rename a cluster, you must delete it and then define a new cluster. If you have started CXFS services on the node, you must either reboot it or reuse the cluster ID number when renaming the cluster.

However, be aware that if you already have CXFS filesystems defined and then rename the cluster, CXFS will not be able to mount the filesystems. For more information, see "Cannot Mount Filesystems" on page 372.

Convert a Cluster to CXFS or FailSafe with cmgr

To convert a cluster, use the following commands:

```
modify cluster clustername
  set is_failsafe to true|false
  set is_cxfs to true|false
  set clusterid to clusterID
```

- `cluster` is the logical name of the cluster. Logical names cannot begin with an underscore (_) or include any whitespace characters, and can be at most 255 characters.
- If you are running just CXFS, set `is_cxfs` to `true` and `is_failsafe` to `false`. If you are running a coexecution cluster, set both values to `true`.
- `clusterid` is a unique number within your network in the range 1 through 255. The cluster ID is used by the operating system kernel to make sure that it does not accept cluster information from any other cluster that may be on the network. The kernel does not use the database for communication, so it requires the cluster ID in order to verify cluster communications. This information in the kernel cannot be changed after it has been initialized; therefore, you must not change a cluster ID after the cluster has been defined.

For example, to convert CXFS cluster `cxfs6-8` so that it also applies to FailSafe, enter the following:

```
cmgr> modify cluster cxfs6-8
Enter commands, when finished enter either "done" or "cancel"

cxfs6-8 ? set is_failsafe to true
```

The cluster must support all of the functionalities (FailSafe and/or CXFS) that are turned on for its nodes; that is, if your cluster is of type `CXFS`, then you cannot modify a node that is part of the cluster so that it is of type `FailSafe` or of type `CXFS and FailSafe`. However, the nodes do not have to support all the functionalities of the cluster; that is, you can have a node of type `CXFS` in a cluster of type `CXFS and FailSafe`.

Delete a Cluster with cmgr

To delete a cluster, use the following command:

`delete cluster` *clustername*

However, you cannot delete a cluster that contains nodes; you must first stop CXFS services on the nodes and then redefine the cluster so that it no longer contains the nodes.

For example, in normal mode:

```
cmgr> modify cluster cxfs6-8
Enter commands, when finished enter either "done" or "cancel"

cxfs6-8 ? remove node cxfs6
cxfs6-8 ? remove node cxfs7
cxfs6-8 ? remove node cxfs8
cxfs6-8 ? done
Successfully modified cluster cxfs6-8

cmgr> delete cluster cxfs6-8

cmgr> show clusters

cmgr>
```

For example, in prompting mode:

```
cmgr> modify cluster cxfs6-8
Enter commands, you may enter "done" or "cancel" at any time to exit

Cluster Notify Cmd [optional] ?
Cluster Notify Address [optional] ?
Cluster mode <normal|experimental>[optional] ? (normal)
Cluster ID ? (55)

Current nodes in cluster cxfs6-8:
Node - 1: cxfs6
Node - 2: cxfs7
Node - 3: cxfs8

Add nodes to or remove nodes from cluster cxfs6-8
Enter "done" when completed or "cancel" to abort

cxfs6-8 ? remove node cxfs6
cxfs6-8 ? remove node cxfs7
```

```
cxfs6-8 ? **remove node cxfs8**
cxfs6-8 ? **done**
Successfully modified cluster cxfs6-8

cmgr> **delete cluster cxfs6-8**

cmgr> **show clusters**

cmgr>
```

Display a Cluster with cmgr

To display the clusters and their contents, use the following commands:

```
show clusters
show cluster clustername
```

For example, the following output shows that cluster `mycluster` has six nodes and two private networks, permitting network failover:

```
cmgr> show cluster mycluster
Cluster Name: mycluster
Cluster Is FailSafe: false
Cluster Is CXFS: true
Cluster ID: 1
Cluster CX mode: normal


Cluster mycluster has following 6 machine(s)
        nodeA
 nodeB
 nodeC
 nodeD
 nodeE
 nodeF


CXFS Failover Networks:
    network 192.168.0.0, mask 255.255.255.0
    network 134.14.54.0, mask 255.255.255.0
```

The multiple networks listed indicates that if the higher priority network should fail, the next priority network will be used. However, the order in which the networks are listed in this output is not an indication of priority. To determine the priority of the networks, you must look at the NIC priorities in the node definition.

Cluster Services Tasks with `cmgr`

The following tasks tell you how to start and stop CXFS services and set log levels.

Start CXFS Services with `cmgr`

To start CXFS services, and set the configuration to automatically restart CXFS services whenever the system is rebooted, use one of the following commands:

```
start cx_services [on node hostname ] for cluster clustername
```

For example, to start CXFS services on all nodes in the cluster:

```
cmgr> start cx_services for cluster cxfs6-8
```

Stop CXFS Services with `cmgr`

When CXFS services are stopped on a node, filesystems are automatically unmounted from that node.

To stop CXFS services on a specified node or cluster, and prevent CXFS services from being restarted by a reboot, use the following command:

```
stop cx_services [on node hostname]for cluster clustername [force]
```

Note: This procedure is only recommended as needed for CXFS administration node because it updates the cluster database and is therefore intrusive to other nodes. When shutting down a CXFS client–only node, do not administratively stop the CXFS services on the node Rather, let the CXFS services stop by themselves when the client-only node is shut down.

For example:

```
cmgr> stop cx_services on node cxfs6 for cluster cxfs6-8

CXFS services have been deactivated on node cxfs6 (cluster cxfs6-8)

cmgr> stop cx_services for cluster cxfs6-8
```

After you have stopped CXFS services in a node, the node is no longer an active member of the cluster.

Caution: If you stop CXFS services, the node will be marked as `INACTIVE` and it will therefore not rejoin the cluster after a reboot. To allow a node to rejoin the cluster, you must restart CXFS services using `cmgr` or the GUI.

Set the Tiebreaker Node with `cmgr`

A *CXFS tiebreaker node* determines whether a CXFS kernel membership quorum is maintained when exactly half of the server-capable nodes can communicate with each other. There is no default CXFS tiebreaker.

Caution: If one of the server-capable nodes is the CXFS tiebreaker in a two server-capable cluster, failure of that node or stopping the CXFS services on that node will result in a cluster-wide forced shutdown. Therefore SGI recommends that you use client-only nodes as tiebreakers so that either server could fail but the cluster would remain operational via the other server.

The reset capability or I/O fencing with switches is **mandatory** to ensure data integrity for all nodes. Clusters should have an odd number of server-capable nodes. If you have an even number of server-capable administration nodes, define a CXFS tiebreaker node. SGI recommends making a client-only node the tiebreaker. (See "CXFS Recovery Issues in a Cluster with Only Two Server-Capable Administration Nodes " on page 443.)

To set the CXFS tiebreaker node, use the `modify` command as follows:

```
modify cx_parameters
[on node nodename] in cluster clustername
set tie_breaker to hostname
```

To unset the CXFS tiebreaker node, use the following command:

```
set tie_breaker to none
```

For example, in normal mode:

```
cmgr> modify cx_parameters in cluster cxfs6-8
Enter commands, when finished enter either "done" or "cancel"

cxfs6-8 ? set tie_breaker to cxfs8
cxfs6-8 ? done
Successfully modified cx_parameters
```

For example, in prompting mode:

```
cmgr> modify cx_parameters in cluster cxfs6-8

(Enter "cancel" at any time to abort)

Tie Breaker Node ? (cxfs7) cxfs8
Successfully modified cx_parameters

cmgr> show cx_parameters in cluster cxfs6-8

_CX_TIE_BREAKER=cxfs8
```

Set Log Configuration with cmgr

For general information about CXFS logs, see "Set Log Configuration with the GUI" on page 190.

Display Log Group Definitions with cmgr

Use the following command to view the log group definitions:

```
show log_groups
```

This command shows all of the log groups currently defined, with the log group name, the logging levels, and the log files.

Use the following command to see messages logged by a specific daemon on a specific node:

```
show log_group LogGroupName [on node Nodename]
```

To exit from the message display, enter `Cntrl-C`.

Configure Log Groups with `cmgr`

You can configure a log group with the following command:

```
define log_group log_group on node adminhostname [in cluster clustername]
  set log_level to log_level
  add log_file log_file
  remove log_file log_file
```

Usage notes:

- `log_group` can be one of the following:

  ```
  clconfd
  cli
  crsd
  diags
  ```

- `log_level` can have one of the following values:
  - `0` gives no logging
  - `1` logs notifications of critical errors and normal operation (these messages are also logged to the `SYSLOG` file)
  - `2` logs `Minimal` notifications plus warnings
  - `5` through `7` log increasingly more detailed notifications
  - `10` through `19` log increasingly more debug information, including data structures
- *log_file*

Caution: Do not change the names of the log files. If you change the names, errors can occur.

For example, to define log group `cli` on node `cxfs6` with a log level of `5`:

```
cmgr> define log_group cli on node cxfs6 in cluster cxfs6-8

(Enter "cancel" at any time to abort)

Log Level ? (11) 5

CREATE LOG FILE OPTIONS

        1) Add Log File.
        2) Remove Log File.
        3) Show Current Log Files.
        4) Cancel. (Aborts command)
        5) Done. (Exits and runs command)

Enter option:5
Successfully defined log group cli
```

Modify Log Groups with `cmgr`

Use the following command to modify a log group:

`modify log_group` *log_group_name* `on node` *hostname* `[in cluster` *clustername*`]`

You modify a log group using the same commands you use to define a log group.

For example, to change the log level of `cli` to be `10`, enter the following:

```
cmgr> modify log_group cli on node cxfs6 in cluster cxfs6-8

(Enter "cancel" at any time to abort)

Log Level ? (2) 10

MODIFY LOG FILE OPTIONS

        1) Add Log File.
        2) Remove Log File.
        3) Show Current Log Files.
        4) Cancel. (Aborts command)
        5) Done. (Exits and runs command)
```

```
Enter option:**5**
Successfully modified log group cli
```

Revoke Membership of the Local Node with cmgr

To revoke CXFS kernel membership for the local node, such as before the forced CXFS shutdown, enter the following on the local node:

```
admin cxfs_stop
```

This command will be considered as a node failure by the rest of the cluster. The rest of the cluster may then fail due to a loss of CXFS kernel membership quorum, or it may decide to reset the failed node. To avoid the reset, you can modify the node definition to disable the system controller status.

Allow Membership of the Local Node with cmgr

Allowing CXFS kernel membership for the local node permits the node to reapply for CXFS kernel membership. You must actively allow CXFS kernel membership for the local node in the following situations:

- After a manual revocation as in "Revoke Membership of the Local Node with `cmgr`" on page 546.
- When instructed to by an error message on the console or in `/var/adm/SYSLOG`.
- After a kernel-triggered revocation. This situation is indicated by the following message in `/var/adm/SYSLOG`:

  ```
  Membership lost - withdrawing from cluster
  ```

To allow CXFS kernel membership for the local node, use the following command:

```
cmgr> admin cxfs_start
```

See also "Shutdown of the Database and CXFS" on page 296.

CXFS Filesystem Tasks with cmgr

This section tells you how to define a filesystem, specify the nodes on which it may or may not be mounted (the *enabled* or *disabled* nodes), and perform mounts and unmounts.

A given filesystem can be mounted on a given node when the following things are true:

- One of the following is true for the node:
  - The default local status is enabled and the node is not in the filesystem's list of explicitly disabled nodes
  - The default local status is disabled and the node is in the filesystem's list of explicitly enabled nodes

  See "Define a CXFS Filesystem with `cmgr`" on page 547.
- The global status of the filesystem is enabled. See "Mount a CXFS Filesystem with `cmgr`" on page 553.

Define a CXFS Filesystem with cmgr

Use the following commands to define a filesystem and the nodes on which it may be mounted:

```
define cxfs_filesystem logical_filesystem_name [in cluster clustername]
   set device_name to devicename
   set mount_point to mountpoint
   set mount_options to mount_options
   set force to true|false
   set dflt_local_status to enabled|disabled
   add cxfs_server admin_nodename
      set rank to 0|1|2|...
   add enabled_node nodename
   add disabled_node nodename
   remove cxfs_server admin_nodename
   remove enabled_node nodename
   remove disabled_node nodename
```

Usage notes:

- Relocation is disabled by default. Recovery and relocation are supported only when using standby nodes. Therefore, you should only define multiple metadata servers for a given filesystem if you are using the standby node model. See "Relocation" on page 25.
- The list of potential metadata servers for any given filesystem must all run the same operating system type.
- `cxfs_filesystem` can be any logical name. Logical names cannot begin with an underscore (_) or include any whitespace characters, and can be at most 255 characters.

Note: Within the GUI, the default is to use the last portion of the device name; for example, for a device name of `/dev/cxvm/d76lun0s0`, the GUI will automatically supply a logical filesystem name of `d76lun0s0`. The GUI will accept other logical names defined with `cmgr` but the GUI will not allow you to modify a logical name; you must use `cmgr` to modify the logical name.

- `device_name` is the device name of an XVM volume that will be shared among all nodes in the CXFS cluster. The name must begin with `/dev/cxvm/`. For more information, see *XVM Volume Manager Administrator's Guide*.
- `mount_point` is a directory to which the specified XVM volume will be attached. This directory name must begin with a slash (`/`). For more information, see the `mount` man page.
- `mount_options` are options that are passed to the `mount` command and are used to control access to the specified XVM volume. For a list of the available options, see the `fstab` man page.
- `force` controls what action CXFS takes if there are processes that have open files or current directories in the filesystem(s) that are to be unmounted. If set to `true`, then the processes will be killed and the unmount will occur. If set to `false`, the processes will not be killed and the filesystem will not be unmounted. The force option off (set to `true`) by default.
- `dflt_local_status` defines whether the filesystem can be mounted on all unspecified nodes or cannot be mounted on any unspecified nodes. You can then use the `add enabled_node` or `add disabled_node` commands as necessary to explicitly specify the nodes that differ from the default. There are multiple combinations that can have the same result.

For example, suppose you had a cluster with 10 nodes (`node1` through `node10`). You could use the following methods:

– If you want the filesystem to be mounted on all nodes, and want it to be mounted on any nodes that are later added to the cluster, you would specify:

```
set dflt_local_status to enabled
```

– If you want the filesystem to be mounted on all nodes except `node5`, and want it to be mounted on any nodes that are later added to the cluster, you would specify:

```
set dflt_local_status to enabled
add disabled_node cxfs5
```

– If you want the filesystem to be mounted on all nodes except `node5`, and you also do **not** want it to be mounted on any nodes that are later added to the cluster, you would specify:

```
set dflt_local_status to disabled
add enabled_node cxfs1
add enabled_node cxfs2
add enabled_node cxfs3
add enabled_node cxfs4
add enabled_node cxfs6
add enabled_node cxfs7
add enabled_node cxfs8
add enabled_node cxfs9
add enabled_node cxfs10
```

– If you want the filesystem to be mounted on `node5` through `node10` and on any future nodes, you could specify:

```
set dflt_local_status to enabled
add disabled_node cxfs1
add disabled_node cxfs2
add disabled_node cxfs3
add disabled_node cxfs4
```

To actually mount the filesystem on the enabled nodes, see "Mount a CXFS Filesystem with `cmgr`" on page 553.

- `cxfs_server` adds or removes the specified CXFS administration node name to the list of potential metadata servers.

Note: After a filesystem has been defined in CXFS, running `mkfs` on it will cause errors to appear in the system log file. To avoid these errors, run `mkfs` before defining the filesystem in CXFS, or delete the CXFS filesystem before running `mkfs`. See "Delete a CXFS Filesystem with `cmgr`" on page 559.

The following examples shows two potential metadata servers for the `fs1` filesystem; if `cxfs6` (the preferred server, with rank 0) is not up when the cluster starts or later fails or is removed from the cluster, then `cxfs7` (rank 1) will be used. The filesystem is mounted on all nodes.

Note: Although the list of metadata servers for a given filesystem is ordered, it is impossible to predict which server will become the server during the boot-up cycle because of network latencies and other unpredictable delays.

For example, in normal mode:

```
cmgr> define cxfs_filesystem fs1 in cluster cxfs6-8

cxfs_filesystem fs1 ? set device_name to /dev/cxvm/d76lun0s0
cxfs_filesystem fs1 ? set mount_point to /mnts/fs1
cxfs_filesystem fs1 ? set force to false
cxfs_filesystem fs1 ? add cxfs_server cxfs6
Enter CXFS server parameters, when finished enter "done" or "cancel"

CXFS server - cxfs6 ? set rank to 0
CXFS server - cxfs6 ? done
cxfs_filesystem fs1 ? add cxfs_server cxfs7
Enter CXFS server parameters, when finished enter "done" or "cancel"

CXFS server - cxfs7 ? set rank to 1
CXFS server - cxfs7 ? done
cxfs_filesystem fs1 ? set dflt_local_status to enabled
cxfs_filesystem fs1 ? done
Successfully defined cxfs_filesystem fs1

cmgr> define cxfs_filesystem fs2 in cluster cxfs6-8

cxfs_filesystem fs2 ? set device_name to /dev/cxvm/d76lun0s1
cxfs_filesystem fs2 ? set mount_point to /mnts/fs2
cxfs_filesystem fs2 ? set force to false
cxfs_filesystem fs2 ? add cxfs_server cxfs8
Enter CXFS server parameters, when finished enter "done" or "cancel"

CXFS server - cxfs8 ? set rank to 0
CXFS server - cxfs8 ? done
cxfs_filesystem fs2 ? set dflt_local_status to enabled
cxfs_filesystem fs2 ? done
Successfully defined cxfs_filesystem fs2
```

For example, in prompting mode:

```
cmgr> define cxfs_filesystem fs1 in cluster cxfs6-8

(Enter "cancel" at any time to abort)

Device ? /dev/cxvm/d76lun0s0
Mount Point ? /mnts/fs1
Mount Options[optional] ?
Use Forced Unmount ? <true|false> ? false
Default Local Status <enabled|disabled> ? (enabled)

DEFINE CXFS FILESYSTEM OPTIONS

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:1

No current servers

Server Node ? cxfs6
Server Rank ? 0

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:1
Server Node ? cxfs7
Server Rank ? 1


        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:9
Successfully defined cxfs_filesystem fs1

cmgr> define cxfs_filesystem fs2 in cluster cxfs6-8

(Enter "cancel" at any time to abort)

Device ? /dev/cxvm/d77lun0s1
Mount Point ? /mnts/fs2
Mount Options[optional] ?
Use Forced Unmount ? <true|false> ? false
Default Local Status <enabled|disabled> ? (enabled)
```

```
DEFINE CXFS FILESYSTEM OPTIONS

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:1

Server Node ? cxfs8
Server Rank ? 0

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:9
Successfully defined cxfs_filesystem fs2
```

Mount a CXFS Filesystem with `cmgr`

To mount a filesystem on the enabled nodes, enter the following:

```
admin cxfs_mount cxfs_filesystem logical_filesystem_name [on node nodename] [in cluster clustername]
```

This command enables the *global status* for a filesystem; if you specify the *nodename*, it enables the *local status*. (The global status is only affected if a node name is not specified.) For a filesystem to mount on a given node, both global and local status must be enabled; see "CXFS Filesystem Tasks with `cmgr`" on page 547.

Nodes must first be enabled by using the `define cxfs_filesystem` and `modify cxfs_filesystem` commands; see "Define a CXFS Filesystem with `cmgr`" on page 547, and "Modify a CXFS Filesystem with `cmgr`" on page 555.

For example, to activate the `f1` filesystem by setting the global status to `enabled`, enter the following:

```
cmgr> admin cxfs_mount cxfs_filesystem fs1 in cluster cxfs6-8
```

The filesystem will then be mounted on all the nodes that have a local status of `enabled` for this filesystem.

To change the local status to `enabled`, enter the following:

```
cmgr> admin cxfs_mount cxfs_filesystem fs1 on node cxfs7 in cluster cxfs6-8
```

If the filesystem's global status is `disabled`, nothing changes. If the filesystem's global status is `enabled`, the node will mount the filesystem as the result of the change of its local status.

Note: If CXFS services are not active, mounting a filesystem will not completely succeed. The filesystem will be marked as ready to be mounted but the filesystem will not actually be mounted until you have started CXFS services. For more information, see "Start CXFS Services with `cmgr`" on page 541.

Unmount a CXFS Filesystem with `cmgr`

To unmount a filesystem, enter the following:

```
admin cxfs_unmount cxfs_filesystem filesystemname [on node nodename] [in cluster clustername]
```

Unlike the `modify cxfs_filesystem` command, this command can be run on an active filesystem.

For example, to deactivate the `f1` filesystem by setting the global status to `disabled`, enter the following:

```
cmgr> admin cxfs_unmount cxfs_filesystem fs1 in cluster cxfs6-8
```

The filesystem will then be unmounted on all the nodes that have a local status of `enabled` for this filesystem.

To change the local status to `disabled`, enter the following:

```
cmgr> admin cxfs_unmount cxfs_filesystem fs1 on node cxfs7 in cluster cxfs6-8
```

If the filesystem's global status is `disabled`, nothing changes. If the filesystem's global status is `enabled`, the node will unmount the filesystem as the result of the change of its local status.

Modify a CXFS Filesystem with `cmgr`

Note: You cannot modify a mounted filesystem.

Use the following commands to modify a filesystem:

```
modify cxfs_filesystem logical_filesystem_name [in cluster clustername]
   set device_name to devicename
   set mount_point to mountpoint
   set mount_options to options
   set force to true|false
   set dflt_local_status to enabled|disabled
   add cxfs_server servername
     set rank to 0|1|2|...
   modify cxfs_server servername
     set rank to 0|1|2|...
   add enabled_node nodename
   add disabled_node nodename
   remove cxfs_server nodename
   remove enabled_node nodename
   remove disabled_node nodename
```

These are the same commands used to define a filesystem; for more information, see "Define a CXFS Filesystem with `cmgr`" on page 547.

For example, in normal mode:

```
cmgr> show cxfs_filesystem fs1 in cluster cxfs6-8

Name: fs1
```

```
Device: /dev/cxvm/d76lun0s0
Mount Point: /mnts/fs1
Forced Unmount: false
Global Status: disabled
Default Local Status: enabled

Server Name: cxfs6
       Rank: 0
Server Name: cxfs7
       Rank: 1
Disabled Client: cxfs8

cmgr> modify cxfs_filesystem fs1 in cluster cxfs6-8
Enter commands, when finished enter either "done" or "cancel"

cxfs_filesystem fs3 ? modify cxfs_server cxfs6
Enter CXFS server parameters, when finished enter "done" or "cancel"

Current CXFS server cxfs6 parameters:
        rank : 0
CXFS server - cxfs6 ? set rank to 2
CXFS server - cxfs6 ? done
cxfs_filesystem fs1 ? done

Successfully modified cxfs_filesystem fs1
cmgr> show cxfs_filesystem fs1 in cluster cxfs6-8

Name: fs1
Device: /dev/cxvm/d76lun0s0
Mount Point: /mnts/fs1
Forced Unmount: false
Global Status: disabled
Default Local Status: enabled

Server Name: cxfs6
       Rank: 2
Server Name: cxfs7
       Rank: 1
Disabled Client: cxfs8
```

In prompting mode:

```
cmgr> show cxfs_filesystem fs1 in cluster cxfs6-8

Name: fs1
Device: /dev/cxvm/d76lun0s0
Mount Point: /mnts/fs1
Forced Unmount: false
Global Status: disabled
Default Local Status: enabled

Server Name: cxfs6
       Rank: 0
Server Name: cxfs7
       Rank: 1
Disabled Client: cxfs8

cmgr> modify cxfs_filesystem fs1 in cluster cxfs6-8

(Enter "cancel" at any time to abort)

Device ? (/dev/cxvm/d76lun0s0)
Mount Point ? (/mnts/fs1)
Mount Options[optional] ?
Use Forced Unmount ? <true|false>  ? (false)
Default Local Status <enabled|disabled> ? (enabled)

MODIFY CXFS FILESYSTEM OPTIONS

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:0
```

```
Current servers:
CXFS Server 1 - Rank: 0         Node: cxfs6
CXFS Server 2 - Rank: 1         Node: cxfs7

Server Node ? cxfs6
Server Rank ? (0) 2

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:7

Current settings for filesystem (fs1)

CXFS servers:
        Rank 2          Node cxfs6
        Rank 1          Node cxfs7

Default local status: enabled

No explicitly enabled clients

Explicitly disabled clients:
        Disabled Node: cxfs8

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:9
Successfully modified cxfs_filesystem fs3
```

Relocate the Metadata Server for a Filesystem with `cmgr`

If relocation is explicitly enabled in the kernel with the `cxfs_relocation_ok` systune (see "Relocation" on page 25), you can relocate a metadata server to another node using the following command if the filesystem must be mounted on the system that is running `cmgr`:

`admin cxfs_relocate cxfs_filesystem` *filesystem_name* `to node` *nodename* `[in cluster` *clustername*`]`

Note: This function is only available on a live system.

To relocate the metadata server from `cxfs6` to `cxfs7` for `fs1` in cluster `cxfs6-8`, enter the following:

```
cmgr> admin cxfs_relocate cxfs_filesystem fs1 to node cxfs7 in cluster cxfs6-8
```

CXFS kernel membership is not affected by relocation. However, users may experience a degradation in filesystem performance while the metadata server is relocating.

For more details, see "Modify a CXFS Filesystem with `cmgr`" on page 555.

Delete a CXFS Filesystem with `cmgr`

Use the following command to delete a filesystem:

`delete cxfs_filesystem` *filesystemname* `[in cluster` *clustername*`]`

For example:

```
cmgr> delete cxfs_filesystem fs2 in cluster cxfs6-8
```

Switches and I/O Fencing Tasks with `cmgr`

The following tasks let you configure switches and I/O fencing. For general information, see "I/O Fencing" on page 51.

Note: Nodes without system controllers require I/O fencing to protect data integrity. A switch is mandatory to support I/O fencing; therefore, multiOS CXFS clusters require a switch. See the release notes for supported switches.

Define a Switch with `cmgr`

This section describes how to use the `cmgr` command to define a new Brocade switch to support I/O fencing in a cluster.

Note: To define a switch other than a Brocade switch, such as a QLogic switch, you must use the GUI or the `cxfs_admin` or `hafence`(1M) commands. (You cannot use the `cmgr` command to completely define a switch other than Brocade.) See "Create a Switch with `cxfs_admin`" on page 263 and "Switch Manipulation Using `hafence`" on page 286.

To define a new Brocade switch, use the following command:

`define switch` *switch_hostname* `username` *username* `password` *password* `[mask` *mask*`]`

Usage notes:

- `switch` specifies the hostname of the Fibre Channel switch; this is used to determine the IP address of the switch.
- `username` specifies the user name to use when sending a `telnet` message to the switch.
- `password` specifies the password for the specified *username*.
- `mask` specifies one of the following:
  - A list of ports in the switch that will never be fenced. The list has the following form, beginning with the # symbol and separating each port number with a comma::

    #port`,`*port*`,`*port*`...`

Each *port* is a decimal integer in the range 0 through 1023. For example, the following indicates that port numbers 2, 4, 5, 6, 7, and 23 will never be fenced:

```
#2,4,5,6,7,23
```

Note: For the bladed Brocade 48000 switch (where the port number is not unique), the value you should use for `mask` is the `Index` value that is displayed by the `switchShow` command. For example, the `switchShow` output below indicates that you would use a `mask` value of `#16` for port `0` in slot `2`:

```
brocade48000:admin> switchShow
Index Slot Port Address Media Speed State     Proto
===================================================
  0    1    0   010000   id    N4   Online    F-Port  10:00:00:00:c9:5f:9b:ea
  1    1    1   010100   id    N4   Online    F-Port  10:00:00:00:c9:5f:ab:d9
....
142    1   30   018e00   id    N4   Online    F-Port  50:06:0e:80:04:5c:0b:46
143    1   31   018f00   id    N4   Online    F-Port  50:06:0e:80:04:5c:0b:66
 16    2    0   011000   id    N4   Online    F-Port  10:00:00:00:c9:5f:a1:f5
 17    2    1   011100   id    N4   Online    F-Port  10:00:00:00:c9:5f:a1:72
...
```

– A hexadecimal string that represents ports in the switch that will never be fenced. Ports are numbered from 0. If a given bit has a binary value of 0, the port that corresponds to that bit is eligible for fencing operations; if 1, then the port that corresponds to that bit will always be excluded from any fencing operations. For an example, see Figure 10-5 on page 196.

CXFS administration nodes automatically discover the available HBAs and, when fencing is triggered, fence off all of the Fibre Channel HBAs when the `Fence` or `FenceReset` fail action is selected. However, masked HBAs will not be fenced. Masking allows you to prevent the fencing of devices that are attached to the SAN but are not shared with the cluster, to ensure that they remain available regardless of CXFS status. You would want to mask HBAs used for access to tape storage, or HBAs that are only ever used to access local (nonclustered) devices.

For example, using the direct port-number specification method:

```
cmgr> define switch ptg-brocade username admin password password mask #2,4,5,7,65
```

Or, using the hexadecimal bitmask method:

```
cmgr> define switch ptg-brocade username admin password password mask 200000000000000F4
```

Modify a Switch Definition with cmgr

To modify the user name, password, or mask for a Brocade switch, use the following command:

```
modify switch switch_hostname username username password password [mask mask]
```

The arguments are the same as for "Define a Switch with cmgr" on page 560.

Note: To modify the definition of another type of switch, such as QLogic, you must use the GUI, hafence(1M), or cxfs_admin(1M) commands. See "Switch Manipulation Using hafence" on page 286.

For example, to change the mask for switch ptg-brocade from A4 to 0 (which means that all of the ports on the switch are eligible for fencing), enter the following:

```
cmgr> modify switch ptg-brocade username admin password password mask 0
```

Raise the I/O Fence for a Node with cmgr

Raising an I/O fence isolates the node from the SAN; CXFS sends a messages via the telnet protocol to the switch and disables the port. After the node is isolated, it cannot corrupt data in the shared CXFS filesystem. Use the following command:

```
admin fence raise [node nodename]
```

nodename is the name of the node to be isolated.

For example, to isolate the default node, enter the following:

```
cmgr> admin fence raise
```

To isolate node Node3, enter the following:

```
cmgr> admin fence raise node Node3
```

Lower the I/O Fence for a Node with `cmgr`

To lower the I/O fence for a given node in order to reenable the port, allowing the node to connect to the SAN and access the shared CXFS filesystem, use the following command:

```
admin fence lower [node nodename]
```

nodename is the name of the node to be reconnected.

For example, to provide access for the default node, enter the following:

```
cmgr> admin fence lower
```

To provide access for node `Node3`, enter the following:

```
cmgr> admin fence lower node Node3
```

Update Switch Port Information with `cmgr`

To update the mappings in the cluster database between the host bus adapters (HBAs) and switch ports, use the following command:

```
admin fence update
```

You should run this command if you reconfigure any switch or add ports.

Delete a Switch Definition with `cmgr`

To delete a switch, use the following command:

```
delete switch switch_hostname
```

switch_hostname is the hostname of the Fibre Channel switch; this is used to determine the IP address of the switch.

For example:

```
cmgr> delete switch ptg-brocade
Successfully updated switch config.
```

Show Switches with cmgr

To display the switches in the system, use the following command:

```
show switches
```

To show the switches for a given node, use the following command:

show switch *hostname*

For example:

```
cmgr> show switch ptg-brocade
  Switch[0]
      *Hostname ptg-brocade Username admin Password password Mask 0
      Vendor BROCADE Number of ports 8
            0 0000000000000000 Reset
            1 210000e08b0102c6 Reset
            2 210000e08b01fec5 Reset
            3 210000e08b019dc5 Reset
            4 210000e08b0113ce Reset
            5 210000e08b027795 Reset thump
            6 210000e08b019ef0 Reset
            7 210000e08b022242 Reset
```

Query Switch Status with cmgr

To query the status of each port on the switch, use the following command:

```
admin fence query
```

For example:

```
cmgr> admin fence query
  Switch[0] "brocade04" has 16 ports
    Port 4 type=FABRIC status=enabled  hba=210000e08b0042d8 on host o200c
    Port 5 type=FABRIC status=enabled  hba=210000e08b00908e on host cxfs30
    Port 9 type=FABRIC status=enabled  hba=2000000173002d3e on host cxfssun3
```

For more verbose display, (which shows all ports on the switch, rather than only those attached to nodes in the default cluster), use the following command:

```
admin fence query verbose
```

For example:

```
cmgr> admin fence query verbose
  Switch[0] "brocade04" has 16 ports
    Port 0 type=FABRIC status=enabled  hba=2000000173003b5f on host UNKNOWN
    Port 1 type=FABRIC status=enabled  hba=2000000173003adf on host UNKNOWN
    Port 2 type=FABRIC status=enabled  hba=210000e08b023649 on host UNKNOWN
    Port 3 type=FABRIC status=enabled  hba=210000e08b021249 on host UNKNOWN
    Port 4 type=FABRIC status=enabled  hba=210000e08b0042d8 on host o200c
    Port 5 type=FABRIC status=enabled  hba=210000e08b00908e on host cxfs30
    Port 6 type=FABRIC status=enabled  hba=2000000173002d2a on host UNKNOWN
    Port 7 type=FABRIC status=enabled  hba=2000000173003376 on host UNKNOWN
    Port 8 type=FABRIC status=enabled  hba=2000000173002c0b on host UNKNOWN
    Port 9 type=FABRIC status=enabled  hba=2000000173002d3e on host cxfssun3
    Port 10 type=FABRIC status=enabled  hba=2000000173003430 on host UNKNOWN
    Port 11 type=FABRIC status=enabled  hba=200900a0b80c13c9 on host UNKNOWN
    Port 12 type=FABRIC status=disabled hba=0000000000000000 on host UNKNOWN
    Port 13 type=FABRIC status=enabled  hba=200d00a0b80c2476 on host UNKNOWN
    Port 14 type=FABRIC status=enabled  hba=1000006069201e5b on host UNKNOWN
    Port 15 type=FABRIC status=enabled  hba=1000006069201e5b on host UNKNOWN
```

Script Example

The following script defines a three-node cluster of type CXFS. The nodes are of type CXFS.

Note: This example only defines one network interface. The hostname is used here for simplicity; however, you may wish to use the IP address instead to avoid confusion. This example does not address the system controller definitions.

```
#!/usr/cluster/bin/cmgr -if
#
#Script to define a three-node cluster


define node cxfs6
        set hostname to cxfs6
        set is_cxfs to true
        set operating_system to irix
        set node_function to server_admin
        add nic cxfs6
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 1
        done
done

define node cxfs7
        set hostname to cxfs7
        set is_cxfs to true
        set operating_system to irix
        set node_function to server_admin
        add nic cxfs7
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 1
        done
done

define node cxfs8
        set hostname to cxfs8
        set is_cxfs to true
        set operating_system to irix
        set node_function to server_admin
        add nic cxfs8
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 1
        done
done
```

```
define cluster cxfs6-8
        set is_cxfs to true
        set is_failsafe to true
        set clusterid to 20
        add node cxfs6
        add node cxfs7
        add node cxfs8
        done
quit
```

After running this script, you would see the following output:

```
Successfully defined node cxfs6

Successfully defined node cxfs7

Successfully defined node cxfs8

Successfully defined cluster cxfs6-8
```

The following script defines two filesystems; `fs1` is mounted on all but node `cxfs8`, and `fs2` is mounted on all nodes:

```
#!/usr/cluster/bin/cmgr -if
# Script to define two filesystems
# Define fs1, do not mount on cxfs8
define cxfs_filesystem fs1 in cluster cxfs6-8
set device_name to /dev/cxvm/d76lun0s0
set mount_point to /mnts/fs1
set force to false
add cxfs_server cxfs6
  set rank to 0
  done
add cxfs_server cxfs7
  set rank to 1
  done
set dflt_local_status to enabled
add disabled_node cxfs8
done
#
# Define fs2, mount everywhere
define cxfs_filesystem fs2 in cluster cxfs6-8
```

```
set device_name to /dev/cxvm/d76lun0s1
set mount_point to /mnts/fs2
set force to false
add cxfs_server cxfs8
set rank to 0
done
set dflt_local_status to enabled
done
```

Creating a `cmgr` Script Automatically

After you have configured the cluster database, you can use the `build_cmgr_script` command to automatically create a `cmgr` script based on the contents of the cluster database. The generated script will contain the following:

- Node definitions
- Cluster definition
- Switch definitions
- CXFS filesystem definitions
- Parameter settings
- Any changes made using either the `cmgr` command or the GUI
- FailSafe information (in a coexecution cluster only)

As needed, you can then use the generated script to recreate the cluster database after performing a `cdbreinit`.

Note: You must execute the generated script on the first node that is listed in the script. If you want to execute the generated script on a different node, you must modify the script so that the node is the first one listed.

By default, the generated script is named:

`/var/cluster/ha/tmp/cmgr_create_cluster_`*clustername_processID*

You can specify an alternative pathname by using the `-o` option:

`build_cmgr_script [-o` *script_pathname*`]`

For more details, see the `build_cmgr_script` man page.

For example:

```
# /var/cluster/cmgr-scripts/build_cmgr_script -o /tmp/newcdb
Building cmgr script for cluster clusterA ...
build_cmgr_script: Generated cmgr script is /tmp/newcdb
```

The example script file contents are as follows; note that because `nodeE` is the first node defined, you must execute the script on `nodeE`:

```
#!/usr/cluster/bin/cmgr -f

# Node nodeE definition
define node nodeE
        set hostname to nodeE.example.com
        set operating_system to IRIX
        set is_failsafe to false
        set is_cxfs to true
        set node_function to server_admin
        set nodeid to 5208
        set reset_type to powerCycle
        add nic nodeE
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 1
        done
done

# Node nodeD definition
define node nodeD
        set hostname to nodeD.example.com
        set operating_system to IRIX
        set is_failsafe to false
        set is_cxfs to true
        set node_function to server_admin
        set nodeid to 5181
        set reset_type to powerCycle
        add nic nodeD
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 1
        done
done

# Node nodeF definition
define node nodeF
        set hostname to nodeF.example.com
        set operating_system to IRIX
        set is_failsafe to false
        set is_cxfs to true
        set node_function to server_admin
        set nodeid to 5401
        set reset_type to powerCycle
        add nic nodeF
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 1
        done
done

# Define cluster and add nodes to the cluster
define cluster clusterA
        set is_failsafe to false
        set is_cxfs to true
        set cx_mode to normal
        set clusterid to 35
done

modify cluster clusterA
        add node nodeD
        add node nodeF
        add node nodeE
done

set cluster clusterA

define cxfs_filesystem fs1
        set device_name to /dev/cxvm/fs1
        set mount_point to /fs1
        set force to false
        set dflt_local_status to enabled
        add cxfs_server nodeE
```

```
                set rank to 1
        done
        add cxfs_server nodeD
                set rank to 2
        done
        add cxfs_server nodeF
                set rank to 0
        done
done

define cxfs_filesystem fs2
        set device_name to /dev/cxvm/fs2
        set mount_point to /fs2
        set force to false
        set dflt_local_status to enabled
        add cxfs_server nodeE
                set rank to 1
        done
        add cxfs_server nodeD
                set rank to 2
        done
        add cxfs_server nodeF
                set rank to 0
        done
done

define cxfs_filesystem fs2
        set device_name to /dev/cxvm/fs2
        set mount_point to /fs2
        set force to false
        set dflt_local_status to enabled
        add cxfs_server nodeE
                set rank to 1
        done
        add cxfs_server nodeD
                set rank to 2
        done
        add cxfs_server nodeF
                set rank to 0
        done
done
```

```
# Setting CXFS parameters
modify cx_parameters
        set tie_breaker to none
done

quit
```

Troubleshooting cmgr

You should only use `cmgr` when you are logged in as `root`.

The following message may appear in `/var/cluster/ha/log/cli_`*hostname* if the underlying command line interface (CLI) was invoked by a login other than root:

```
CI_IPCERR_AGAIN, ipcclnt_connect(): file
/var/cluster/ha/comm/clconfd-ipc_cxfs0 lock failed - Permission
denied
```

Additional cmgr Examples

This section contains the following:

- "Example of Normal CXFS Shutdown Using `cmgr`" on page 573
- "Example of Forced CXFS Shutdown Using `cmgr`" on page 573
- "Example of Rejoining the Cluster after a Stopping CXFS Services Using `cmgr`" on page 574
- "Example of Rejoining the Cluster after a Forced CXFS Shutdown Using `cmgr`" on page 574
- "Example of Configuring Private Network Failover Using `cmgr`" on page 574
- "Example of Configuring a Large Cluster Using `cmgr`" on page 581
- "Example of Performing a Forced CXFS Shutdown Using `cmgr`" on page 582
- "Example of Relocation Error Using `cmgr`" on page 582

- "Example of Migration from an IRIX Cluster to an SGI ProPack Cluster Using `cmgr`" on page 582
- "Example of Querying Node Status Using `cmgr`" on page 590
- "Example of Pinging the System Controller Using `cmgr`" on page 590
- "Example of Monitoring Reset Lines Using `cmgr`" on page 591
- "Example of I/O Fencing Status Using `cmgr`" on page 591
- "Example of Using `build_cmgr_script` to Recreate the Cluster Database" on page 591

Also see:

- "System Reset Connection for Server-Capable Administration Nodes" on page 142
- "Performing a Power Cycle on a Node with `cmgr`" on page 415
- "Reseting a Node with `cmgr`" on page 415

Example of Normal CXFS Shutdown Using `cmgr`

To perform a normal CXFS shutdown, for example enter the following `cmgr` command:

```
cmgr> stop cx_services on node nodename for cluster clustername
```

This action deactivates CXFS services on **one** node, forming a new CXFS kernel membership after deactivating the node. If you want to stop CXFS services on multiple nodes, you must enter this command multiple times or perform the task using the GUI.

After you stop CXFS services on a node, the node is marked as inactive and is no longer used when calculating the CXFS kernel membership.

Example of Forced CXFS Shutdown Using `cmgr`

To perform an administrative stop, enter the following `cmgr` command to revoke the CXFS kernel membership of the local node:

```
cmgr> admin cxfs_stop
```

This action can also be triggered automatically by the kernel after a loss of CXFS kernel membership quorum.

Example of Rejoining the Cluster after a Stopping CXFS Services Using cmgr

The node will not rejoin the cluster after a reboot. The node will rejoin the cluster only when CXFS services are explicitly reactivated with the CXFS GUI (see "Start CXFS Services with the GUI" on page 189) or the following command:

```
cmgr> start cx_services on node nodename for cluster clustername
```

In `cxfs_admin`, you can disable individual nodes with the `disable` command.

Example of Rejoining the Cluster after a Forced CXFS Shutdown Using cmgr

After a forced CXFS shutdown, the local node will not resume CXFS kernel membership until the node is rebooted or until you explicitly allow CXFS kernel membership for the local node for example by entering the following `cmgr` command:

```
cmgr> admin cxfs_start
```

Example of Configuring Private Network Failover Using cmgr

This section provides an example of modifying a cluster to provide private network failover by using the `cmgr` command.

Suppose your cluster has the following configuration:

```
irix# cxfs-config
Global:
    cluster: mycluster (id 1)
    cluster state: enabled
    tiebreaker: yellow

Networks:

Machines:
...
    node red: node 55    cell 4  enabled  IRIX    server_admin
        hostname: red.example.com
        fail policy: Fence, Shutdown
        nic 0: address: 192.168.0.1 priority: 1

    node yellow: node 2     cell 3  enabled  IRIX    server_admin
        hostname: yellow.example.com
        fail policy: Fence, Shutdown
        nic 0: address: 192.168.0.2 priority: 1
```

To change the configuration to support private network failover, you would do the following:

1. Ensure that CXFS services are not active.

   Note: You cannot add a NIC or a network grouping while CXFS services are active (that is, when `start cx_services` has been executed); doing so can lead to cluster malfunction.

   If services have been started, stopped them as follows:

```
[root@linux root]# **cmgr -p**
Welcome to SGI Cluster Manager Command-Line Interface

cmgr> **stop cx_services for cluster mycluster**

CXFS services have been deactivated in cluster
mycluster
```

2. Add another set of NICs to support a second CXFS network. (The second network will be used as the failover network and can be the public network and thus does not have to be a second CXFS private network.) For example:

```
cmgr>
**modify node red**
Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional] ? (red.example.com)
Is this a FailSafe node <true|false> ? (false)
Is this a CXFS node <true|false> ? (true)
Partition ID[optional] ? (0)
Do you wish to modify failure hierarchy[y/n]:**n**
Reset type <powerCycle|reset|nmi> ? (powerCycle)
```

```
Do you wish to modify system controller
info[y/n]:n
Number of Network Interfaces ? (1) 2
NIC 1 - IP Address ? (192.168.0.1)
NIC 1 - Heartbeat HB (use network for heartbeats) <true|false> ?
(true)
NIC 1 - (use network for control messages) <true|false> ? (true)
NIC 1 - Priority <1,2,...> ? (1)
NIC 2 - IP Address ? 192.168.1.1
NIC 2 - Heartbeat HB (use network for heartbeats) <true|false> ?
true
NIC 2 - (use network for control messages) <true|false> ? true
NIC 2 - Priority <1,2,...> ? 2

Successfully modified node red

cmgr> modify node yellow
Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional] ? (yellow.example.com)
Is this a FailSafe node <true|false> ? (false)
Is this a CXFS node <true|false> ? (true)
Partition ID[optional] ? (0)
Do you wish to modify failure hierarchy[y/n]:n
Reset type <powerCycle|reset|nmi> ? (powerCycle)
Do you wish to modify system controller
info[y/n]:n
Number of Network Interfaces ? (1) 2
NIC 1 - IP Address ? (192.168.0.2)
NIC 1 - Heartbeat HB (use network for heartbeats) <true|false> ?
(true)
NIC 1 - (use network for control messages) <true|false> ? (true)
NIC 1 - Priority <1,2,...> ? (1)
NIC 2 - IP Address ? 192.168.1.2
NIC 2 - Heartbeat HB (use network for heartbeats) <true|false> ?
true
NIC 2 - (use network for control messages) <true|false> ? true
NIC 2 - Priority <1,2,...> ? 2

Successfully modified node yellow
```

Repeat this process for each node. You can use the `cxfs-config` command to display the defined NICs. For example:

```
irix# cxfs-config
...
        node red: node 55    cell 4  enabled  IRIX    server_admin
        hostname: red.example.com
        fail policy: Fence, Shutdown
        nic 0: address: 192.168.0.1 priority: 1
        nic 1: address: 192.168.1.1 priority: 2

    node yellow: node 2     cell 3  enabled  IRIX    server_admin
        hostname: yellow.example.com
        fail policy: Fence, Shutdown
        nic 0: address: 192.168.0.2 priority: 1
        nic 1: address: 192.168.1.2 priority: 2
```

3. Configure the NICs into networks. (CXFS will ignore NICs other than priority 1 unless you configure the NICs into networks.)

   a. Configure the primary network:

```
cmgr> modify cluster mycluster
Enter commands, you may enter "done" or "cancel" at any time to exit

Is this a FailSafe cluster <true|false> ? (false)
Is this a CXFS cluster <true|false> ? (true)
Cluster Notify Cmd [optional] ?
Cluster Notify Address [optional] ?
Cluster CXFS mode <normal|experimental>[optional] ? (normal)
Cluster ID ? (1)

Current nodes in cluster mycluster:
Node - 1: green
Node - 2: orange
Node - 3: red
Node - 4: purple
Node - 5: yellow
Node - 6: blue


No networks in cluster mycluster
```

```
Add nodes to or remove nodes/networks from cluster mycluster
Enter "done" when completed or "cancel" to abort

mycluster ? add net network 192.168.0.0 mask 255.255.255.0
mycluster ? done
Successfully modified cluster mycluster
```

At this point, `cxfs-config` will show the primary network (network `0`):

```
irix#
cxfs-config
...
Networks:
    net 0: type tcpip  192.168.0.0      255.255.255.0

Machines:
...
    node red: node 55    cell 4  enabled  IRIX    server_admin
        hostname: red.example.com
        fail policy: Fence, Shutdown
        nic 0: address: 192.168.0.1 priority: 1 network: 0
        nic 1: address: 192.168.1.1 priority: 2 network: none

    node yellow: node 2     cell 3  enabled  IRIX    server_admin
        hostname: yellow.example.com
        fail policy: Fence, Shutdown
        nic 0: address: 192.168.0.2 priority: 1 network: 0
        nic 1: address: 192.168.1.2 priority: 2 network: none
...
```

b. Configure the secondary network:

```
cmgr> modify cluster mycluster
Enter commands, you may enter "done" or "cancel" at any time to exit

Is this a FailSafe cluster <true|false> ? (false)
Is this a CXFS cluster <true|false> ? (true)
Cluster Notify Cmd [optional] ?
Cluster Notify Address [optional] ?
Cluster CXFS mode <normal|experimental>[optional] ? (normal)
Cluster ID ? (1)

Current nodes in cluster mycluster:
Node - 1: green
Node - 2: orange
Node - 3: red
Node - 4: purple
Node - 5: yellow
Node - 6: blue


No networks in cluster mycluster

Add nodes to or remove nodes/networks from cluster mycluster
Enter "done" when completed or "cancel" to abort

mycluster ? add net network 192.168.1.0 mask 255.255.255.0
mycluster ? done
Successfully modified cluster mycluster
```

The `cxfs-config` command will now display the secondary network (network 1):

```
irix#
cxfs-config
...
Networks:
    net 0: type tcpip  192.168.0.0      255.255.255.0
    net 1: type tcpip  192.168.1.0      255.255.255.0

Machines:
...
    node red: node 55    cell 4  enabled  IRIX    server_admin
        hostname: red.example.com
        fail policy: Fence, Shutdown
        nic 0: address: 192.168.0.1 priority: 1 network: 0
        nic 1: address: 192.168.1.1 priority: 2 network: 1

    node yellow: node 2     cell 3  enabled  IRIX    server_admin
        hostname: yellow.example.com
        fail policy: Fence, Shutdown
        nic 0: address: 192.168.0.2 priority: 1 network: 0
        nic 1: address: 192.168.1.2 priority: 2 network:
1
```

When you restart `cx_services`, the first membership delivered message will appear:

```
NOTICE: Membership delivered for cells 0x14.
Cell(age): 3(1) 4(1)
```

To delete a network, do the following:

```
cmgr> modify cluster mycluster
Enter commands, you may enter "done" or "cancel" at any time to exit

Is this a FailSafe cluster <true|false> ? (false)
Is this a CXFS cluster <true|false> ? (true)
Cluster Notify Cmd [optional] ?
Cluster Notify Address [optional] ?
Cluster CXFS mode <normal|experimental>[optional] ? (normal)
Cluster ID ? (1)

Current nodes in cluster mycluster:
Node - 1: green
Node - 2: orange
Node - 3: red
Node - 4: purple
Node - 5: yellow
Node - 6: blue


Current networks in cluster mycluster:
Network 0 - network 192.168.0.0, mask 255.255.255.0
Network 1 - network 192.168.1.0, mask 255.255.255.0

cmgr> modify cluster mycluster
Enter commands, when finished enter either "done" or "cancel"

mycluster ? remove net network 192.168.1.0
mycluster ? done
Successfully modified cluster mycluster
```

While there are networks defined, the cluster will try to use the highest priority network and failover as needed to the lower priority networks as possible. Deleting all networks will return the cluster to the default mode, in which a network consisting only of the priority 1 NICs is used.

For more information, see "Define a Cluster with `cmgr`" on page 533 and "Modify a Cluster with `cmgr`" on page 537.

Example of Configuring a Large Cluster Using `cmgr`

Following is an example `cmgr` script for configuring a one-node cluster that can be copied and repeated for the number of nodes required:

```
#!/usr/cluster/bin/cmgr -f
# Node nodename definition
define node nodename
        set hostname to nodename
        set operating_system to OS
        set node_function to server_admin|client_admin|client_only
        set is_failsafe to false
        set is_cxfs to true
        set nodeid to nodeID#
        set hierarchy to [system][fence][reset][fencereset][shutdown]
        set reset_type to powerCycle|reset|nmi
        add nic IP address  or nodename
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 1
        done
done
# Define cluster and add nodes to the cluster
define cluster clustername
        set is_failsafe to false
        set is_cxfs to true
        set cx_mode to normal
        set clusterid to clusterID#
done
modify cluster clustername
        add node nodename
done
set cluster clustername
define cxfs_filesystem filesystemname
        set device_name to /dev/cxvm/volumename
        set mount_point to /mountpoint
        set force to false
        set dflt_local_status to enabled
        add cxfs_server server1, server2, etc
                set rank to 0
        done
done
# Setting CXFS parameters
modify cx_parameters
        set tie_breaker to none
done
start cx_services for cluster clustername
quit
```

Example of Performing a Forced CXFS Shutdown Using cmgr

```
# /usr/cluster/bin/cmgr -p
cmgr> admin cxfs_stop
```

Example of Relocation Error Using cmgr

If you try to relocate a filesystem and see an error similar to the following `cmgr` example, it means that relocation has not been enabled: :

```
CMD(/bin/mount -n -o remount,set_server=node1 /lsan1): exited with status
32 (0x20)

Failed to admin:
        cxfs_relocate

admin command failed
```

To allow the relocation to occur, you must enable relocation as specified in "Relocation" on page 25.

Example of Migration from an IRIX Cluster to an SGI ProPack Cluster Using cmgr

CXFS supports a running cluster with a single type of operating system for administration nodes: either all IRIX or all SGI ProPack. To migrate from an IRIX cluster to an SGI ProPack cluster, follow the instruction in this chapter. For assistance, contact SGI Managed Services.

You should also be aware the differences between basic IRIX and Linux system administration. See "An Overview of Differences Between IRIX and Linux System Administration" on page 597.

Note: The following procedure assumes that the filesystems in the cluster you want to migrate do not have block sizes greater than the system page size and that they are not real-time filesystems. These types of filesystems are supported on IRIX but not on SGI ProPack.

This example uses `cmgr`, but you could perform a similar procedure using `cxfs_admin` or the GUI. The example begins with a cluster named `performance` having a two IRIX server-capable nodes named `rum` and `snake` and a Solaris client-only node named `ray`:

```
rum # clconf_info

Event at [2004-02-13 07:57:17]

Membership since Thu Feb 12 15:15:26 2004
____________ ______ ________ ______ ______
Node         NodeID Status   Age    CellID
____________ ______ ________ ______ ______
snake             1 up            2      1
rum               2 up            2      2
ray               3 up            1      0
____________ ______ ________ ______ ______
1 CXFS FileSystems
/dev/cxvm/V9500 on /cxfs/V9500  enabled  server=(snake)  2 client(s)=(ray,rum)  status=UP
```

Do the following:

1. Unmount the CXFS filesystems cluster-wide within CXFS. For example:

   ```
   cmgr> admin cxfs_unmount cxfs_filesystem V9500

   cxfs_unmount operation successful
   ```

2. Mount and unmount the filesystems locally, which will ensure that the XFS log plays back cleanly. For example:

   ```
   # mount /dev/cxvm/V9500 /mnt
   # umount /mnt
   ```

3. Stop CXFS services on all nodes. For example on the IRIX node `rum`:

   ```
   cmgr> stop cx_services for cluster performance

   CXFS services have been deactivated in cluster performance
   ```

   Note: If you use `cxfs_admin`, you must issue a command for each node.

   ---

4. Define the administration node with the SGI ProPack operating system type. For example on the IRIX node `rum`:

```
cmgr> define node bang
Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional] ? bang
Is this a FailSafe node <true|false> ? false
Is this a CXFS node <true|false> ? true
Operating System <IRIX|Linux32|Linux64|AIX|HPUX|Solaris|MacOSX|Windows> ? Linux64
Node Function <server_admin|client_admin|client_only> ? server_admin
Node ID[optional] ? 64
Partition ID[optional] ? (0)
Do you wish to define failure hierarchy[y/n]:y
Hierarchy option 0 <System|FenceReset|Fence|Reset|Shutdown>[optional] ? Fence
Hierarchy option 1 <System|FenceReset|Fence|Reset|Shutdown>[optional] ?
Reset type <powerCycle|reset|nmi> ? (powerCycle)
Do you wish to define system controller info[y/n]:n
Number of Network Interfaces ? (1)
NIC 1 - IP Address ? bang-p
NIC 1 - Heartbeat HB (use network for heartbeats) <true|false> ? true
```

```
NIC 1 - (use network for control messages) <true|false> ? true
NIC 1 - Priority <1,2,...> ? 1

Successfully defined node bang
```

5. Add the SGI ProPack administration node to the cluster. For example on the IRIX node `rum`:

```
cmgr> modify cluster performance
Enter commands, you may enter "done" or "cancel" at any time to exit

Is this a FailSafe cluster <true|false> ? (false)
Is this a CXFS cluster <true|false> ? (true)
Cluster Notify Cmd [optional] ?
Cluster Notify Address [optional] ?
Cluster CXFS mode <normal|experimental>[optional] ? (normal)
Cluster ID ? (1)

Current nodes in cluster performance:
Node - 1: ray
Node - 2: snake
Node - 3: rum


No networks in cluster performance

Add nodes to or remove nodes/networks from cluster performance
Enter "done" when completed or "cancel" to abort

performance ? add node bang
performance ? done
Added node <bang> to cluster <performance>
Successfully modified cluster performance
```

6. Modify the CXFS filesystems to remove the IRIX administration nodes as metadata servers and add the new SGI ProPack administration node as metadata server. For example, on the IRIX node `rum`:

   ```
   cmgr> modify cxfs_filesystem V9500

   (Enter "cancel" at any time to abort)
   ```

```
Device ? (/dev/cxvm/V9500)
Mount Point ? (/cxfs/V9500)
Mount Options[optional] ?
Use Forced Unmount ? <true|false> ? (false)
Grio Qualififed Bandwidth[optional] ?
Grio managed filesystem ? <true|false>[optional] ?
Default Local Status  ? (enabled)

MODIFY CXFS FILESYSTEM OPTIONS

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:2

Current servers:
CXFS Server 1 - Rank: 0         Node: rum
CXFS Server 2 - Rank: 1         Node: snake

Server Node ? rum

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:2
```

```
Current servers:
CXFS Server 1 - Rank: 1         Node: snake

Server Node ? snake

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:1

No current servers

Server Node ? bang
Server Rank ? 1

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:9
Successfully modified cxfs_filesystem V9500
```

After you complete this step, the filesystems would show the following information:

```
cmgr> show cxfs_filesystem V9500

Name: V9500
Device: /dev/cxvm/V9500
Mount Point: /cxfs/V9500
Forced Unmount: false
Global Status: disabled
Default Local Status: enabled

Server Name: bang
       Rank: 1
```

7. Remove the IRIX administration nodes from the cluster. For example, switching to the SGI ProPack node bang:

```
cmgr> modify cluster performance
Enter commands, you may enter "done" or "cancel" at any time to exit

Is this a FailSafe cluster <true|false> ? (false)
Is this a CXFS cluster <true|false> ? (true)
Cluster Notify Cmd [optional] ?
Cluster Notify Address [optional] ?
Cluster CXFS mode <normal|experimental>[optional] ? (normal)
Cluster ID ? (1)

Current nodes in cluster performance:
Node - 1: ray
Node - 2: snake
Node - 3: rum
Node - 4: bang


Add nodes to or remove nodes/networks from cluster performance
Enter "done" when completed or "cancel" to abort

performance ? remove node rum
performance ? remove node snake
performance ? done
Successfully modified cluster performance
```

8. Delete the IRIX administration nodes from the pool. For example, from the SGI ProPack node `bang`:

   ```
   cmgr> delete node rum
   Deleted node (rum).

   cmgr> delete node snake
   Deleted node (snake).
   ```

9. Start CXFS services for all nodes in the cluster. For example, from the SGI ProPack node `bang`:

```
cmgr> start cx_services for cluster performance

CXFS services have been activated in cluster performance
```

10. Mount the CXFS filesystems. For example, from the SGI ProPack node `bang`:

    ```
    cmgr> admin cxfs_mount cxfs_filesystem V9500

    cxfs_mount operation successful
    ```

After completing this procedure, the cluster information is as follows:

```
[root@bang root]# clconf_info

Event at [2004-02-13 08:44:18]

Membership since Fri Feb 13 08:44:13 2004
____________ ______ ________ ______ ______
Node         NodeID Status   Age    CellID
____________ ______ ________ ______ ______
ray               3 up            1      0
bang             64 up            1      3
____________ ______ ________ ______ ______
1 CXFS FileSystems
/dev/cxvm/V9500 on /cxfs/V9500  enabled  server=(bang)  1 client(s)=(ray)  status=UP
```

For more information about using the `cmgr` command to perform this procedure, see the following:

- "Unmount a CXFS Filesystem with `cmgr`" on page 554
- "Stop CXFS Services with `cmgr`" on page 541
- "Define a Node with `cmgr`" on page 513
- "Modify a Cluster with `cmgr`" on page 537
- "Modify a CXFS Filesystem with `cmgr`" on page 555
- "Modify a Cluster with `cmgr`" on page 537
- "Delete a Node with `cmgr`" on page 529
- "Start CXFS Services with `cmgr`" on page 541
- "Mount a CXFS Filesystem with `cmgr`" on page 553

Example of Checking Cluster Status Using `cmgr`

To query node and cluster status, use the following `cmgr` command on a CXFS administration node:

```
cmgr> show status of cluster cluster_name
```

Example of Querying Node Status Using `cmgr`

To query node status, use the following `cmgr` command:

```
cmgr> show status of node node_name
```

Example of Pinging the System Controller Using `cmgr`

When CXFS is running, you can determine whether the system controller on a node is responding by using the following `cmgr` command:

```
cmgr> admin ping node node_name
```

Note: This is not required when using `cxfs_admin` because it will attempt to verify that each node is connected as it is added to the cluster.

This command uses the CXFS daemons to test whether the system controller is responding.

You can verify reset connectivity on a node in a cluster even when the CXFS daemons are not running by using the `standalone` option of the `admin ping` command:

```
cmgr> admin ping standalone node node_name
```

This command calls the `ping` command directly to test whether the system controller on the indicated node is responding.

Example of Monitoring Reset Lines Using `cmgr`

You can use the `cmgr` command to ping the system controller at a node as follows (line break for readability):

```
cmgr> admin ping dev_name device_name of dev_type device_type
with sysctrl_type system_controller_type
```

Note: This is not required when using `cxfs_admin`.

Example of I/O Fencing Status Using `cmgr`

To check the current fencing status, use the `admin fence query` command in `cmgr`

To check current failure action settings, use the `show node` *nodename* command in `cmgr`.

Example of Using `build_cmgr_script` to Recreate the Cluster Database

Note: Also see "Saving and Recreating the Current Configuration with `cxfs_admin`" on page 269.

You can use the `build_cmgr_script` command from one node in the cluster to create a `cmgr` script that will recreate the node, cluster, switch, and filesystem definitions for all nodes in the cluster database. You can then later run the resulting script to recreate a database with the same contents; this method can be used for missing or corrupted cluster databases.

Note: The `build_cmgr_script` script does not contain local logging information, so it cannot be used as a complete backup/restore tool.

To perform a database backup, use the `build_cmgr_script` script from one node in the cluster, as described in "Creating a `cmgr` Script Automatically" on page 568.

Caution: Do not make configuration changes while you are using the `build_cmgr_script` command.

By default, this creates a `cmgr` script in the following location:

`/var/cluster/ha/tmp/cmgr_create_cluster`*_clustername_processID*

You can specify another filename by using the `-o` option.

To perform a restore on all nodes in the pool, do the following:

1. Stop CXFS services on all nodes in the cluster.
2. Stop the cluster database daemons on each node.
3. Remove all copies of the old database by using the `cdbreinit` command on each node.
4. Execute the `cmgr` script (which was generated by the `build_cmgr_script` script) on the node that is defined first in the script. This will recreate the backed-up database on each node.

   Note: If you want to run the generated script on a different node, you must modify the generated script so that the node is the first one listed in the script.

   ---

5. Restart cluster database daemons on each node.

For example, to back up the current database, clear the database, and restore the database to all administration nodes, do the following on administration nodes as directed:

On one node:

```
# /var/cluster/cmgr-scripts/build_cmgr_script -o /tmp/newcdb
Building cmgr script for cluster clusterA ...
build_cmgr_script: Generated cmgr script is /tmp/newcdb
```

On one node:

```
     cmgr> stop cx_services for cluster clusterA
```

On each node:

```
# /etc/init.d/cxfs stop
```

On each node:

 IRIX:

```
     # /etc/init.d/cluster stop
```

 SGI ProPack:

```
     # /etc/init.d/cxfs_cluster stop
```

On each node:

```
# /usr/cluster/bin/cdbreinit
```

On each node:

 IRIX:

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

 SGI ProPack:

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

*On the *first* node listed in the /tmp/newcdb script:*

```
# /tmp/newcdb
```

Migration from `cmgr` to `cxfs_admin`

If you have scripts that use `cmgr`, you should do the following to use `cxfs_admin`:

1. Do one of the following:
   - Run the `cmgr` script and build the cluster configuration from scratch
   - Start with the cluster database in the desired configuration
2. Run the `cxfs_admin config` command to generate a `cxfs_admin` script. This script should be the `cxfs_admin` equivalent of the `cmgr` script. See "Saving and Recreating the Current Configuration with `cxfs_admin`" on page 269.
3. Modify the script generated in step 2 so that the server-capable administration node on which `cxfs_admin` will be run to generate the cluster is the first node created in the script. (By default, the `cxfs_admin config` command output lists nodes in alphabetical order by node name without regard to node type.) This script can then be used to regenerate the cluster later. See "Saving and Recreating the Current Configuration with `cxfs_admin`" on page 269.

Note: The `bash` shell interprets `cmgr`-generated scripts, but `cxfs_admin` interprets `cxfs_admin`—generated scripts.

Migration from a Cluster with IRIX Server-Capable Administration Nodes

The information in this appendix will help you migrate from a cluster with IRIX server-capable administration nodes to a cluster with SGI ProPack for Linux server-capable administration nodes. It discusses the following:

- "An Overview of Differences Between IRIX and Linux System Administration" on page 597
- "Caveats for Migrating from IRIX" on page 598
- "Migration Procedure Using `cxfs_admin`" on page 601
- "Migration Procedure Using `cmgr`" on page 603
- "Migration Troubleshooting" on page 611

You must ensure that you have the correct network configuration for the new Linux server-capable administration nodes; see Chapter 6, "Preinstallation Steps" on page 103. For assistance, contact SGI Managed Services.

An Overview of Differences Between IRIX and Linux System Administration

If you are migrating from a cluster with IRIX metadata servers to a cluster with SGI ProPack metadata servers, you should understand the differences between IRIX and Linux system administration. The details of these differences are beyond the scope of this guide, but a brief overview includes:

- Installation tools
- Mount options
- Paths
- Location of kernel system-tunable parameters

For more information, see the operating system documentation.

See also:

- "SGI ProPack Limitations and Considerations" on page 111
- Chapter 12, "Administration and Maintenance" on page 275
- Appendix D, "Path Summary" on page 449

Caveats for Migrating from IRIX

This section discusses the following:

- "Changing `SGIRDAC` Mode to `SGIAVT` Mode for SGI RAID" on page 598
- "Recreating Filesystems with the Appropriate Naming Version" on page 599
- "Recreating Filesystems with Large Block Sizes" on page 600

Changing `SGIRDAC` Mode to `SGIAVT` Mode for SGI RAID

CXFS does not support `SGIRDAC` mode. To convert from `SGIRDAC` to `SGIAVT`, do the following:

1. Install the latest supported firmware on the RAID.
2. Determine the IP address for one of the controllers on each RAID box.
3. Make a script `settype.scr` that contains the following line:

   ```
   set storageArray defaultHostType="modename";
   ```

   Note: The capitalization and punctuation in the above line are required.

   To switch to `SGIAVT` mode, use the following line:

   ```
   set storageArray defaultHostType="SGIAVT";
   ```

   For the InfiniteStorage 220, use the CLI client to set the host type to `SGIAVT`:

   ```
   smicli -w SA_WWID -c 'set storageArray defaultHostType="SGIAVT";'
   ```

To determine the value for *SA_WWID*, invoke the following:

```
smicli -d -w
```

For example:

```
# smicli -d -w
unit1  600a0b80002459d40000000045003fbc        localhost
          |---> SA_WWID
```

4. Run the following for one of the controllers per RAID box:

```
/opt/tpssm/client/tpssmcli RAID_IPaddress -f settype.scr
```

For example:

```
# /opt/tpssm/client/tpssmcli 192.168.0.1 -f settype.scr
Performing syntax check...

Syntax check complete.

Executing script...

Script execution complete.

tpssmcli completed successfully.
```

Recreating Filesystems with the Appropriate Naming Version

Linux does not support version 1 naming on filesystems. You must examine the parameters for each filesystem by using the following command on the active metadata server for each filesystem:

```
xfs_growfs -n mountpoint
```

For example, the following shows that the `/stripe1` filesystem has version 2 naming:

```
irixMDS# xfs_growfs -n /stripe1
meta-data=/stripe1              isize=256    agcount=16, agsize=1663360 blks
         =                      sectsz=512   attr=0, parent=0
data     =                      bsize=4096   blocks=26611968, imaxpct=25
         =                      sunit=128    swidth=768 blks, unwritten=1
         =                      mmr=0
naming   =version 2             bsize=4096   mixed-case=Y
```

```
log      =internal               bsize=4096   blocks=13056 version=1
         =                       sectsz=512   sunit=0 blks lazy-count=0
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

If you have a filesystem that displays version 1 naming, you must recreate it on a Linux server-capable administration node so that it will have version 2 naming. Do the following:

1. Dump the filesystem data to backup media by using the `xfsdump`(1M) command.

   Warning: If this step is not performed, all data currently on the filesystems will become inaccessible and will be permanently lost.

2. Unmount the CXFS filesystems clusterwide by using the `cxfs_admin` command.
3. Recreate the filesystem by using the `mkfs.xfs`(8) command on a Linux server-capable administration node. Preserve any nondefault filesystem parameters by specifying the applicable options to the `mkfs.xfs` command.
4. Restore the filesystem data from backup media by using the `xfsrestore`(8) command.

For more information, see the man pages for the above commands and the following:

- *IRIX Admin: Disks and Filesystems*
- *XFS for Linux Administration*

Recreating Filesystems with Large Block Sizes

The maximum block size on Linux nodes varies by architecture:

- For SGI Altix ia64 server-capable administration nodes, the maximum block size is 16K.
- For SGI Altix XE x86_64 server-capable administration nodes, the maximum block size is 4K.

If you have filesystems with larger block sizes than the above, you must recreate them by performing a dump and a restore, changing the block size so that it does not exceed the maximum for the architecture of your systems.

Note: SGI recommends a block size of 4K for CXFS filesystems in a multiOS cluster because this is the only block size that is supported on all CXFS hardware platforms.

Migration Procedure Using `cxfs_admin`

Note: The following procedure assumes that the filesystems in the cluster you want to migrate do not have block sizes greater than the system page size and that they are not real-time filesystems or external log filesystems. These types of filesystems are supported on IRIX but not on SGI ProPack.

Following is a procedure for using `cxfs_admin` to migrate from a cluster with IRIX server-capable administration nodes to a cluster with SGI ProPack for Linux server-capable administration nodes. For details about using `cxfs_admin`, see Chapter 11, "Reference to `cxfs_admin` Tasks" on page 217.

Note: The cluster cannot be used for product work during this process.

1. As a precaution, save the current cluster database contents by using the `build_cmgr_script` command. See "Creating a `cmgr` Script Automatically" on page 568.

2. Ensure you have CXFS 5.0 and XVM 5.0 licenses installed on the Linux server-capable administration nodes. See:

   - "Upgrading Licenses from 4.*X* to 5.0" on page 278
   - Chapter 5, "CXFS License Keys" on page 89

3. From an IRIX server-capable administration node:

   a. Unmount the CXFS filesystems clusterwide by using the `cxfs_admin` command. For example, to unmount the CXFS `V9500` filesystem clusterwide:

```
cxfs_admin:mycluster> unmount V9500
```

b. Use the IRIX `mount` and`umount` commands to mount and unmount the filesystems locally, which will ensure that the XFS log plays back cleanly. For example, mount and unmount the `V9500` filesystem locally:

```
irixadmin# mount /dev/cxvm/V9500 /mnt
irixadmin# umount /mnt
```

c. Disable each IRIX server-capable administration node by using the `cxfs_admin` command. For example, to disable the node named `irixnode1`:

```
cxfs_admin:mycluster> disable irixnode1
```

Repeat the `disable` command for each IRIX server-capable administration node.

d. Create a Linux server-capable administration node in the cluster. This will allow the cluster database information to be transferred to the Linux nodes.

Note: This migration situation is the only time that mixed OS server-capable administration nodes are allowed in the same cluster.

For example:

```
cxfs_admin:mycluster> create node name=linuxnode1 type=server_admin os=Linux
private_net=10.0.10.1 enabled=true hostname=linuxnode1.domain.com
```

4. From the Linux server-capable administration node created in step d above, do the following:

a. Delete each IRIX server-capable administration node by using the `cxfs_admin` command. For example, to delete the node named `irixnode1`:

```
cxfs_admin:mycluster> delete irixnode1
```

Repeat the `delete` command for each IRIX server-capable administration node.

b. Create any additional Linux server-capable administration nodes in the cluster. For example, to create a node named `linuxnode2`:

```
cxfs_admin:mycluster> create node name=linuxnode2 type=server_admin os=Linux
private_net=10.0.10.2 enabled=true hostname=linuxnode2.domain.com
```

Repeat the `create` command for each Linux server-capable administration node.

c. Modify the CXFS filesystems to assign the new Linux server-capable administration nodes as potential metadata servers. For example:

```
cxfs_admin:mycluster> modify filesystem name=V9500 servers=linuxnode1,linuxnode2
```

d. Mount the CXFS filesystems by using the `cxfs_admin` command. For example, to mount the `V9500` filesystem:

```
cxfs_admin:mycluster> mount V9500
```

e. Check cluster configuration:

```
cxfs_admin:mycluster> status
```

You could also use the `cxfs-config` command to verify the status.

5. Dispose of the former IRIX server-capable administration nodes properly:

   a. Stop CXFS processes on each IRIX server-capable administration node:

   ```
   irixadmin# service grio2 stop (if running GRIOv2)
   irixadmin# service cxfs stop
   irixadmin# service cluster stop
   ```

   Repeat these commands on each IRIX server-capable administration node.

   b. Do one of the following:

      - Transform the nodes into IRIX client-only nodes and add them to the cluster. Execute the `stop` commands above and follow the procedure in "Transforming a Server-Capable Administration Node into a Client-Only Node" on page 290.

      - Remove the IRIX server-capable administration nodes physically from the network.

Migration Procedure Using cmgr

Note: The following procedure assumes that the filesystems in the cluster you want to migrate do not have block sizes greater than the system page size and that they are not real-time filesystems or external log filesystems. These types of filesystems are supported on IRIX but not on SGI ProPack.

The example in this chapter uses `cmgr`, but you could perform a similar procedure using `cxfs_admin` or the GUI; for a brief procedure using `cxfs_admin`, see "Migration Procedure Using `cxfs_admin`" on page 601. The example begins with a cluster named `performance` having a two IRIX server-capable administration nodes named `rum` and `snake` and a Solaris client-only node named `ray`:

```
rum # clconf_info

Event at [2004-02-13 07:57:17]

Membership since Thu Feb 12 15:15:26 2004
____________ ______ ________ ______ ______
Node         NodeID Status   Age    CellID
____________ ______ ________ ______ ______
snake             1 up            2      1
rum               2 up            2      2
ray               3 up            1      0
____________ ______ ________ ______ ______
1 CXFS FileSystems
/dev/cxvm/V9500 on /cxfs/V9500  enabled  server=(snake)  2 client(s)=(ray,rum)  status=UP
```

Do the following:

1. As a precaution, save the current cluster database contents by using the `build_cmgr_script` command. See "Creating a `cmgr` Script Automatically" on page 568.
2. Ensure you have CXFS 5.0 and XVM 5.0 licenses installed on the Linux server-capable administration nodes. See:
   - "Upgrading Licenses from 4.*X* to 5.0" on page 278
   - Chapter 5, "CXFS License Keys" on page 89
3. Unmount the CXFS filesystems cluster-wide within CXFS. For example:

   ```
   cmgr> admin cxfs_unmount cxfs_filesystem V9500

   cxfs_unmount operation successful
   ```

4. Mount and unmount the filesystems locally, which will ensure that the XFS log plays back cleanly. For example:

   ```
   # mount /dev/cxvm/V9500 /mnt
   # umount /mnt
   ```

5. Stop CXFS services on all nodes. For example on the IRIX node `rum`:

   ```
   cmgr> stop cx_services for cluster performance

   CXFS services have been deactivated in cluster performance
   ```

   Note: If you use `cxfs_admin`, you must issue a command for each node.

6. Define the server-capable administration node with the SGI ProPack operating system type. For example on the IRIX node `rum`:

   Note: This migration situation is the only time that mixed OS server-capable administration nodes are allowed in the same cluster.

```
cmgr> define node bang
Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional] ? bang
Is this a FailSafe node <true|false> ? false
Is this a CXFS node <true|false> ? true
Operating System <IRIX|Linux32|Linux64|AIX|HPUX|Solaris|MacOSX|Windows> ? Linux64
Node Function <server_admin|client_admin|client_only> ? server_admin
Node ID[optional] ? 64
Partition ID[optional] ? (0)
Do you wish to define failure hierarchy[y/n]:y
Hierarchy option 0 <System|FenceReset|Fence|Reset|Shutdown>[optional] ? Fence
Hierarchy option 1 <System|FenceReset|Fence|Reset|Shutdown>[optional] ?
Reset type <powerCycle|reset|nmi> ? (powerCycle)
Do you wish to define system controller info[y/n]:n
Number of Network Interfaces ? (1)
NIC 1 - IP Address ? bang-p
NIC 1 - Heartbeat HB (use network for heartbeats) <true|false> ? true
NIC 1 - (use network for control messages) <true|false> ? true
NIC 1 - Priority <1,2,...> ? 1
```

```
Successfully defined node bang
```

7. Add the SGI ProPack server-capable administration node to the cluster. For example on the IRIX node `rum`:

```
cmgr> modify cluster performance
Enter commands, you may enter "done" or "cancel" at any time to exit

Is this a FailSafe cluster <true|false> ? (false)
Is this a CXFS cluster <true|false> ? (true)
Cluster Notify Cmd [optional] ?
Cluster Notify Address [optional] ?
Cluster CXFS mode <normal|experimental>[optional] ? (normal)
Cluster ID ? (1)

Current nodes in cluster performance:
Node - 1: ray
Node - 2: snake
Node - 3: rum


No networks in cluster performance

Add nodes to or remove nodes/networks from cluster performance
Enter "done" when completed or "cancel" to abort

performance ? add node bang
performance ? done
Added node <bang> to cluster <performance>
Successfully modified cluster performance
```

8. Modify the CXFS filesystems to remove the IRIX server-capable administration nodes as metadata servers and add the new SGI ProPack server-capable administration node as metadata server. For example, on the IRIX node `rum`:

   ```
   cmgr> modify cxfs_filesystem V9500

   (Enter "cancel" at any time to abort)

   Device ? (/dev/cxvm/V9500)
   Mount Point ? (/cxfs/V9500)
   Mount Options[optional] ?
   ```

```
Use Forced Unmount ? <true|false> ? (false)
Grio Qualififed Bandwidth[optional] ?
Grio managed filesystem ? <true|false>[optional] ?
Default Local Status  ? (enabled)

MODIFY CXFS FILESYSTEM OPTIONS

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:2

Current servers:
CXFS Server 1 - Rank: 0         Node: rum
CXFS Server 2 - Rank: 1         Node: snake

Server Node ? rum

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:2

Current servers:
CXFS Server 1 - Rank: 1         Node: snake
```

```
Server Node ? snake

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:1

No current servers

Server Node ? bang
Server Rank ? 1

        0) Modify Server.
        1) Add Server.
        2) Remove Server.
        3) Add Enabled Node.
        4) Remove Enabled Node.
        5) Add Disabled Node.
        6) Remove Disabled Node.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:9
Successfully modified cxfs_filesystem V9500
```

After you complete this step, the filesystems would show the following information:

```
cmgr> show cxfs_filesystem V9500

Name: V9500
Device: /dev/cxvm/V9500
```

```
Mount Point: /cxfs/V9500
Forced Unmount: false
Global Status: disabled
Default Local Status: enabled

Server Name: bang
       Rank: 1
```

9. Remove the IRIX server-capable administration nodes from the cluster. For example, switching to the SGI ProPack node bang:

```
cmgr> modify cluster performance
Enter commands, you may enter "done" or "cancel" at any time to exit

Is this a FailSafe cluster <true|false> ? (false)
Is this a CXFS cluster <true|false> ? (true)
Cluster Notify Cmd [optional] ?
Cluster Notify Address [optional] ?
Cluster CXFS mode <normal|experimental>[optional] ? (normal)
Cluster ID ? (1)

Current nodes in cluster performance:
Node - 1: ray
Node - 2: snake
Node - 3: rum
Node - 4: bang


Add nodes to or remove nodes/networks from cluster performance
Enter "done" when completed or "cancel" to abort

performance ? remove node rum
performance ? remove node snake
performance ? done
Successfully modified cluster performance
```

10. Delete the IRIX server-capable administration nodes from the pool. For example, from the SGI ProPack node bang:

```
cmgr> delete node rum
Deleted node (rum).
```

```
cmgr> delete node snake
Deleted node (snake).
```

11. Start CXFS services for all nodes in the cluster. For example, from the SGI ProPack node `bang`:

```
cmgr> start cx_services for cluster performance

CXFS services have been activated in cluster performance
```

12. Mount the CXFS filesystems. For example, from the SGI ProPack node `bang`:

```
cmgr> admin cxfs_mount cxfs_filesystem V9500

cxfs_mount operation successful
```

13. If you are running other storage software products on server-capable administration nodes, confirm that you have installed either TPSSM or SMI, as appropriate for your hardware.

After completing this procedure, the cluster information is as follows:

```
[root@bang root]# clconf_info

Event at [2004-02-13 08:44:18]

Membership since Fri Feb 13 08:44:13 2004
____________ ______ ________ ______ ______
Node         NodeID Status   Age    CellID
____________ ______ ________ ______ ______
ray               3 up            1      0
bang             64 up            1      3
____________ ______ ________ ______ ______
1 CXFS FileSystems
/dev/cxvm/V9500 on /cxfs/V9500  enabled  server=(bang)  1 client(s)=(ray)  status=UP
```

For more information about using the GUI, see the following:

- "Unmount CXFS Filesystems with the GUI" on page 208
- "Stop CXFS Services with the GUI" on page 189
- "Define a Node with the GUI" on page 171
- "Add or Remove Nodes in the Cluster with the GUI" on page 180

- "Modify a CXFS Filesystem with the GUI" on page 206
- "Add or Remove Nodes in the Cluster with the GUI" on page 180
- "Delete a Node with the GUI" on page 185
- "Start CXFS Services with the GUI" on page 189
- "Mount CXFS Filesystems with the GUI" on page 207

Migration Troubleshooting

The following sections discuss possible problems you may encounter after migrating from an IRIX cluster to an SGI ProPack cluster:

- "Filesystems Will Not Mount" on page 611
- "DMF Filesystems Will Not Mount" on page 612
- "DMF Filesystems Will Not Mount" on page 612

Filesystems Will Not Mount

Messages such as the following indicate that the filesystem was not cleanly unmounted from the IRIX metadata server:

```
Jan 29 22:06:07 4A:cxfs2 kernel: XFS: nil uuid in log - IRIX style log
Jan 29 22:06:07 5A:cxfs2 kernel: Starting XFS recovery on filesystem:
xvm-0 (dev: xvm-0)
Jan 29 22:06:07 4A:cxfs2 kernel: XFS: dirty log written in incompatible
format - can’t recover
```

To resolve this problem, you must return to the IRIX node and then mount and umount the filesystem locally on the IRIX node in order to replay the dirty log messages (as in step 4 above in "Migration Procedure Using `cmgr`" on page 603).

Caution: Do not steal the XVM volumes to the local host. Mounting `/dev/cxvm/`*volname* locally on `/mnt` is sufficient.

DMF Filesystems Will Not Mount

If you have DMF filesystems and have `dmi` as a mount option,you must edit the`/etc/sysconfig/sysctl`file to turn on DMAPI probing in order to mount CXFS filesystems. Change the bottom line from:

```
DMAPI_PROBE="no"
```

to:

```
DMAPI_PROBE="yes"
```

Do Not Use `extlog` or `rtfs` Filesystems

If you have SGI ProPack server-capable administration nodes, you cannot use `extlog` or `rtfs` filesystems.

Appendix J

Deprecated Commands

Table J-1 lists commands whose use has been generally replaced by `cxfs_admin` and/or the CXFS GUI.

Table J-1 Deprecated Commands

Command	Description
`clconf_status`	Provides provides a `curses` interface to display status information gathered by the `cad` daemon.
`cluster_status`	Obtains configuration and status information.
`cmgr`	Configures the cluster database and administers CXFS

Initial Configuration Checklist

Following is a checklist of the steps you must perform when installing and configuring a CXFS system.

Caution: CXFS is a complex product. To ensure that it is installed and configured in an optimal manner, you **must** purchase initial setup services from SGI.

This checklist is not intended to be used directly by the customer, but is provided for reference. It is intended to be used only as an aid; you must be certain to read this entire manual, especially "Configuration Best Practices" on page 43 and Chapter 15, "Troubleshooting" on page 353, before attempting to complete these procedures.

[ ] Understand the application use, storage configuration, and I/O patterns at the site. (Not all applications benefit from CXFS; see "Comparison of XFS and CXFS" on page 3.)

[ ] Connect the SAN hardware. See the RAID documents.

[ ] Is there a private network? This is a requirement.

[ ] Is system reset configured for all potential metadata servers, as recommended by SGI? Is system reset or I/O fencing configured for all client-only nodes? One of these solutions is required to ensure data integrity for **all** nodes. See "Protect Data Integrity on All Nodes" on page 50.

[ ] Are there an odd number of server-capable administration nodes with CXFS services running? Is there a client-only tiebreaker node? See "Use an Odd Number of Server-Capable Administration Nodes" on page 48 and "Use a Client-Only Tiebreaker" on page 49.

[ ] Read this book, `README` files and release notes provided with the release, and the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

[ ] Verify that the network is usable.

[ ] Install the CXFS software. See Chapter 7, "Server-Capable Administration Node Installation" on page 109 and the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

[ ] Modify the configuration files and perform other tasks as needed. See Chapter 8, "Postinstallation Steps" on page 119.

[ ] Completely configure and run a **small** cluster (3 nodes). See Chapter 9, "Initial Setup of the Cluster" on page 127.

[ ] Look for errors in the daemon log files in the `/var/cluster/ha/logs` directory.

[ ] If all is well, add the rest of the nodes. If there are problems, see Chapter 15, "Troubleshooting" on page 353.

[ ] Set up the filesystems. See Chapter 9, "Initial Setup of the Cluster" on page 127.

Summary of New Features from Previous Releases

This appendix contains a summary of the new features for each version of this guide.

CXFS Version 1: Original Implementation

CXFS version 1 is the original implementation of CXFS.

IRIX 6.5.6f

Original publication (007–4016–001).

IRIX 6.5.6f

The 007–4016–002 update contains additional troubleshooting information and instructions for unmounting and remounting filesystems with the command line interface. It was reorganized to make the tasks of installation and configuration clearer.

IRIX 6.5.7f

The 007–4016–003 update contains the following:

- Metadata server recovery information
- Administrative shutdown procedures
- Additional troubleshooting information
- Instructions for unmounting and remounting filesystems with the CLI
- Reorganized installation and configuration information

IRIX 6.5.8f

The 007–4016–004 update contains the following:

- Support for hierarchical storage management (HSM) through data management application programming interface (DMAPI), also know as X/Open data storage management specification (XDSM)
- Changes to administrative shutdown, including two new `cmgr` subcommands to stop CXFS services on the local nodes: `admin cxfs_stop` and `admin cxfs_stop`
- Quorum changes without panics

IRIX 6.5.9f

The 007–4016–005 update contains the following:

- Coexecution of CXFS and IRIS FailSafe 2.1, including commands to convert nodes and clusters to apply to both utilities
- Ability to use the `cmgr` command without extra prompting (`-p`), permitting the use of scripts
- New tasks to revoke and allow membership of the local node
- Ability to specify the tie-breaker node, which is used in the process of computing node membership for the cluster when exactly half the nodes in the cluster are up and can communicate with each other
- Clarification that a single subnet should be used

IRIX 6.5.10f

The 007–4016–006 update contains the following:

- Clarifications about CXFS shutdown and database shutdown
- Additional information about CXFS daemons
- Clarifications to the comparison of XFS and CXFS

IRIX 6.5.11f

The 007–4016–007 update contains the following:

- Addition of the Origin 3000 partition ID to node configuration
- Troubleshooting information for a two-node cluster when both nodes go down
- Information about editing the `/etc/hosts` file to use an alternate interface for heartbeat and control.
- Clarification about the use of hardware reset and tie-breaker nodes
- Ability to unset the tie-breaker node
- Use of `fsr`

CXFS Version 2: MultiOS Cluster

CXFS version 2 includes client-only nodes on operating system platforms other than IRIX (*multiOS cluster*, or *heterogeneous clients*).

IRIX 6.5.12f

The 007–4016–008 update contains the following:

- A cluster of at least **three** weighted nodes is recommended for a production environment (that is, one requiring relocation and recovery of the metadata server).

  If you use a two-weighted-node cluster for production, you must do one of the following:

  – Use reset lines to avoid data corruption and ensure that only one node is running in error conditions (reset lines are recommended for all CXFS clusters and required for use with IRIS FailSafe).

  – Weight one node as 1 and the other as 0.

  – Set a tie-breaker node.

  However, there are issues with a two-weighted-node cluster.

- The new `cluster_status` command, which provides a `curses` interface to display status information gathered by the `cad` daemon (this information is also displayed by the `cxdetail` command).
- Cluster nodes can run adjacent levels of the IRIX operating system (OS); for example, 6.5.11f and 6.5.12f (this applies as of 6.5.12f).
- The ability to execute your own custom scripts around mounting operations.
- Partition ID information.
- Clarification about the following:
  - Hostname resolution rules; it is **critical** that you understand these rules and have files configured poperly before attempting to configure a cluster.
  - The difference between *CXFS membership* and `fs2d` *membership*.
  - Configuration of nodes supported in an IRIS FailSafe and CXFS coexecution environment.
  - Unwritten extent tracking (`unwritten=1|0`) with CXFS.
  - Including the CXFS mount point in the `/etc/exports` file.
  - Number of nodes supported: 16 CXFS nodes, and up to 8 IRIS FailSafe nodes with coexecution.
  - The flow of information in a coexecution cluster.

IRIX 6.5.13f

The 007–4016–009 update contains the following:

- The structure of the CXFS filesystem configuration has changed. CXFS filesystems can now be defined, modified, managed and deleted independently of each other and of the cluster definition. (Previously, the CXFS filesystems were defined as attributes to the cluster definition.)

  The new design improves the performance, flexibility and reliability of filesystem configuration. To accommodate clusters mixing nodes running 6.5.12 and 6.5.13, backwards compatibility is enforced by default in 6.5.13. The result is that the performance achievements are not visible; however, if you are 6.5.13 on all nodes in the cluster, you may wish to turn off backwards compatibility. Backwards compatibility will be turned off in the 6.5.14 release.

- Information about locating the `xfsdump` inventory in a shared directory.
- Information about the IRIS FailSafe `CXFS` resource type that can be used to failover applications that use CXFS filesystems.
- The `-p` option is no longer required when defining filesystems with the `cmgr` command; the scripting capability is therefore provided.
- For certain GUI tasks, the ability to select all nodes at once in addition to specifying nodes individually.
- New `/var/cluster/cmgr-scripts/rotatelogs` script to save log files with day and month name as suffixes.
- The setting to force an unmount of a filesystem using the `umount -k` option is turned off by default in the GUI. There is no default when using the `cmgr` command.
- Clarification of the term *CLI* to mean the underlying set of commands that are used by the `cmgr` cluster manager tool and by the GUI.
- Use of `sgi_apache.sw.server`.
- Correction: real-time filesystems are not currently supported. Changes to reflect this have been made in text.
- New and revised figures.

IRIX 6.5.14f

The 007–4016–011 update contains the following:

- The graphical user interface (GUI) has been improved. The separate cluster view (the `cxdetail` command) and task manager (the `cxtask` command) have been streamlined into one window, the **CXFS Manager**. Both the `cxtask` and `cxdetail` commands are kept for historical purposes; this document refers to just `cxtask` for simplicity.

  The new GUI provides the following features:

  – Access to tasks through the menu bar or by clicking the right mouse button within the tree view

  – Faster filesystem status and cluster status updates

  - Access to the `salog`(4) file, which shows every command run from the GUI
  - A **Find** textfield helps you find components within the displayed tree-view
- Information about the use of `xfs_repair` and CXFS filesystems.

Caution: Do not use `xfs_repair` on a CXFS filesystem unless you are certain there is a problem.

- Information about using `cmgr`(1M):
  - Invoking subcommands directly on the command line with the `-c` option
  - Using template scripts provided in the `/var/cluster/cmgr-templates` directory
- Information about MAC labels in a mixed Trusted IRIX and IRIX cluster.
- The structure of the CXFS filesystem configuration was changed with the release of IRIX 6.5.13. Backward compatibility with earlier versions is no longer maintained as of IRIX 6.5.14, because all nodes in the cluster must be running the same or adjacent releases.

  If you are upgrading from 6.5.13f entirely to 6.5.14f, there is no further impact.

  If you intend to run a mixture of 6.5.13f and 6.5.14f nodes, you must turn off backward compatibility.

  If you are upgrading from 6.5.12f or earlier without first installing and running 6.5.13f, then you must perform a one-time manual conversion of your CXFS filesystem definitions.

IRIX 6.5.15f

The 007–4016–012 update contains the following:

Note: Relocation and recovery are deferred in this release.

- Support for clients of other operating systems such as Solaris and Windows NT as defined in the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*. These clients will be released asynchronously from the IRIX release. This support will require a minimum of IRIX 6.5.15f plus appropriate patches. For more information, see your SGI support contact.

- Default scripts are now provided in the `/var/cluster/clconfd-scripts` directory to permit NFS-exporting of CXFS filesystems listed in `/etc/exports`.
- Reset lines are mandatory for two-node and two-weighted node clusters. Larger clusters should have an odd number of weighted nodes, or must have serial reset lines if only two of the nodes are weighted.
- Simplification of Chapter 1. General information about the CXFS Manager GUI and `cmgr` have been moved to their respective reference chapters, coexecution details have been moved into a separate chapter, and the communication flow diagrams and daemon information have been moved into an appendix.
- Information about the error messages that may cause administrators to use `xfs_repair` inappropriately.
- Changes to the `rotatelogs` script syntax. The `root crontab` file now has an entry to run the `rotatelogs` script weekly. If you run the script twice in one day, it will append the current log file to the previous saved copy, rather than overwriting it.
- A new figure describing some of the various combinations of node and cluster types in a coexecution cluster.

IRIX 6.5.16f

The 007–4016–013 update contains the following:

Note: Relocation and recovery are fully implemented, but the number of associated problems prevents support of these features in CXFS. While data integrity is not compromised, cluster node panics or hangs are likely to occur. These features will be fully supported when these issues are resolved.

- Support for Solaris and Windows NT systems in a multiple operating system (multiOS) cluster, including the following:
  – Information about defining the operating system for a node. For existing clusters that are upgraded to IRIX 6.5.16f, existing nodes will be assigned an operating system type of `IRIX`.

  - Information about I/O fencing, which allows a problem node to be isolated from the storage area network (SAN) so that it cannot corrupt data in the shared CXFS filesystem. Solaris and Windows NT nodes require a Brocade switch in order to support I/O fencing for data integrity protection; therefore, the Brocade switch is a required piece of hardware in a cluster running multiple operating systems.
  - The new terms *multiOS* and *CXFS client-only node*.
- Support for the L1 controller on SGI Origin 300, SGI Origin 3200C, SGI Onyx 300, and SGI Onyx 3200C systems.
- Information about the CXFS GUI tasks to define and modify a filesystem, which have been split into two pages for ease of use.
- New GUI icons.

IRIX 6.5.17f

The 007–4016–014 update contains the following:

- A new appendix contains an example `/etc/ipfilterd.conf` file that can be used to provide IP filtering for the CXFS private network.
- The `build_cmgr_script` command, which generates a `cmgr` script from the cluster database. The script can be used later to recreate the cluster database after performing a `cdbreinit` command.
- A sample script to unexport and locally unmount an `lofs` filesystem.
- Use of the new command name `cxfsmgr`. The `cxfsmgr` command has the same function as the `cxtask` and `cxdetail` commands, which are kept for historical purposes.
- Clarifications to the following:
  - Starting the CXFS Manager graphical user interface
  - Masking and I/O fencing
  - Terminology such as *cluster*, *node*, and *pool*
  - Terminology used to describe the GUI

IRIX 6.5.18f

The 007–4016–015 update contains the following:

Note: In this release, relocation is disabled by default and recovery is supported only when using standby nodes.

A *standby node* is a metadata server-capable administration node that is configured as a potential metadata server for a given filesystem, but does not currently run any applications that will use that filesystem. To use recovery, you must not run any applications on any of the potential metadata servers for a given filesystem; after the active metadata server has been chosen by the system, you can then run applications that use the filesystem on the active metadata server and client-only nodes.

Relocation and recovery are fully implemented, but the number of associated problems prevents full support of these features in the current release. Although data integrity is not compromised, cluster node panics or hangs are likely to occur. Relocation and recovery will be fully supported in a future release when these issues are resolved.

- IRIX nodes may now be CXFS client-only nodes, meaning that they run a minimal implementation of the CXFS and cluster services, and do not contain a copy of the CXFS cluster database. Client-only nodes are installed with the `cxfs_client` software product.

  This change also introduces the term *CXFS administration node*, which is a node that is installed with the `cluster_admin` software product, allowing the node to perform cluster administration tasks and contain a copy of the cluster database. Nodes that you want to run as metadata servers must be installed as CXFS server-capable administration nodes; SGI recommends that all other nodes be installed as client-only nodes.

  When you define a node, you no longer need to specify the node weight. This has been replaced by the **Node Function** field, allowing you to choose **Server-capable Admin**, **Client Admin**, or **Client-Only**. (For Solaris and Windows nodes, **Client-Only** is automatically selected for you.) Similar fields are provided for the `cmgr` command.

  When upgrading to 6.5.18f, already existing IRIX nodes will by default be assigned as **Server-capable Admin** if they had a weight of 1.

  This version also clarifies the terms used for membership: *CXFS kernel membership* and *cluster database membership*.

- New system-tunable parameters:
  - `cxfs_relocation_ok` lets you enable or disable the relocation feature; relocation is disabled by default in this release, and SGI recommends that you **do not** enable it.
  - `cxfsd_min` and `cxfsd_max` let you specify the minimum and maximum number of `cxfsd` threads to run per CXFS filesystem.
- New commands:
  - `cxfs_info` provides status information about the cluster, nodes, and filesystems and is run from a client-only node.
  - `cxfsdump` gathers CXFS configuration information.
- A CXFS cluster is supported with as many as 32 nodes. As many as 16 of those nodes can be CXFS administration nodes and all other nodes can be client-only nodes. You can choose to define a node as a CXFS client administration node, however, SGI strongly recommends that only potential metadata servers be configured as CXFS server-capable administration nodes and that there be an odd number of server-capable administration nodes for quorum calculation purposes.
- The graphical user interfaces for XVM and CXFS have been combined into one. This guide provides an overview of the XVM-specific tasks provided by the GUI; for details about these tasks, see the *XVM Volume Manager Administrator's Guide*.

  The tasks to make, grow, mount/unmount a filesystem are now provided in the GUI.
- Tips about using CXFS and Trusted IRIX.
- Support for Microsoft Windows 2000 systems as client-only nodes. (This guide uses *Windows* to refer to both Microsoft Windows NT and Microsoft Windows 2000 nodes when the information applies equally to both. Information that applies to only one of these types of nodes is identified.)

IRIX 6.5.19f

The 007–4016–016 update contains the following:

- The new rolling annual upgrade policy that permits you to upgrade from 6.5.*n* to the *n*+1 or *n*+4 release, as of 6.5.18f.

- The time required to update and propagate the database across nodes in the cluster has been significantly decreased.
- If you use I/O fencing and `ipfilterd` on a node, the `ipfilterd` configuration must allow communication between the node and the `telnet`(1) port on the switch.
- The following nodes do not contain system controllers and therefore require I/O fencing for data integrity protection:
  - Silicon Graphics Fuel visual workstation
  - Silicon Graphics Octane system
  - Silicon Graphics Octane2 system
- The CXFS Manager graphical user interface (GUI) has added a new icon to represent client-only nodes.
- In preparation for future CXFS MultiOS client releases, the CXFS software now also allows you to specify the Linux, IBM AIX, and Hewlett-Packard HP-UX operating systems when defining a node. For support details, see the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage* and release notes.
- This version clarifies the various methods to perform cluster database backups and restorations.
- Application programmers should be aware that XFS recently relaxed the requirement that direct I/O be aligned along filesystem block boundaries. As of IRIX 6.5.19f, direct I/O will also be accepted using 512-byte alignment.

  This change makes the use of direct I/O on a CXFS partition more consistent with that of other vendor's requirements and thus makes the use of CXFS more transparent. See the description of direct I/O requirements in the `fcntl` man page.
- This version lists the system tunable parameters found in the `/var/sysgen/mtune/cell` file, some of which should not be modified.

IRIX 6.5.20f

The 007–4016–017 update contains the following:

- Changes to the CXFS graphical user interface (GUI):
  - New login connection choices, including support for a remote shell connection, which connects to the server via a user-specified command shell, such as `rsh` or `ssh`.
  - The ability for the `root` user to grant other users permission to execute specific GUI tasks.
  - Use of Java2 for the CXFS GUI, which simplifies the Java installation and co-operation with third-party GUIs. This also enhances the ability to run the GUI through a web browser (via http://*server*/CXFSManager/).
  - Information about using the right mouse button to access tasks appropriate to the selected GUI item.
- Changes to the `cxfsd_min` and `cxfsd_max` defaults, and the `cxfsd_max` legal values.
- More information about memberships, quorums, and tiebreakers.
- A new figure describing standby mode.
- More information about IRIX client-only issues:
  - Client-only node system files
  - Status in log files
  - `cxfs_client` error messages

CXFS Version 3: IRIX or SGI ProPack (Linux 2.4 Kernel) Servers

CXFS version 3 adds support for CXFS metadata servers on SGI Altix systems running Linux (2.4 kernel).

CXFS 3.0

The 007–4016–018 update contains the following:

- Support for SGI ProPack metadata servers on SGI Altix 3000 family of servers and superclusters. A CXFS cluster can contain either SGI ProPack 2.3 server-capable administration nodes on Altix systems or IRIX server-capable administration nodes; you cannot mix IRIX and SGI ProPack server-capable administration nodes within one cluster.

  CXFS does not support the relocation or recovery of DMAPI filesystems that are being served by SGI ProPack metadata servers.

  Coexecution with FailSafe is not supported on SGI ProPack nodes.

- Due to packaging enhancements, CXFS may now be installed on the M stream or the F stream.

  The IRIX CXFS software will no longer be bundled in the IRIX overlay CDs but instead is on a separate *CXFS IRIX Server and Client 3.0 for IRIX 6.5.22* CD. This changes the installation procedure.

  Note: If you are upgrading from a previous IRIX release and have CXFS installed, you must upgrade both IRIX and CXFS. If you try to upgrade one without the other, conflicts will occur.

- Information about defining networks for CXFS kernel messaging (in addition to the network used for heartbeat/control). However, use of these networks is **deferred**.
- Support for IRIX real-time filesystems.
- Suggestions for configuring large clusters.
- Information about using `ping` to verify general connectivity and CXFS heartbeat in a multicast environment.
- The GUI has been changed to show a single display for the nodes in the cluster and nodes that are in the pool but not in the cluster. This new selection is **View: Nodes and Cluster**.
- Information about information retaining system core files and the output from the `cxfsdump` utility when reporting problems.

- Information about monitoring heartbeat timeouts for IRIX using Performance Co-Pilot or the `icrash` command.
- The ability to define multiple CXFS filesystems at one time with the GUI.

CXFS 3.1

The 007–4016–019 update contains the following:

- Information about migrating from an IRIX cluster to a SGI ProPack cluster
- Support for a cluster of up to 64 nodes.
- Information about the TP9300 RAID.
- Information about the `cxfs-config` command.
- Clarification that serial hardware reset lines or I/O fencing is **required for all nodes** in order to protect data integrity.
- The ability to define a reset method for a given node to one of the following:
  - `powerCycle` to turn power off and on
  - `reset` to perform a serial reset
  - `nmi` to perform a nonmaskable interrupt

  You can define this method using either the `cmgr` command or the GUI. You can manually perform a powercycle or an NMI with the `cmgr` command.
- New appendixes summarizing operating system path differences and new features from previous releases

CXFS 3.2

The 007–4016–021 update contains the following:

- Information about private network failover as defined with the `cmgr` command. (Although the primary network must be private, the backup network may be public.)
- Support for Guaranteed-rate I/O version 2 (GRIOv2) in the IRIX installation procedure.

- Corrections to CXFS and cluster administration path differences between IRIX and SGI ProPack on SGI Altix systems.
- Updated the example for `clconf_info` command. The `clconf_info` command now reports a node as `inactive` rather than `DOWN*` and the unused incarnation number has been removed.
- Support for token obtain optimization. To disable, use the `cxfs_prefetch` system tunable parameter.
- If you have a cluster with an even number of server-capable administration nodes and no tiebreaker: to avoid a split-brain scenario, you should not use the **Shutdown** setting on any server-capable administration node.
- Information about multiple Ethernet interfaces on SGI Altix systems and providing persistent device naming.
- Clarification about the `chkconfig` arguments used for IRIX administration nodes, SGI ProPack administration nodes, and client-only nodes.
- Information about the correct options to use for quotas on SGI ProPack clusters (`uquota` and `gquota`).
- Information about serial reset configurations.
- Information about using the `hafence`(1M) command to define a QLogic switch. (You cannot use the GUI or the `cmgr` command to define or modify a switch other than a Brocade switch.)
- If you want to use quotas on a CXFS filesystem, you must install the quota package.
- Information about removing a metadata server from the cluster for maintenance.
- Mount options information.
- Addition of the XVM graphical user interface (GUI) to the CXFS SGI ProPack package.

CXFS 3.3

The 007–4016–022 update contains the following:

- Procedures to remove a single client from the cluster and restore it, and shut down the entire cluster.

- A new chapter about best practices.
- Information about XVM failover.
- Updates to the rolling upgrade policy.
- Information about performing a miniroot install.
- Information about installing the latest Java2 software for use with the CXFS GUI and SGI ProPack.
- Information about modifying the `httpd.conf` file in order to use the CXFS GUI on SGI ProPack.
- Clarifications to terminology.

CXFS 3.4

The 007–4016–023 update contains the following:

- Information about discoverying the WWNs
- Support for system reset for SGI Altix systems that use an integrated L2, such as a NUMAlink 4 R-brick, or SGI Altix 3000 Bx2 systems. Configuring a node of this type requires use of `cmgr`.
- Support for SGI ProPack for Linux 4 as a client-only node.
- Support for the `cxfsdump -secure` option for secure remote connections. In cluster mode (the default), `cxfsdump` requires `rsh/ssh` and `rcp/scp` access across all nodes in the cluster.

CXFS Version 4: IRIX or SGI ProPack (Linux 2.6 Kernel) Servers

CXFS version 4 adds support for CXFS metadata servers on SGI Altix systems running Linux 2.6 kernel.

CXFS 4.0

The 007–4016–024 update contains the following:

- Support for SGI ProPack 4 for Linux Service Pack 3.

- Support for IRIX 6.5.29.
- Server-side CXFS client license keys are now supported on server-capable administration nodes, allowing a client without a node-locked client-side license key to request a license key from the server. Server-side license keys are optional on IRIX metadata servers, but are required on SGI ProPack metadata servers. The licensing software is based on the FLEXlm product from Macrovision Corporation.
- Support for the `cxfs_admin` cluster configuration and administration command, which waits for a command to be completed before continuing, provides the ability to limit which nodes can mount a filesystem, provides an improved interface over `cmgr` (including `<TAB>` completion of commands), and scripting capabilities. In a future release, `cxfs_admin` will replace `cmgr`.
- The availability of dynamic heartbeat monitoring.
- Information about choosing the correct version of XVM failover for your cluster.
- Support for an SGI ProPack node as a server for the Guaranteed Rate I/O version 2 (GRIOv2) feature of CXFS.
- Support for GPT labels.
- Information about using the `md` driver on large SGI Altix systems with CXFS.
- Updates to IRIX CXFS installation procedures.
- Support for specifying a direct list of port numbers to be masked (that is, ports that will never be fenced), rather than using a hexadecimal string that represents the list. Ports that can be masked are now in the range 0 through 1023.
- Ability to specify a vendor when defining a Fibre Channel switch using the GUI.
- Requirement to start the file alteration monitoring (`fam`) service on SGI ProPack nodes in order to use the CXFS graphical user interface (GUI).
- Information about switching between `SGIRDAC` and `SGIAVT` mode for SGI RAID.
- Information about CXFS port usage. CXFS Port Usage.
- Information about the recovery timeout mechanism, in which nodes are polled for progress after a recovery has begun. If recovery for a node is not progressing according to the specified polls, the recovery is considered stalled and the node will then shut down or panic; this prevents the cluster from hanging and keeps filesystems available for the rest of the nodes in the cluster. The following site-changeable system tunable parameters are used:

```
cxfs_recovery_timeout_panic
cxfs_recovery_timeout_period
cxfs_recovery_timeout_stalled
cxfs_recovery_timeout_start
```

- Information about using the `filestreams` mount option to optimize disk layout.
- Best-practices information about:
  - Using proper storage management procedures
  - Being aware of differences between IRIX and Linux system administration
  - Modifying `updatedb` to avoid unnecessary load
  - Using fast copying for large CXFS files
  - Minimizing the number of switches

  Note: The contents of the former "Avoid Problems" section from the Troubleshooting chapter was moved into the Best Practices chapter.

- Addition of SGI RAID information (this information was removed from the release notes).
- Addition of switches information (this information was removed from the release notes).
- A complete list of system-tunable parameters.
- Support for drag-and-drop within the GUI to move nodes between the pool and the cluster.
- Client administration nodes are only supported and appropriate for IRIX nodes running in coexecution with FailSafe. If you are running an earlier release and have an SGI ProPack node that is defined as a client administration node, you must delete the node from the cluster database, uninstall CXFS, reinstall CXFS, and redefine the node as a client-only node. For simplicity, this guide no longer refers to *client-administration node* or *administration node* except in specific instances.
- For clarity, this book now uses the term *SGI ProPack* rather than *Linux* when referring to CXFS nodes running SGI ProPack for Linux.

CXFS 4.1

The 007–4016–025 version includes the following:

- Support for SGI IRIX 6.5.30.
- Support for SGI ProPack 5 running SUSE Linux Enterprise Server 10 (SLES 10).
- Support for SGI License Key (LK) software on SGI ProPack server-capable administration nodes.

  Server-side licensing is required on the following client-only nodes (to determine the Linux architecture type, use the `uname -i` command):

  – SGI ProPack 5

  – Red Hat Enterprise Linux (RHEL) 4 on x86_64

  – SLES 9 on x86_64

  – SLES 10 on x86_64 or ia64

  Note: For specific release levels, see the release notes.

  Other client-only nodes can use either server-side or client-side licensing. However, if one node within a cluster requires server-side licensing, all nodes must use server-side licensing. If no nodes in the cluster require server-side licensing, the nodes can continue to use existing client-side licensing.

  Note: Server-side licensing is preferred, and no new client-side licenses will be issued. Customers with support contracts can exchange their existing client-side licenses for new server-side licenses. A future release will not support client-side licensing. For more information, contact SGI customer support.

- Support for the following RAID hardware:

  SGI InfiniteStorage 10000
  SGI InfiniteStorage 6700
  SGI InfiniteStorage 4500
  SGI InfiniteStorage 4000

- DMAPI is disabled by default on SGI ProPack 5 systems. When you install DMF on a server-capable administration node, it automatically enables DMAPI.

However, if you want to mount filesystems on an SGI ProPack 5 client-only node with the `dmi` mount option, you must enable DMAPI.

- Information about the appropriate use of `lcrash` for SGI ProPack.
- Information about the procedure required when upgrading from CXFS 3.4.1 or earlier for clusters with three or more server capable nodes.
- Information about removing a metadata server from the cluster while avoiding a reset.
- New `mtcp_rpc_thread` system tunable parameter (for SGI ProPack and Linux third-party nodes) that specifies whether metadata messages are sent from a separate thread in order to save stack space.
- Information about rotating log files on SGI ProPack.
- Information about `SYSLOG credid` warnings.
- The table of mount options is now located in the *CXFS MultiOS Client-Only Guide for SGI InfiniteStorage*.

CXFS 4.2

The 007–4016–026 version includes the following:

- Support for the SGI InfiniteStorage 220 RAID.
- Support for Intelligent Platform Management Interface (IPMI) reset using a baseboard management controller (BMC).
- As of CXFS 4.2, all server-capable administration nodes running 4.2 and client-only nodes running 4.2 require server-side licensing. If **all** existing client-only nodes are running a prior supported release, they may continue to use client-side license as part of the rolling upgrade policy until they are upgraded to 4.2. All client-only nodes in the cluster must use the same licensing type — if any client-only node in the cluster is upgraded to 4.2 or if a new 4.2 client-only node is added, then all nodes must use server-side licensing.

  Customers with support contracts can exchange their existing client-side licenses for new server-side licenses. For more information, contact SGI customer support.
- Support for GPT-labeled LUNs larger than 2 TB. (All nodes that mount a filesystem using LUNs larger than 2 TB must be upgraded to CXFS 4.2 or later.)

- Disk layout optimization for approved media customers
- If you have multiple clusters using the same public network as the backup CXFS metadata network, use the `-i` option to `cxfs_admin` to identify the cluster name.
- Precedence of configuration options
- Support for printing `hafence` debug information to the specified file *debugfile* by using the `-d` option in the `/etc/cluster/config/clconfd.options` file.
- Using `cxfs-reprobe` on client-only nodes (SGI ProPack)
- Information about parameters that must be set for QLogic switches.
- The ability to use environment variables or the `.cxfs_admin` file to specify defaults for `cxfs_admin`, in addition to the `set` command.
- Documentation for the support of XVM failover version 2 on Windows (first supported in the CXFS 4.1.1 release).
- A new section that describes how to view the current CXFS licenses with the `cxfs_admin` command.
- `clconfd.options` on CXFS Administration Nodes
- Information about the `cmgr` command has been moved to an appendix. With the exception of performing the following administrative `cmgr`commands, the preferred CXFS configuration tools are `cxfs_admin` and the CXFS graphical user interface (GUI):

  ```
  admin ping
  admin reset
  admin powerCycle
  ```

  As of the CXFS 5.0 release, this functionality will be provided by the `cxfs_admin` command and the `cmgr` command will not be supported.

Glossary

active metadata server

A server-capable administration node chosen from the list of potential metadata servers. There can be only one active metadata server for any one filesystem. See also *metadata*.

ACL

Access control list.

administration node

See *server-capable administration node*.

administrative stop

See *forced CXFS shutdown*.

ARP

Address resolution protocol.

bandwidth

Maximum capacity for data transfer.

blacklisted

A node that is explicitly not permitted to be automatically configured into the cluster database.

BMC

Baseboard management controller.

cell ID

A number associated with a node that is allocated when a node is added into the cluster definition with the GUI or `cxfs_admin`. The first node in the cluster has cell ID of 0, and each subsequent node added gets the next available (incremental) cell ID.

If a node is removed from the cluster definition, its cell ID becomes available. It is not the same thing as the *node ID*.

CLI

Underlying command-line interface commands used by the CXFS Manager graphical user interface (GUI).

client

In CXFS, a node other than the active metadata server that mounts a CXFS filesystem. A *server-capable administration node* can function as either an active metadata server or as a CXFS client, depending upon how it is configured and whether it is chosen to be the active metadata server. A *client-only node* always functions as a client.

client-only node

A node that is installed with the `cxfs_client.sw.base` software product; it does not run cluster administration daemons and is not capable of coordinating CXFS metadata. Any node can be client-only node. See also *server-capable administration node*.

cluster

A *cluster* is the set of systems (nodes) configured to work together as a single computing resource. A cluster is identified by a simple name and a cluster ID. A cluster running multiple operating systems is known as a *multiOS cluster*.

There is only one cluster that may be formed from a given pool of nodes.

Disks or logical units (LUNs) are assigned to clusters by recording the name of the cluster on the disk (or LUN). Thus, if any disk is accessible (via a Fibre Channel connection) from machines in multiple clusters, then those clusters must have unique names. When members of a cluster send messages to each other, they identify their cluster via the cluster ID. Cluster names must be unique.

Because of the above restrictions on cluster names and cluster IDs, and because cluster names and cluster IDs cannot be changed once the cluster is created (without deleting the cluster and recreating it), SGI advises that you choose unique names and cluster IDs for each of the clusters within your organization.

cluster administration daemons

The set of daemons on a server-capable administration node that provide the cluster infrastructure: `cad`, `cmond`, `fs2d`, `crsd`. See "CXFS Control Daemons" on page 41.

cluster administration tools

The CXFS graphical interface (GUI) and the `cxfs_admin` command-line tools that let you configure and administer a CXFS cluster, and other tools that let you monitor the state of the cluster. See "CXFS Tools" on page 36.

cluster administrator

The person responsible for managing and maintaining a cluster.

cluster database

Contains configuration information about all nodes and the cluster. The database is managed by the cluster administration daemons.

cluster domain

XVM concept in which a filesystem applies to the entire cluster, not just to the local node. See also *local domain*.

cluster database membership

The group of server-capable administration nodes in the **pool** that are accessible to cluster administration daemons and therefore are able to receive cluster database updates; this may be a subset of the nodes defined in the pool. The cluster administration daemons manage the distribution of the cluster database (CDB) across the server-capable administration nodes in the pool. (Also known as *user-space membership* and *fs2d database membership*.)

cluster ID

A unique number within your network in the range 1 through 255. The cluster ID is used by the operating system kernel to make sure that it does not accept cluster information from any other cluster that may be on the network. The kernel does not use the database for communication, so it requires the cluster ID in order to verify cluster communications. This information in the kernel cannot be changed after it has been initialized; therefore, you must not change a cluster ID after the cluster has been defined. Clusters IDs must be unique.

cluster mode

One of two methods of CXFS cluster operation, `Normal` or `Experimental`. In `Normal` mode, CXFS monitors and acts upon CXFS kernel heartbeat or cluster database heartbeat failure; in `Experimental` mode, CXFS ignores heartbeat failure. `Experimental` mode allows you to use the kernel debugger (which stops heartbeat) without causing node failures. You should only use `Experimental` mode during debugging with approval from SGI support.

control messages

Messages that the cluster software sends between the cluster nodes to request operations on or distribute information about cluster nodes. Control messages, CXFS kernel heartbeat messages, CXFS metadata, and cluster database heartbeat messages are sent through a node's network interfaces that have been attached to a private network.

cluster node

A node that is defined as part of the cluster. See also *node*.

control network

See *private network*.

CXFS

Clustered XFS, a clustered filesystem for high-performance computing environments. See "What is CXFS?" on page 1

CXFS client daemon

The daemon (`cxfs_client`) that controls CXFS services on a client-only node.

CXFS control daemon

The daemon (`clconfd`) that controls CXFS services on a server-capable administration node. See "Cluster Administration Daemons" on page 40.

CXFS database

See *cluster database*.

CXFS kernel membership

The group of CXFS nodes that can share filesystems in the cluster, which may be a subset of the nodes defined in a cluster. During the boot process, a node applies for CXFS kernel membership. Once accepted, the node can share the filesystems of the cluster. (Also known as *kernel-space membership*.) CXFS kernel membership differs from *cluster database membership*.

CXFS services

The enabling/disabling of a node, which changes a flag in the cluster database. This disabling/enabling does not affect the daemons involved. The daemons that control CXFS services are `clconfd` on a server-capable administration node and `cxfs_client` on a client-only node. See "CXFS Services" on page 28.

CXFS services start

To enable a node, which changes a flag in the cluster database, by using an administrative task in the CXFS GUI or the `cxfs_admin enable` command.

CXFS services stop

To disable a node, which changes a flag in the cluster database, by using the CXFS GUI or the `cxfs_admin disable` command. See also *forced CXFS shutdown*.

CXFS shutdown

See *forced CXFS shutdown* and *shutdown*.

CXFS tiebreaker node

A node identified as a tiebreaker for CXFS to use in the process of computing CXFS kernel membership for the cluster, when exactly half the nodes in the cluster are up and can communicate with each other. There is no default CXFS tiebreaker. SGI recommends that the tiebreaker node be a client-only node.

database

See *cluster database*.

database membership

See *cluster database membership*.

details area

The portion of the GUI window that displays details about a selected component in the view area. See also *view area*.

domain

See *cluster domain* and *local domain*.

dynamic heartbeat monitoring

Starts monitoring CXFS kernel heartbeat only when an operation is pending. Once monitoring initiates, it monitors at 1-second intervals and declares a timeout after 5 consecutive missed seconds, just like *static heartbeat monitoring*.

DVH

Disk volume header.

fail policy hierarchy

See *fail policy*.

failure policy

The set of instructions that determine what happens to a failed node; the second instruction will be followed only if the first instruction fails; the third instruction will be followed only if the first and second fail. The available actions are: *fence*, *fencereset*, *reset*, and *shutdown*.

fence

The failure policy method that isolates a problem node so that it cannot access I/O devices, and therefore cannot corrupt data in the shared CXFS filesystem. I/O fencing can be applied to any node in the cluster (CXFS clients and metadata servers). The rest of the cluster can begin immediate recovery.

fencereset

The failure policy method that fences the node and then, if the node is successfully fenced, performs an asynchronous system reset; recovery begins without waiting for reset acknowledgment. If used, this fail policy method should be specified first. If the fencing action fails, the reset is not performed; therefore, `reset` alone is also highly recommended for all server-capable administration nodes (unless there is a single server-capable administration node in the cluster).

fencing recovery

The process of recovery from fencing, in which the affected node automatically withdraws from the CXFS kernel membership, unmounts all filesystems that are using an I/O path via fenced HBA(s), and then rejoins the cluster.

forced CXFS shutdown

The withdrawl of a node from the CXFS kernel membership, either due to the fact that the node has failed somehow or by issuing an `admin cxfs_stop` command. This disables filesystem and cluster volume access for the node. The node remains enabled in the cluster database. See also *CXFS services stop* and *shutdown*.

fs2d database membership

See *cluster database membership*.

gratuitous ARP

ARP that broadcasts the MAC address to IP address mappings on a specified interface.

GUI

Graphical user interface. The CXFS GUI lets you set up and administer CXFS filesystems and XVM logical volumes. It also provides icons representing status and structure. See Chapter 10, "Reference to GUI Tasks" on page 147.

GPT

GUID partition table

heartbeat messages

Messages that cluster software sends between the nodes that indicate a node is up and running. CXFS kernel heartbeat messages, cluster database heartbeat messages, CXFS metadata, and control messages are sent through the node's network interfaces that have been attached to a private network.

heartbeat timeout

If no CXFS kernel heartbeat or cluster database heartbeat is received from a node in this period of time, the node is considered to be dead. The heartbeat timeout value must be at least 5 seconds for proper CXFS operation.

I/O fencing

See *fence*.

IPMI

Intelligent Platform Management Interface.

ISSP

SGI Infinite Storage Software Platform, the distribution method for CXFS software.

kernel-space membership

See *CXFS kernel membership*.

LAN

Local area network.

local domain

XVM concept in which a filesystem applies only to the local node, not to the cluster. See also *cluster domain*.

log configuration

A log configuration has two parts: a *log level* and a *log file*, both associated with a *log group*. The cluster administrator can customize the location and amount of log output, and can specify a log configuration for all nodes or for only one node. For example, the `crsd` log group can be configured to log detailed level-10 messages to the `crsd-foo` log only on the node `foo` and to write only minimal level-1 messages to the `crsd` log on all other nodes.

log file

A file containing notifications for a particular *log group*. A log file is part of the *log configuration* for a log group.

log group

A set of one or more CXFS processes that use the same log configuration. A log group usually corresponds to one daemon, such as `gcd`.

log level

A number controlling the number of log messages that CXFS will write into an associated log group's log file. A log level is part of the log configuration for a log group.

logical volume

A logical organization of disk storage in XVM that enables an administrator to combine underlying physical disk storage into a single unit. Logical volumes behave like standard disk partitions. A logical volume allows a filesystem or raw device to be larger than the size of a physical disk. Using logical volumes can also increase disk I/O performance because a volume can be striped across more than one disk. Logical volumes can also be used to mirror data on different disks. For more information, see the *XVM Volume Manager Administrator's Guide*.

LUN

Logical unit. A logical disk provided by a RAID. A logical unit number (LUN) is a representation of disk space. In a RAID, the disks are not individually visible because they are behind the RAID controller. The RAID controller will divide up the total disk space into multiple LUNs. The operating system sees a LUN as a hard disk. A LUN is what XVM uses as its physical volume (*physvol*). For more information, see the *XVM Volume Manager Administrator's Guide*.

membership

See *cluster database membership* and *CXFS kernel membership*.

membership version

A number associated with a node's cell ID that indicates the number of times the CXFS kernel membership has changed since a node joined the membership.

metadata

Information that describes a file, such as the file's name, size, location, and permissions.

metadata server

The server-capable administration node that coordinates updating of metadata on behalf of all nodes in a cluster. There can be multiple potential metadata servers, but only one is chosen to be the active metadata server for any one filesystem.

metadata server recovery

The process by which the metadata server moves from one node to another due to an interruption in CXFS services on the first node. See also *recovery*.

multiOS cluster

A cluster that is running multiple operating systems, such as SGI ProPack and Solaris.

multiport serial adapter cable

A device that provides four DB9 serial ports from a 36-pin connector.

node

A *node* is an operating system (OS) image, usually an individual computer. (This use of the term *node* does not have the same meaning as a node in an SGI Origin 3000 or SGI 2000 system and is different from the NUMA definition for a brick/blade on the end of a NUMAlink cable.)

A given node can be a member of only one pool (and therefore) only one cluster.

See also *client-only node*, *server-capable administration node*, and *standby node*.

node ID

An integer in the range 1 through 32767 that is unique among the nodes defined in the pool. You must not change the node ID number after the node has been defined. It is not the same thing as the *cell ID*.

node membership

The list of nodes that are active (have CXFS kernel membership) in a cluster.

notification command

The command used to notify the cluster administrator of changes or failures in the cluster and nodes. The command must exist on every node in the cluster.

owner host

A system that can control a node remotely, such as power-cycling the node. At run time, the owner host must be defined as a node in the pool.

owner TTY name

The device file name of the terminal port (TTY) on the *owner host* to which the system controller is connected. The other end of the cable connects to the node with the system controller port, so the node can be controlled remotely by the owner host.

peer-to-disk

A model of data access in which the shared files are treated as local files by all of the hosts in the cluster. Each host can read and write the disks at near-local disk speeds; the data passes directly from the disks to the host requesting the I/O, without passing through a data server or over a LAN. For the data path, each host is a peer on the SAN; each can have equally fast direct data paths to the shared disks.

physvol

Physical volume. A disk that has been labeled for use by XVM. For more information, see the *XVM Volume Manager Administrator's Guide*.

pool

The set of nodes from which a particular cluster may be formed. Only one cluster may be configured from a given pool, and it need not contain all of the available nodes. (Other pools may exist, but each is disjoint from the other. They share no node or cluster definitions.)

A pool is formed when you connect to a given node and define that node in the cluster database using the CXFS GUI. You can then add other nodes to the pool by defining them while still connected to the first node, or to any other node that is already in the pool. (If you were to connect to another node and then define it, you would be creating a second pool).

port password

The password for the system controller port, usually set once in firmware or by setting jumper wires. (This is not the same as the node's root password.)

potential metadata server

A server-capable administration node that is listed in the metadata server list when defining a filesystem; only one node in the list will be chosen as the active metadata server.

private network

A network that is dedicated to CXFS kernel heartbeat messages, cluster database heartbeat messages, CXFS metadata, and control messages. The private network is accessible by administrators but not by users. Also known as *control network*.

quorum

The number of nodes required to form a cluster, which differs according to membership:

- For CXFS kernel membership:
  - A majority (>**50%**) of the server-capable administration nodes in the cluster are required to **form** an initial membership
  - Half (**50%**) of the server-capable administration nodes in the cluster are required to **maintain** an existing membership
- For cluster database membership, **50%** of the **nodes in the pool** are required to form and maintain a cluster.

quorum master

The node that is chosen to propagates the cluster database to the other server-capable administration nodes in the pool.

RAID

Redundant array of independent disks.

recovery

The process by which a node is removed from the CXFS kernel membership due to an interruption in CXFS services. It is during this process that the remaining nodes in the CXFS kernel membership resolve their state for cluster resources owned or shared with the removed node. See also *metadata server recovery*.

relocation

The process by which the metadata server moves from one node to another due to an administrative action; other services on the first node are not interrupted.

reset

The failure policy method that performs a system reset via a serial line connected to the system controller. The reset may be a powercycle, serial reset, or NMI (nonmaskable interrupt).

SAN

Storage area network. A high-speed, scalable network of servers and storage devices that provides storage resource consolidation, enhanced data access, and centralized storage management.

server-capable administration node

A node that is installed with the `cluster_admin` product and is also capable of coordinating CXFS metadata.

server-side licensing

Licensing that uses license keys on the CXFS server-capable administration nodes; it does not require node-locked license keys on CXFS client-only nodes. The license keys are node-locked to each server-capable administration node and specify the number and size of client-only nodes that may join the cluster membership. All nodes require server-side licensing.

shutdown

The fail policy that tells the other nodes in the cluster to wait before reforming the CXFS kernel membership. The surviving cluster delays the beginning of recovery to allow the node time to complete the shutdown. See also *forced CXFS shutdown*.

split cluster

A situation in which cluster membership divides into two clusters due to an event such as a network partition, or unresponsive server-capable administration node and the lack of reset and/or CXFS tiebreaker capability. This results in multiple clusters, each claiming ownership of the same filesystems, which can result in filesystem data corruption. Also known as *split-brain syndrome*.

snooping

A security breach involving illicit viewing.

split-brain syndrome

See *split cluster*.

spoofing

A security breach in which one machine on the network masquerades as another.

standby node

A server-capable administration node that is configured as a potential metadata server for a given filesystem, but does not currently run any applications that will use that filesystem.

static heartbeat monitoring

Monitors CXFS kernel heartbeat constantly at 1-second intervals and declares a timeout after 5 consecutive missed seconds (default). See also *dynamic heartbeat monitoring*.

storage area network

See *SAN*.

system controller port

A port sitting on a node that provides a way to power-cycle the node remotely. Enabling or disabling a system controller port in the cluster database tells CXFS whether it can perform operations on the system controller port.

system log file

Log files in which system messages are stored.

tiebreaker node

See *CXFS tiebreaker node*.

transaction rates

I/O per second.

user-space membership

See *cluster database membership*.

view area

The portion of the GUI window that displays components graphically. See also *details area*.

VLAN

Virtual local area network.

whitelisted

A node that is explicitly allowed to be automatically configured into the cluster database.

XFS

A filesystem implementation type for the Linux operating system. It defines the format that is used to store data on disks managed by the filesystem.

Index