sgi

CXFS™ 7 Administrator Guide
for SGI® InfiniteStorage™

New Features in This Guide

Note: Be sure to read the release notes for the SGI® InfiniteStorage™ Software Platform, your specific CXFS™ software platforms, and the late-breaking caveats pages on Supportfolio™ to learn about any changes to the installation and configuration procedures.

This guide includes the following:

- "IPv6: Define `failover_net` to Order Addresses for Automatic CXFS Configuration" on page 75

- "Use a Warm Start for `rpcbind`" on page 94

- "Installing InfiniBand RAID" on page 100

- "XVM Volume Problems on Only One Node" on page 428

- Corrections and clarifications

Record of Revision

Version	Description
001	January 2010 Supports the CXFS 6.0 product in SGI InfiniteStorage Software Platform (ISSP) 2.0 release.
002	September 2010 Supports the CXFS 6.2 product in the SGI InfiniteStorage Software Platform (ISSP) 2.2 release.
003	April 2011 Supports the CXFS 6.4 product in the ISSP 2.4 release.
004	April 2012 Supports the CXFS 6.6 product in the ISSP 2.6 release.
005	April 2013 Supports the CXFS 7.0 product in the ISSP 3.0 release.
006	November 2013 Supports the CXFS 7.1 product in the ISSP 3.1 release.
007	May 2014 Supports the CXFS 7.2 product in the ISSP 3.2 release.
008	October 2014 Supports the CXFS 7.3 product in the ISSP 3.3 release.
009	July 2015 Supports the CXFS 7.4 product in the ISSP 3.4 release.

Contents

Contents

Figures

Tables

About This Guide

This guide describes how to administer the CXFS™, parallel-access filesystem for high-performance computing environments. It assumes that you are already familiar with the XFS filesystem and you have access to the *XVM Volume Manager Administrator Guide.*

You should read through this entire book, especially Chapter 15, "Troubleshooting" on page 393, before attempting to install and configure a CXFS cluster.

Related Publications

For information about this release, see the following release notes:

- SGI® InfiniteStorage™ Software Platform (ISSP): `README.txt`

- CXFS:

   - `README_CXFS_GENERAL.txt`

   - `README_CXFS_LINUX.txt`

   - `README_CXFS_MACOSX.html`

   - `README_CXFS_WINDOWS.html`

The following documents contain additional information:

- *DMF 6 Administrator Guide*

- *CXFS 7 Client-Only Guide for SGI InfiniteStorage*

- *Linux Configuration and Operations Guide*

- *XVM Volume Manager Administrator Guide*

- The user guide and quick start guide for your hardware

- *NIS Administrator's Guide*

- *Personal System Administration Guide*

- *Performance Co-Pilot for Linux User's and Administrator's Guide*

- *SGI L1 and L2 Controller Software User's Guide*

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

Man Page	Software Product
cbeutil(**8**)	cluster_admin
cdbBackup(**8**)	cluster_admin
cdbRestore(**8**)	cluster_admin
cdbconfig(**8**)	cluster_admin
cdbutil(**8**)	cluster_admin
cmond(**8**)	cluster_admin
cms_failconf(**8**)	cluster_control
cms_intervene(**8**)	cluster_control
crsd(**8**)	cluster_control
cxfslicense(**8**)	cluster_admin
cxfs_admin(**8**)	cxfs_admin
cxfs-config(**8**)	cxfs_util
cxfsdump(**8**)	cxfs_util
cxfsmgr(**8**)	sgi-sysadm_cxfs-client
cxfscp(**1**)	cxfs_util
fs2d(**8**)	cluster_admin
xvm(**5**)	sgi-xvm-commands
xvm(**8**)	sgi-xvm-commands
xvmgr(**1**)	sgi-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, man pages, and other information.

- The /docs directory on the ISSP DVD or in the Supportfolio™ download directory contains the following:

  - The ISSP release note: /docs/README.txt

  - Other release notes: /docs/README_*NAME*.txt

  - A complete list of the packages and their location on the media: /docs/RPMS.txt

  - The packages and their respective licenses: /docs/PACKAGE_LICENSES.txt

- The release notes and manuals are provided in the noarch/sgi-isspdocs RPM and will be installed on the system into the following location:

  /usr/share/doc/packages/sgi-issp-*ISSPVERSION*/*TITLE*

- You can view man pages by typing man *title* at a command line.

Note: The external websites referred to in this guide were correct at the time of publication, but are subject to change.

Conventions

This guide uses the following terminology abbreviations:

- *Linux* refers to the supported distribution of Linux defined in the CXFS release notes

- *Windows* refers to Microsoft Windows 2000, Microsoft Windows 2003, Microsoft Windows XP, and Microsoft Windows Vista

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
server-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
#	In an example, this prompt indicates that the command is executed on an any node
specificnode#	In an example, this prompt indicates that the command is executed on a node named *specificnode* or of node type *specificnode*

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 either 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:

    http://www.sgi.com/support/supportcenters.html

SGI values your comments and will respond to them promptly.

Introduction to CXFS™

This chapter discusses the following:

What is CXFS?

CXFS is a parallel-access 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 with one of the following switch types:

- Fibre Channel
- Serial-attached storage (SAS)
- InfiniBand®

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.

CXFS provides centralized administration with an intuitive graphical user interface (GUI) and command-line interface. See "CXFS Tools" on page 39.

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

- Server-capable administration nodes built on x86_64 architecture and running one of the following operating systems, as documented in the ISSP release note:

  – SUSE® Linux® Enterprise Server (SLES®)

  – Red Hat® Enterprise Linux (RHEL)

  Also see "Hardware and Software Requirements for Server-Capable Administration Nodes" on page 34.

- Client-only nodes running any mixture of the following operating systems:

  – Apple® Mac OS X®

  – Red Hat Enterprise Linux (RHEL)

  – SUSE Linux Enterprise Server (SLES)

  – Microsoft® Windows®

See the CXFS release notes for the supported kernels, update levels, and service pack levels. For additional details about client-only nodes, see the *CXFS 7 Client-Only Guide for SGI InfiniteStorage*.

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

- Replicated cluster database
- XVM path failover (failover version 2)
- Server failover
- Network failover

You can use the supported high-availability (HA) product in conjunction with CXFS to failover associated applications. See "Highly Available Services" on page 30.

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 6

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 39.

    – 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 39).

    – 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)

    – Fast metadata operations

- No restriction on the maximum I/O size (any I/O size supported by the operating
  system)

- Dynamically allocated metadata space.

- Quotas. See "Quotas Differences" on page 33.

- Filesystem reorganizer (defragmenter), which must be run from the CXFS metadata server for a given filesystem. See the `fsr_xfs`(8) 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.

- External log filesystems.

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 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.

For high-performance NFS edge-serving (in which CXFS client nodes can export data with NFS), contact SGI Professional Services.

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.

This section discusses the following:

- "CXFS Features" on page 8

- "CXFS Restrictions" on page 9

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 gigabit 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 SGI DMF™ tiered-storage virtualization software 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

- Swap to a file residing on a CXFS filesystem.

- XVM failover version 1

- Real-time filesystems

Note: CXFS does support the following newer products:

- "Guaranteed-Rate I/O (GRIO) Version 2 Overview" on page 31
- "XVM Path Failover Overview" on page 145

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 and Cluster ID" on page 11

- "Pool" on page 11

Also see the Glossary on page 577.

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.

RAID

A redundant array of independent disks.

LUN

A logical unit (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 Guide.*

Cluster and Cluster ID

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 through the SAN 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.

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.

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.)

When you define a node using cxfs_admin, it is automatically added to both the pool (the poolnode class) and the specific cluster definition. If you define a node with the CXFS GUI, it is merely added to the pool and you must explicitly add it to the cluster. Nodes are in the pool regardless of whether they are enabled or disabled.

When using the GUI, you can 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).

If necessary, you can use the detach command in cxfs_admin(8) to move a node from the cluster to the pool; see "Move a Node Between the Cluster and the Pool with cxfs_admin" on page 269.

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.
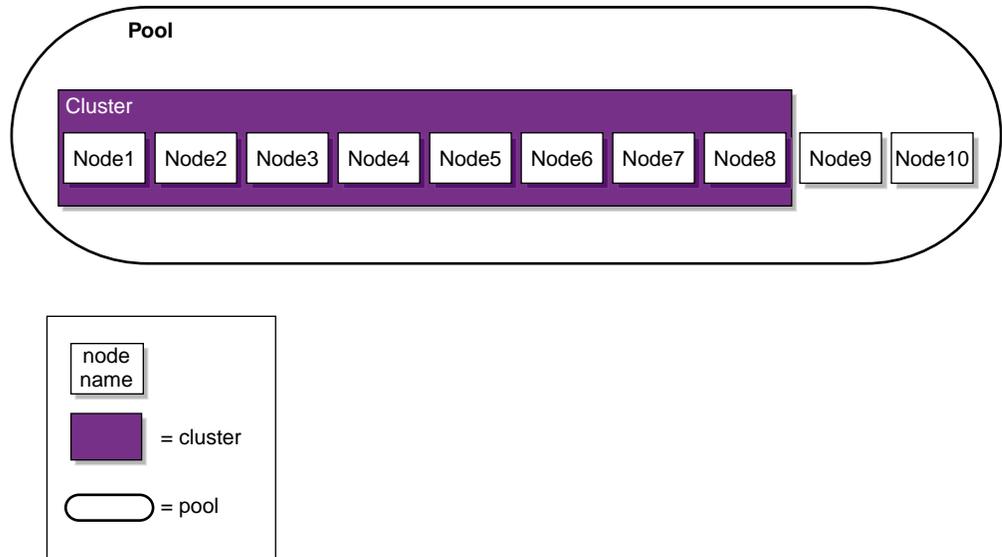
Figure 1-1 Pool and Cluster Concepts

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, 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 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 43

– "CXFS Control Daemons" on page 44

– Appendix A, "CXFS Software Architecture" on page 475

- *Client-only node*, which is installed with the cxfs_client software product, has a minimal implementation of CXFS that runs the CXFS client control daemon (cxfs_client). This node can run any supported operating system.

  A client-only node can safely mount CXFS filesystems but it cannot become a CXFS metadata server or perform cluster administration. A client-only node retrieves the information necessary for its tasks by communicating with a server-capable administration node.

  A client-only node does 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 node. (If the set of server-capable administration nodes changes, another node may become responsible for updating the client-only nodes.)

  For more details, see:

  – "CXFS Control Daemons" on page 44

  – Appendix A, "CXFS Software Architecture" on page 475

  – *CXFS 7 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 SAN 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 for the filesystem exists when the active metadata server fails, recovery will take place. For more information, see "Recovery" on page 26.

- 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 should be exported from the active metadata server for best performance (which might require manually relocating the metadata server). For more information on NFS exporting of CXFS filesystems, see "CXFS Mount Scripts" on page 318. For a high-performance NFS edge-serving configuration, contact SGI Professional Services.

- There should be an odd number of server-capable administration nodes with CXFS services running for quorum calculation purposes (see "Quorum" on page 23). If you have a cluster that consists of only two server-capable administration nodes (and no client-only nodes), you should use system reset and you should not use a tiebreaker. If the cluster has more than one server-capable administration node and at least one client-only node, SGI recommends that you define a stable client-only node as the CXFS tiebreaker. See "CXFS Tiebreaker" on page 25 and "Use an Odd Number of Server-Capable Administration Nodes" on page 54.

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 43 and Table 1-3 on page 44.)
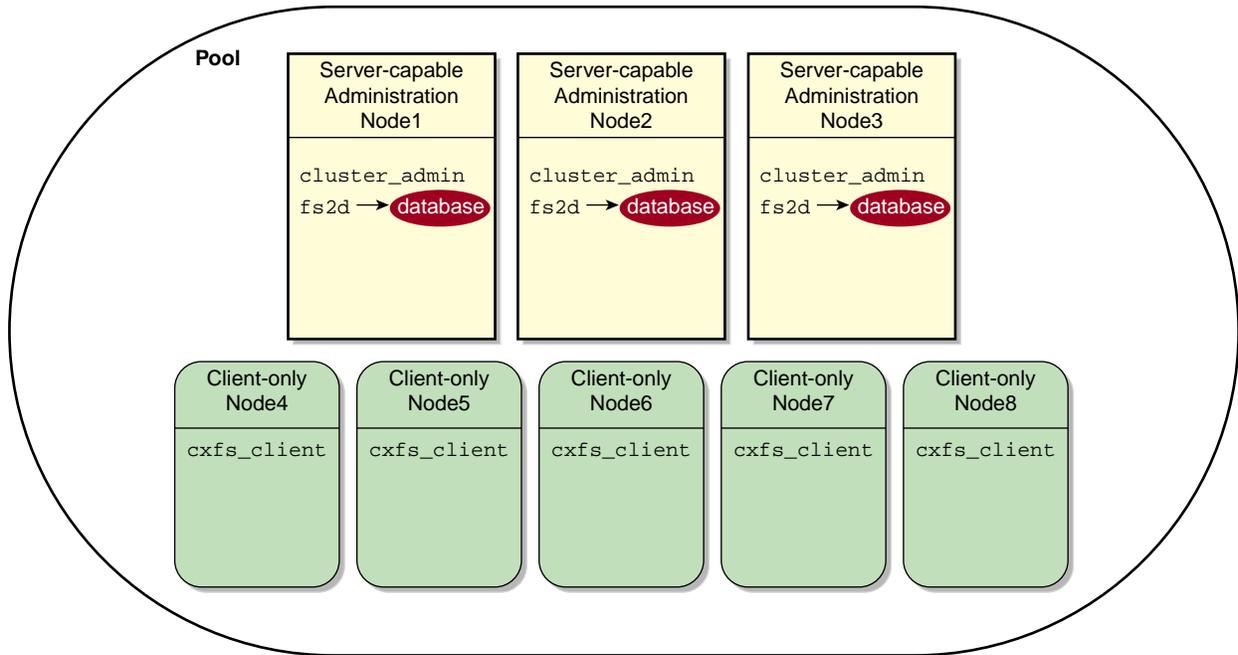
Figure 1-2 Installation Differences

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 301.

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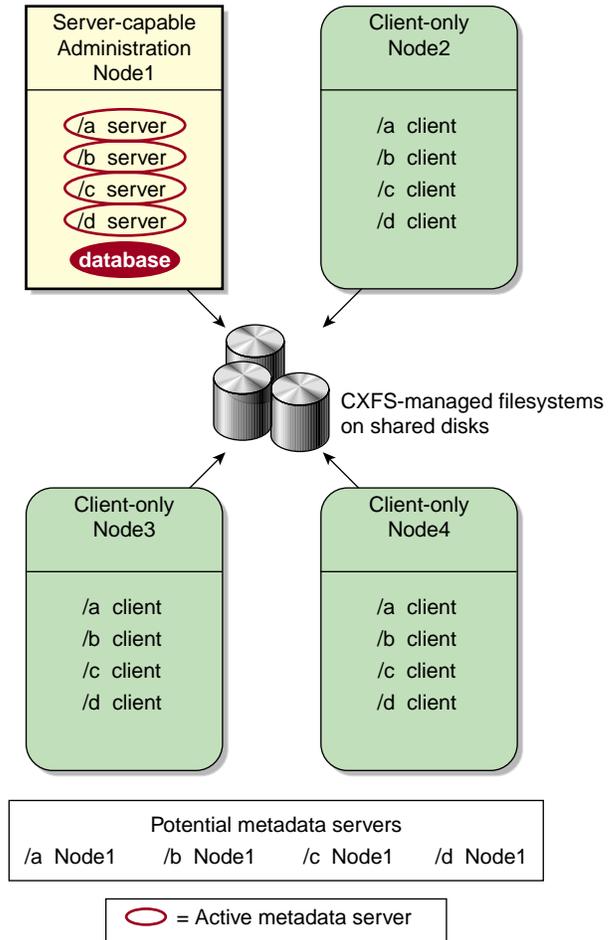
Figure 1-3 One Metadata Server (No Relocation or Recovery)

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.
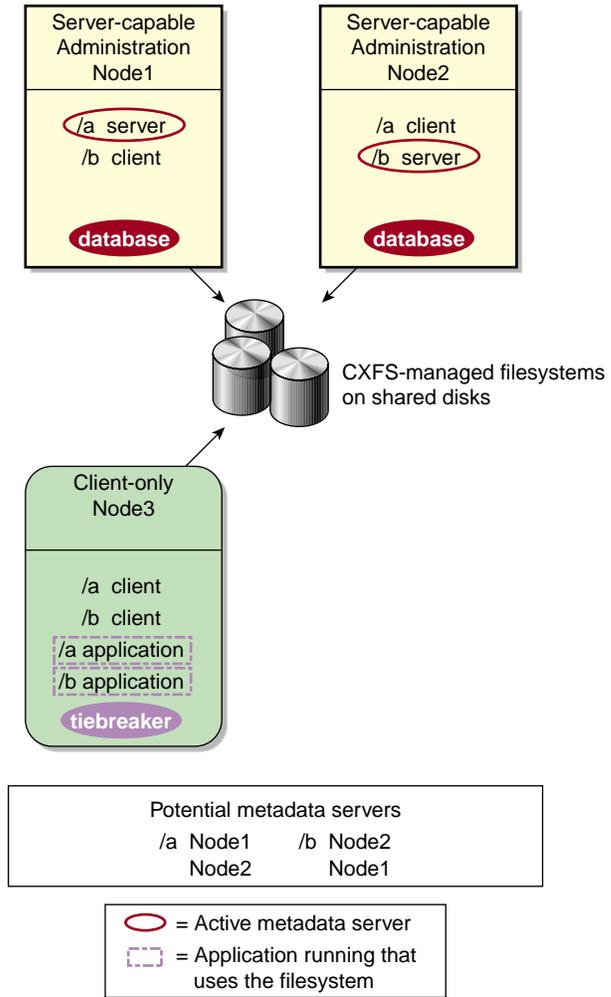
Figure 1-4 Two Metadata Servers for Two Filesystems

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.
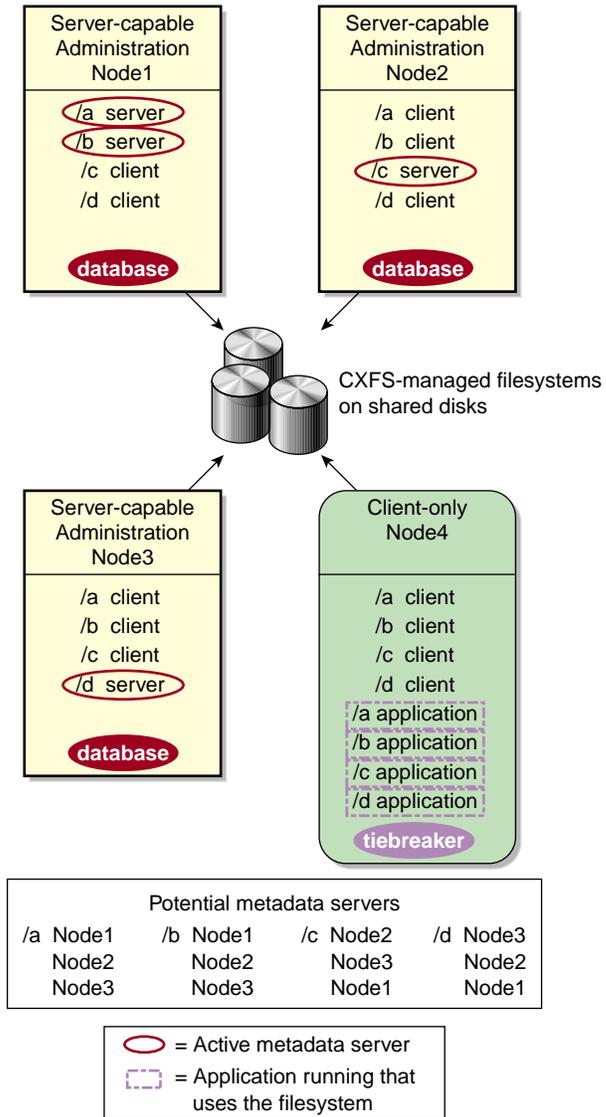
Figure 1-5 Three Metadata Servers for Four Filesystems

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*:

- *CXFS kernel membership* is the group of CXFS nodes in the cluster that can actively share filesystems, as determined by 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.

- *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 cluster database membership work together to coordinate configuration changes to the cluster database.

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

Quorum

Quorum is 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 **defined in the cluster** plus (if defined) the tiebreaker node (usually a client-only node) are required to **form** an initial membership

    - Half (**50%**) of the server-capable administration nodes **defined in the cluster** are required to **maintain** an existing membership

    Nodes that are defined in the cluster and enabled are used in kernel quorum calculations (disabled nodes or nodes that are only in the pool are not used in kernel quorum calculations).

- For **cluster database** membership, **50%** of the server-capable administration nodes **in the pool** are required to **form and maintain** a cluster.

    Note: When using the CXFS GUI, a newly defined node is added to the pool and must be explicitly added to the cluster definition; when using the cxfs_admin tool, a newly defined node is added automatically to both the pool and the cluster definition.

    All pool nodes are used for the cluster database quorum calculations, regardless of whether they are enabled or disabled.

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.)

Note: When NFS or Samba serving from a CXFS cluster, the network used for remote fileserving cannot be a backup private network for CXFS. Using the fileserving network as a backup private network for the CXFS private network may result in heartbeat timeouts, which will cause a severe drop in CXFS and fileserving performance.

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 49

- "Use a Private Network and Unique Cluster Name/ID" on page 50

- "Network Failover Modification Tasks with `cxfs_admin`" on page 286

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 shared storage I/O ports (the ports on a Fibre Channel, SAS, or InfiniBand switch). After fencing, 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* 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 shared-storage I/O ports.

- Node `shutdown` fail policy for client-only nodes that use static CXFS kernel heartbeat monitoring.

  Note: You cannot use the node `shutdown` fail policy for nodes that use dynamic kernel heartbeat monitoring.

- Cluster tiebreaker on a stable client-only node.

For more information, see "Protect Data Integrity on All Nodes" on page 57,

CXFS Tiebreaker

The *CXFS tiebreaker node* is used in the process of computing the CXFS kernel membership for the cluster when exactly half of 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 25.

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, see:

* "CXFS Relocation Capability" on page 309
* "Use Relocation Properly" on page 80

Recovery

Metadata-server 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.

Both of the following are required for metadata-server recovery:

* A CXFS kernel membership quorum

- A potential metadata server with the filesystem currently mounted

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 in a process known as *client recovery*. 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 node's 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 triggers that will initiate metadata-server recovery:

- A metadata server panics

- A metadata server locks up, causing CXFS kernel heartbeat timeouts on metadata clients

- A metadata server loses connection to all of the CXFS networks

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	Server-capable Administration Node B
Metadata server	
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	Server-capable Administration Node B
Metadata server	
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 for Metadata Servers

See also "Use Relocation Properly" on page 80.

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 *shut down CXFS services* withdraws a node from the CXFS kernel membership, due to the fact that the node has failed somehow. The node remains enabled in the cluster database.

Starting, stopping, or shutting down CXFS services does not affect the daemons involved. See also:

- "CXFS Control Daemons" on page 44
- "Start CXFS Services with the GUI" on page 206
- "Stop CXFS Services with the GUI" on page 206
- "Enable a Node with `cxfs_admin`" on page 263
- "Disable a Node with `cxfs_admin`" on page 263
- "Enable or Disable CXFS Kernel Membership (`cmsd`) for the Local Node" on page 267
- "Forced CXFS Shutdown: Revoke Membership of Local Node" on page 328

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 and DMF) into a single distribution that is tested for interoperability.

This section discusses the following:

- "DMF" on page 30
- "Highly Available Services" on page 30
- "GPT Labels Overview" on page 31
- "Guaranteed-Rate I/O (GRIO) Version 2 Overview" on page 31

- "Storage Management" on page 32

- "Volume Management with XVM" on page 34

DMF

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

If DMF is managing a CXFS filesystem, DMF will ensure that the filesystem's CXFS metadata server is the DMF server and will use metadata server relocation if necessary to achieve that configuration. CXFS client-only nodes may be DMF parallel data-mover nodes or DMF clients. With the Parallel Data Mover Option, DMF must always run in a CXFS environment.

Note: Each parallel data-mover node counts towards the total CXFS cluster node count. If you have a cluster with 2 CXFS server-capable administration nodes and 2 CXFS client-only nodes installed as parallel data-mover nodes, then you could have a total maximum number of 92 other CXFS client-only nodes doing normal client-only work (2+2+92=96).

DMF parallel data-mover nodes must be x86_64 architecture and run the same operating system as the DMF server and CXFS server-capable administration nodes.

For more information, see:

- "Using DMF" on page 319

- *DMF 6 Administrator Guide*

Highly Available Services

The CXFS resource agent reports on whether the metadata server for a filesystem on a given node is running, not running, or has an error. The HA software then moves services accordingly, based upon constraints that the customer sets. For more information about the CXFS resource agent, see the *High Availability Guide for SGI InfiniteStorage*.

GPT Labels Overview

CXFS supports XVM labels on LUNs with globally unique identifier (GUID) partition table (GPT) labels. You can create these labels on server-capable administration nodes and Linux client-only nodes.

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.

For information about creating GPT labels, see the *XVM Volume Manager Administrator Guide*.

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

CXFS supports guaranteed-rate I/O version 2 (GRIOv2) clients on all platforms, with a GRIOv2 server on a server-capable administration node. GRIOv2 is disabled by default on the server-capable administration nodes and on Linux client-only nodes.

On Linux, GRIOv2 supports node-level static reservations.

Note: GRIOv2 application reservations are functional only for Linux and Windows nodes; they are not functional on Mac OS X nodes.

As the superuser, you can run the following commands from any node in the cluster:

- `grioadmin` provides stream and bandwidth management
- `griomon` provides information about GRIOv2 status
- `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 GRIOv2 commands differ by platform. See Appendix B, "Path Summary" on page 485, and the appendix in the *CXFS 7 Client-Only Guide for SGI InfiniteStorage*.

For details about GRIOv2, see the *Guaranteed-Rate I/O Version 2 for Linux Guide*.

Storage Management

This section describes the CXFS differences for the following:

- "Storage Backup Differences" on page 32
- "NFS Differences" on page 32
- "Quotas Differences" on page 33
- "Samba Differences" on page 33

Storage Backup Differences

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; 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 Differences

SGI Enhanced NFS is installed by default when you install the CXFS server or CXFS edge-server software and will be used rather than stock NFS. You can put an Enhanced 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 Enhanced NFS server on the CXFS active metadata server for optimal performance.

Quotas Differences

XFS quotas are supported. However, the quota mount options must be the same on all mounts of the filesystem.

You can administer user and group quotas from any Linux node in the cluster.

With a Linux metadata server, the only supported use of project quotas is as directory quotas: for this purpose, the `/etc/projects` configuration file specifies which directory tree falls under which project, while the `/etc/projid` configuration file maps numeric project IDs to names.

You can view or modify project quotas only on the active CXFS metadata server, not on a potential metadata server or on client-only nodes.

For more information about setting quotas, see *XFS for Linux Administration*.

Samba Differences

A CXFS filesystem may be shared via Samba from a CXFS node to other types of machines that are not running CXFS software, such as a Windows machine. The Samba server should run on the active metadata server for optimal performance. There can be one Samba server per CXFS filesystem. You must 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.

$\triangle$ **Caution:** SGI recommends that you do not use multiple Samba servers.

Volume Management with XVM

CXFS uses the XVM volume manager to combine underlying physical disk storage into a single logical unit, known as a *logical volume* (abbreviated to *volume*). Volumes behave like standard disk partitions and you can use them as arguments anywhere that you can specify a partition. A volume allows a filesystem or raw device to consist of multiple physical disks. Using volumes can also increase disk I/O performance because a volume can be distributed (or striped) across multiple disks. Volumes can also be used to mirror data on different disks

CXFS uses XVM to provide the following:

- Striping

- Mirroring

- Concatenation

- Advanced recovery features

You must perform XVM configuration on a server-capable administration node using either the xvm(8) command or the **XVM Manager** graphical user interface (GUI).

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

Hardware and Software Requirements for Server-Capable Administration Nodes

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

- "Cluster Size and Composition" on page 35

- "Reset or I/O Fencing Capability" on page 36

- "RAID Storage Devices" on page 36
- "Network and Connectivity" on page 36
- "Licenses" on page 36
- "XVM Volume Manager" on page 37

Cluster Size and Composition

A CXFS cluster is supported with as many as 96 nodes, of which as many as 16 can be server-capable administration nodes:

- There must be 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.)

- All server-capable administration nodes within the cluster must have similar capabilities. They must all have x86_64 architecture and have the same flavor of OS (you must use either all RHEL nodes or all SLES nodes). See also "Provide Enough Memory" on page 51. For the required release levels and kernels, see the release notes.

  Note: CXFS does not support server-capable administration nodes that are members of a partitioned system with a single power supply, because failure of that power supply may result in failure of the CXFS cluster.

- Server-capable administration nodes must be dedicated to CXFS and filesystems work. See "Dedicate Server-Capable Administration Nodes to CXFS Work" on page 53.

- All nodes in the CXFS require adequate compute power. This is particularly true for server-capable administration nodes, which must deal with the required communication and I/O overhead. A server-capable administration node should have 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 51.

Reset or I/O Fencing Capability

CXFS requires system reset capability and/or supported Fibre Channel, SAS, or InfiniBand switches. For supported switches, see the release notes and Chapter 4, "Switch Configuration" on page 103. 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 (or optionally the `ssh` port).

RAID Storage Devices

CXFS supports the RAID hardware as specified in the release notes and Chapter 3, "SGI RAID for CXFS Clusters" on page 95.

Network and Connectivity

CXFS has the following network requirements:

- 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.)

- A network switch of at least 100baseT. (A network hub is not supported.)

- A private IPv4 network address or an IPv6 link-local network address using 100baseT or Gigabit Ethernet TCP/IP must be connected to each node.

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

Licenses

CXFS requires licenses. See the general release notes and Chapter 5, "CXFS Licensing" on page 113.

XVM Volume Manager

CXFS requires the XVM volume manager. The functionality of the XVM Manager graphical user interface (GUI) is incorporated into the CXFS Manager GUI. For more information about XVM, see *XVM Volume Manager Administrator Guide*.

Hardware and Software Requirements for Edge-Serving Nodes

In an edge-serving configuration, the edge-serving nodes should use the same operating system as the server-capable administration nodes and must be x86_64 architecture.

CXFS Software Products Installed on Server-Capable Administration Nodes

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

- Licensing and Performance Co-Pilot:

  ```
  lkSGI
  lkSGI-mgr
  pcp
  pcp-gui
  pcp-libs
  pcp-libs-devel
  pcp-pmda-infiniband (optional)
  ```

- Application binaries, documentation, and support tools:

  ```
  cluster_admin
  cluster_control
  cxfs_admin
  cxfs_cluster
  cxfs_util
  sgi-pm-commands
  sgi-xvm-commands
  ```

- Enhanced NFS:

  ```
  kmod-enhancednfs         (RHEL)
  sgi-enhancednfs-kmp-default    (SLES)
  ```

```
pmdanfs
sgi-enhancednfs-debuginfo
sgi-nfsinit
sgi-nfs-utils
```

- Enhanced XFS:

  sgi-acl   *(SLES only)*
  sgi-attr
  sgi-dmapi
  sgi-dmapi-devel
  sgi-enhancedxfs-debuginfo
  kmod-enhancedxfs   *(RHEL)*
  sgi-enhancedxfs-kmp-default   *(SLES)*
  sgi-libacl
  sgi-libacl-devel
  sgi-libattr
  sgi-libattr-devel
  sgi-xfsdump
  sgi-xfsprogs
  sgi-xfsprogs-devel

- GRIOv2 software: *(optional)*:

  grio2-cmds
  grio2-server

- GUI tools:

  sgi-sysadm_base-client
  sgi-sysadm_base-lib
  sgi-sysadm_base-server
  sgi-sysadm_cluster_base-client
  sgi-sysadm_cluster_base-server
  sgi-sysadm_cxfs-client
  sgi-sysadm_cxfs-server
  sgi-sysadm_cxfs-web
  sgi-sysadm_xvm-client
  sgi-sysadm_xvm-server
  sgi-sysadm_xvm-web

- Kernel module:

  ```
  kmod-cxfs          (RHEL)
  sgi-cxfs-kmp-default    (SLES)
  ```

- Other ISSP common software:

  ```
  libbapi
  libbapi-devel
  sgi-build-key
  sgi-issp-release
  sgi-isspdocs
  sgi-support-tools
  sgtools
  ```

For information about client-only nodes, see *CXFS 7 Client-Only Guide for SGI InfiniteStorage.*

CXFS Tools

This section discusses the following:

- "Configuration Commands" on page 39
- "Cluster Administration Daemons" on page 43
- "CXFS Control Daemons" on page 44
- "Other Administrative Commands" on page 44

See also "CXFS and Cluster Administration `service` Commands" on page 309.

Configuration Commands

You can perform CXFS configuration tasks using the GUI (`cxfsmgr`) 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	sgi-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. The CXFS GUI also provides access to the XVM GUI.
xvmgr	sgi-sysadm_xvm-client	Directly 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 normally access the XVM GUI as part of the CXFS GUI.

The rest of this section discusses:

- "CXFS GUI Overview" on page 40

- "cxfs_admin Command-Line Configuration Tool Overview" on page 42

CXFS GUI Overview

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 web browser enabled to use Java™, including Windows® and Linux computers and laptops.

Figure 1-7 shows an example GUI screen. For more information, see Chapter 10, "CXFS GUI" on page 167.

Note: The GUI must be connected to a server-capable administration node, but it can be launched elsewhere; see "Starting the GUI via the Command Line" on page 168.
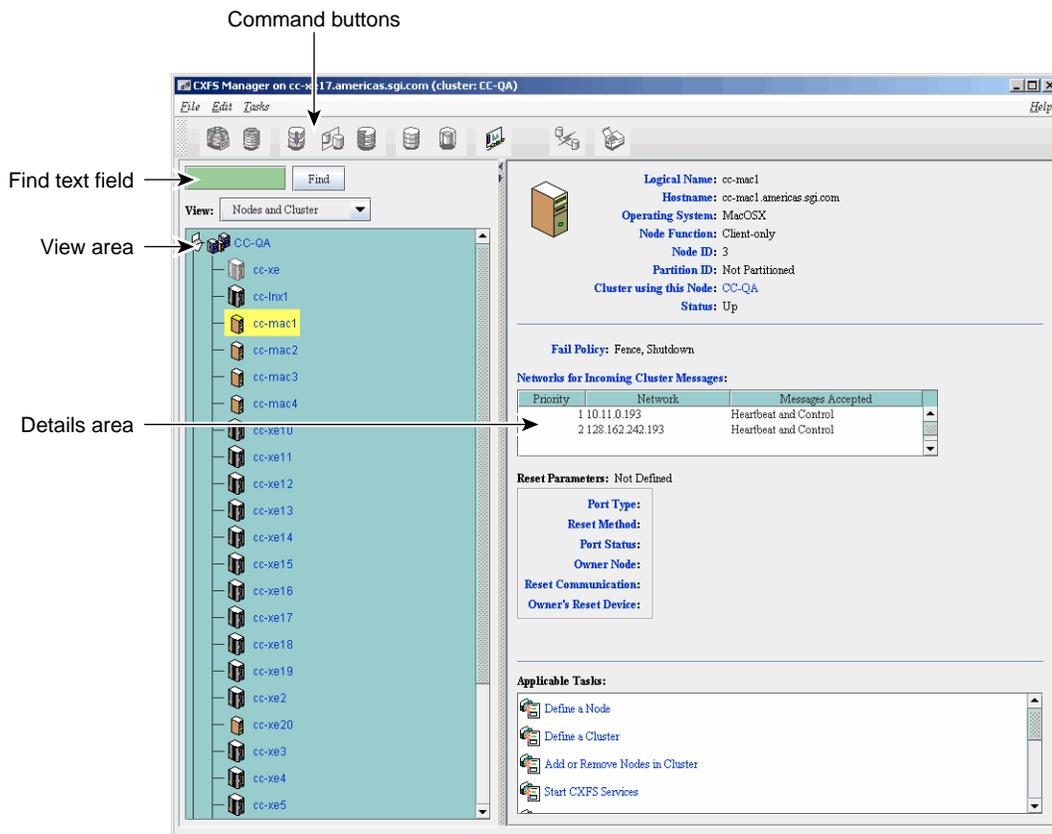
Figure 1-7 CXFS Manager GUI

`cxfs_admin` Command-Line Configuration Tool Overview

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 a list of possible choices (by using the <TAB> key).

- Validates all input before a command is completed.

- Provides a step-by-step mode with 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, "cxfs_admin Command" on page 233.

Cluster Administration Daemons

Table 1-2 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 315
- "Manually Starting/Stopping CXFS" on page 337
- Appendix A, "CXFS Software Architecture" on page 475

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, SAS, or InfiniBand 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 315

- "Manually Starting/Stopping CXFS" on page 337

- Appendix A, "CXFS Software Architecture" on page 475

Other 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 7 Client-Only Guide for SGI InfiniteStorage.*

Table 1-4 Other Administrative Commands

Command	Software Product	Description
cbeutil	cluster_admin	Examines the dynamic state of the cluster database, which includes information that is not stored on disk (for debugging purposes)
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	Examines the on-disk cluster database by means of commands that correspond to functions in the libcdb library (for debugging purposes)
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_control	Obtains I/O fencing configuration and status information
hafence	cxfs_cluster	Administers the CXFS I/O fencing configuration stored in the cluster database (normally, you will perform this task using the GUI or cxfs_admin)
sysadmd	sgi-sysadm_base-server	Allows clients to perform remote system administration for the GUI server

CXFS Best Practices

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

- "Configuration Best Practices" on page 47
- "Administration Best Practices" on page 76

For the latest information and a matrix of supported CXFS and operating system software, see the following page on Supportfolio™:

 http://support.sgi.com/content_request/838562/index.html

Configuration Best Practices

This section discusses the following configuration topics:

- "Fix Network Issues First" on page 49
- "Save the Current Configuration Before Making Changes" on page 49
- "Use a Private Network and Unique Cluster Name/ID" on page 50
- "Take Care When Using YAST for Configuration" on page 51
- "Use the Same OS Distribution for All Server-Capable Administration Nodes" on page 51
- "Provide Enough Memory" on page 51
- "Configure for Best Performance" on page 52
- "Use CXFS Configuration Tools Appropriately" on page 52
- "Restart the CXFS Cluster using `restart`" on page 52
- "Ensure Cluster Database Membership Quorum Stability" on page 53
- "Be Consistent in Configuration" on page 53
- "Dedicate Server-Capable Administration Nodes to CXFS Work" on page 53
- "Use an Odd Number of Server-Capable Administration Nodes" on page 54

- "Create an Initial Cluster of All Server-Capable Administration Nodes" on page 54

- "Unmount Filesystems Before Adding or Deleting Server-Capable Administration Nodes" on page 55

- "Make Most Nodes Client-Only" on page 55

- "Use a Client-Only Tiebreaker" on page 55

- "Protect Data Integrity on All Nodes" on page 57

- "Avoid CXFS Kernel Heartbeat Issues on Large Systems" on page 67

- "Minimize the Number of Switches" on page 69

- "Configure Filesystems Properly" on page 69

- "Enable Forced Unmount When Appropriate" on page 70

- "Use the Appropriate CXFS Kernel Heartbeat Monitoring" on page 71

- "Verify the Configuration" on page 71

- "Use the Recovery Timeout Mechanism" on page 72

- "Use Proper Storage Management Procedures" on page 72

- "Run Samba Appropriately" on page 73

- "Install `ipmitool` on Server-Capable Administration Nodes" on page 73

- "Use the LSI Drivers that Ship with the Linux OS" on page 73

- "Run the `xvm` Command in Cluster Mode" on page 73

- "Specify XVM Path Failover and HBA Usage in `failover2.conf`" on page 74

- "Prevent Stalled-Recovery Timeout in a Non-HA DMF Environment" on page 74

- "Configure Appropriately for IPv6" on page 74

- "Use RAID Mirroring Not XVM Mirroring" on page 75

- "Avoid Dropping Out-Of-Order Frames" on page 75

- "Suppress Change Notification for Switch Ports Connected to Nodes" on page 76

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 125.

When you install the CXFS software on the client-only node, 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.

Save the Current Configuration Before Making Changes

After establishing the configuration and before making any modifications, you should save the current configuration information so that you can return to it later in case of failure:

- "Save the CXFS Configuration"

- "Save the XVM Configuration"

Save the CXFS Configuration

Before making significant changes to an existing CXFS configuration, run the `cxfs_admin config` command to create a copy of the current database. If needed, you can then use the output to recreate the cluster database after performing a `cdbreinit`. See "Saving and Recreating the Current Configuration with `cxfs_admin`" on page 295.

Save the XVM Configuration

Occasionally, XVM labels can become corrupted. You should perform a backup of the XVM configuration using `xvm dump` any time it is changed so that you will be able to recover from potential problems. You should save the `xvm dump` output into an XFS (not CXFS) filesystem. For example, the following will dump all `xvm_labels` to the file `/var/xvm_config`:

```
xvm dump -topology -f /var/xvm_config phys/'*' vol/'*'
```

For more information, see the section about saving and regenerating XVM configurations in the *XVM Volume Manager Administrator Guide.* If repair is required, contact SGI Support.

Use a Private Network and Unique Cluster Name/ID

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 connected to the same public network, use unique cluster names and cluster IDs.

  Note: To be prompted to specify a cluster ID when using the `cxfs_admin`(8) command, you must use advanced mode. See "Basic and Advanced Mode" on page 245, and "Create a Cluster with `cxfs_admin`" on page 271.

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

- When NFS or Samba serving from a CXFS cluster, the network used for remote fileserving cannot be a backup private network for CXFS. Using the fileserving network as a backup private network for CXFS private network may result in

heartbeat timeouts, which will cause a severe drop in CXFS and fileserving performance.

Take Care When Using YAST for Configuration

You must be careful when configuring private network interfaces with YAST because it may create duplicate entries using the same hostname.

Use the Same OS Distribution for All Server-Capable Administration Nodes

All server-capable administration nodes within the cluster must have similar capabilities. You must use all RHEL systems or all SLES systems. See also "Provide Enough Memory" on page 51.

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 (where *#inodes* is the maximum number of open files at any one time):

$$2 \text{ KB} \quad \text{x} \quad \textit{#inodes} \quad = \quad \textit{server-capable\_administration\_node\_memory}$$

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 at 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.

Configure for Best Performance

For best performance, SGI recommends the following:

- Use preallocation with direct I/O and stripe-width sized writes
- Never mix direct I/O and buffered I/O
- Use multithread I/O to reduce the effect of I/O latency
- Write files sequentially to avoid seek penalties and fragmentation

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 via the Command Line" on page 168.

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

Restart the CXFS Cluster using `restart`

To restart the CXFS cluster services without interruption, enter the following the following:

```
server# service cxfs_cluster restart
```

Note: If the node is the member of an active CXFS cluster, do not use the following sequence because it will cause interruption:

```
server# service cxfs_cluster stop
server# service cxfs_cluster start
```

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:

```
server-admin# tail -f /var/log/cxfs/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:

```
server-admin# service cxfs stop
server-admin# service cxfs_cluster stop

server-admin# service cxfs_cluster start
server-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 Chapter 8, "Postinstallation Steps" on page 137.

Dedicate Server-Capable Administration Nodes 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 you want to use an application 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. See "Use a Client-Only Tiebreaker" on page 55.

Create an Initial Cluster of All Server-Capable Administration Nodes

Ensure that you follow the instructions in "Preliminary Cluster Configuration Steps" on page 150.

SGI recommends that you form an initial cluster that consists of all of the server-capable administration nodes, to ensure that these nodes will have lower cell ID numbers than any client-only nodes. By ensuring that the server-capable administration nodes all have the low cell ID numbers, you will avoid potential problems. (The *cell ID* is the number associated with a node that is allocated when a node is added into the cluster definition with the GUI or when it is defined with cxfs_admin. The first node in the cluster has cell ID 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 differs from *node ID*.)

> **Note:** Because `cxfs_admin` automatically adds a node to the cluster when you define the node, you should not define any client-only nodes until after you have defined all server-capable administration nodes. (Unlike `cxfs_admin`, defining and adding are two separate tasks within the GUI.)

After you have added all of the server-capable administration nodes, you can then add the client-only nodes. For large clusters, SGI recommends that you build up the remainder of the cluster in small groups of client-only nodes. This makes it easier to locate and fix problems, should any occur. See "Configuring a Large Cluster" on page 162.

Unmount Filesystems Before Adding or Deleting Server-Capable Administration Nodes

Before adding or removing a server-capable administration node, you must first unmount any filesystems for which the node is a potential metadata server.

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 to examine.

See "Transforming a Server-Capable Administration Node into a Client-Only Node" on page 317.

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 server-capable administration nodes 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 25.

SGI recommends that you start CXFS services on the tiebreaker client after the server-capable administration nodes are all up and running, but before CXFS services are started on any other clients. See "Bring Up the Cluster In an Orderly Fashion" on page 86.

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.

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 25.

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 is over the network and 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

  ---

  Note: See "Install `ipmitool` on Server-Capable Administration Nodes" on page 73.

- `NMI` (nonmaskable interrupt) performs a core-dump of the operating system kernel, which may be useful when debugging a faulty machine

> **Note:** NMI should only be used when directed to by SGI Service personnel and only on SGI ia64 client-only nodes. When used on SGI ia64 client-only nodes, the node will not restart automatically, but will stop in the `kdb` debugger, which requires human intervention to perform debugging and reset the node manually. (See SGI Bulletin TIB 200908 for information on debugging with `kdb`.) This mode is not applicable to SGI x86-64 nodes, which use BMC system controllers.

On a client-only node that has a system controller, you would want to use `reset` for I/O protection when CXFS is a primary activity and you want to get it back online fast; for example, for 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. You must limit the `admin` account to a single login session. For details, see "Limiting Login Sessions" on page 105.

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:

```
server-admin# /usr/cluster/bin/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 switch:

- Fibre Channel

- Serial-attached storage (SAS)

- InfiniBand

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` or `ssh`protocol to the appropriate switch and disables the port.

> ⚠ **Caution:** You must keep the `telnet` or `ssh` port free in order for I/O fencing to succeed; **do not** perform a login 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` or `ssh` session. See "Keep the `telnet` or `ssh` Port Open On the Switch" on page **88** 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.
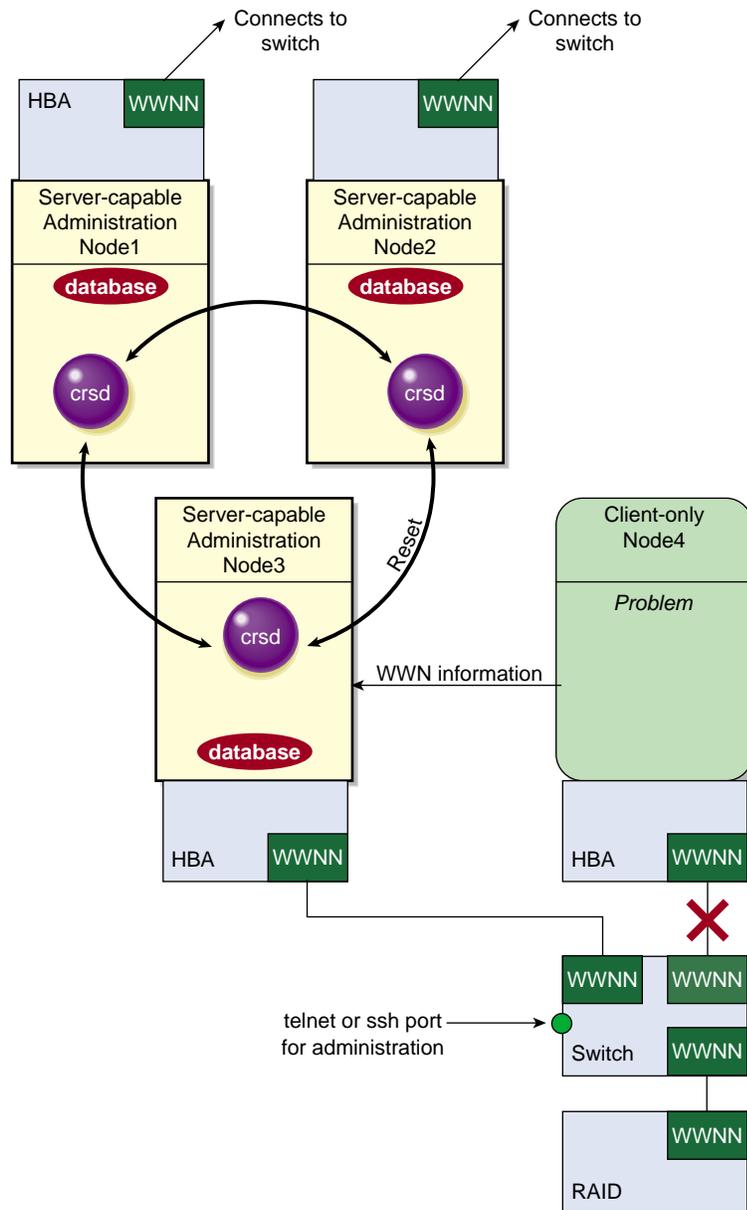
Figure 2-1 I/O Fencing

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 211

*   "Create or Modify a Switch with cxfs_admin" on page 288

For nodes running other supported operating systems, see *CXFS 7 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 65

*   "Define a Node with the GUI" on page 188

*   "Create or Modify a Switch with cxfs_admin" on page 288

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 215, and "Using hafence" on page 310.

⚠️ **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 SAN 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 211

- "Create or Modify a Node with cxfs_admin" on page 253

- "Switch Tasks with cxfs_admin" on page 287

Node Shutdown

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

Note: You cannot use the node shutdown fail policy for nodes that use dynamic kernel heartbeat monitoring.

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.

Note: 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 installation 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, SAS, or InfiniBand switch 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 58.

- `Reset`, which performs a system reset via a system controller. See "System Reset" on page 57.

- `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 58 and "System Reset" on page 57.

- `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 "Node Shutdown" on page 62.

⚠ **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 "Node Shutdown" on page 62.
- 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
```

- 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
```

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

```
Fence (Most common)
Fence, Reset
FenceReset
FenceReset, Reset
Reset
Reset, Fence
```

For more information, see "Use the Appropriate CXFS Kernel Heartbeat Monitoring" on page 71.

> **Note:** 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`(8) man page.

See also:

- "Protect Data Integrity on All Nodes" on page 57

- "Define a Node with the GUI" on page 188

- "Create or Modify a Node with `cxfs_admin`" on page 253

Avoid CXFS Kernel Heartbeat Issues on Large Systems

To avoid CXFS kernel heartbeat issues on large SGI 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 497.

- 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) :

- Allocate 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 a 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= [ NumberOfNodes*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. For more information, see:

- "Configuring XVM Path Failover" on page 145

- *XVM Volume Manager Administrator Guide*

  – 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 497.

- 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" on page 413.

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.

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 7 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.)

• If you are using NFS, you should have the NFS server run on the active metadata server.

Note: If your NFS client applications use file locks, you should either run in an HA configuration or export a given filesystem from only one node. When exporting NFS filesystems from multiple nodes without HA, file locks taken by NFS clients will not be preserved across NFS server outages.

• If you are using CIFS, the SAMBA server can export filesystems from any node in the CXFS cluster, but only one node at a time.

• 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.

Enable Forced Unmount When Appropriate

Normally, an unmount operation will fail if any process has an open file on the filesystem. The *forced unmount* feature allows the unmount to proceed regardless of whether the filesystem is still in use.

If you enable the forced unmount feature for CXFS filesystems (which is turned off by default), you may be able to improve the stability of the CXFS cluster, particularly in situations where the filesystem must be unmounted. However, be aware that forced

unmount will kill running processes to unmount a filesystem, which is potentially destructive.

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

- "Unmount CXFS Filesystems with the GUI" on page 224

- "Create or Modify a CXFS Filesystem with `cxfs_admin`" on page 277

- "Unmount a CXFS Filesystem with `cxfs_admin`" on page 283

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 a Cluster with `cxfs_admin`" on page 271.

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:

```
server-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 "Dynamic Parameters for Debugging Purposes Only" on page 510.

⚠ **Caution:** These parameters are provided for debugging purposes. You should only reset these parameters if advised to do so by SGI support.

Use Proper Storage Management Procedures

You should configure storage management hardware and software according to its documentation and use proper storage management 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 SAN 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

Run Samba Appropriately

If you are using Samba, you should have the Samba server run on the active metadata server. You must not use multiple Samba servers to export the same CXFS filesystem. For more information, see "Samba Differences" on page 33.

Install `ipmitool` on Server-Capable Administration Nodes

The `crsd` daemon issues a system reset via `ipmitool`. Therefore, you must install `ipmitool` on any server capable administration node.

Use the LSI Drivers that Ship with the Linux OS

You should use the LSI drivers that ship with the Linux operating systems. The newer drivers that are available on the LSI web site may not work and are not supported by SGI or the kernels CXFS uses in this release.

Run the `xvm` Command in Cluster Mode

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*.

If you define filesystems when you are in XVM's local domain, you will not see those filesystems when you later start up CXFS. (They will show up as `foreign` devices because they are not part of the cluster domain.) 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 Guide*.

Specify XVM Path Failover and HBA Usage in `failover2.conf`

You should use the `failover2.conf` file to specify path failover for RAIDs and to set HBA usage for each LUN. The values set in the `failover2.conf` file for a particular RAID controller should be identical on every node in the CXFS cluster.

See "Configuring XVM Path Failover" on page 145, and the chapter about XVM path failover in *XVM Volume Manager Administrator Guide*.

Prevent Stalled-Recovery Timeout in a Non-HA DMF Environment

If you use CXFS and DMF in a non-HA environment, you must disable the stalled-recovery timeout feature for all potential CXFS metadata servers of DMF-managed filesystems. This will prevent a recovery timeout while waiting for DMF to be manually started.

For example, add the following lines to the `/etc/modprobe.d/sgi-cxfs-xvm.conf` file on all potential CXFS metadata servers for the DMF-managed filesystems:

```
# Disable recovery timeout feature to allow for
# manual startup of DMF on the potential MDS during recovery
options sgi-cell cxfs_recovery_timeout_stalled=0
```

See "`cxfs_recovery_timeout_stalled`" on page 516.

Configure Appropriately for IPv6

This section discusses the following:

- "IPv6: Use a Link-Local Address for the Private Network" on page 74

- "IPv6: Use Any Legitimate Address Representation" on page 75

- "IPv6: Define `failover_net` to Order Addresses for Automatic CXFS Configuration" on page 75

IPv6: Use a Link-Local Address for the Private Network

You should use a link-local IPv6 address (one that begins with `fe80::`) for the CXFS private network.

IPv6: Use Any Legitimate Address Representation

When using an IPv6 address as input to a CXFS command, you can use any legitimate IPv6 text representation as defined by Internet Engineering Task Force (IETF) request for comment (RFC) 4291. However, the CXFS tools and the examples in this book represent the addresses in the shortest recommended form, as provided in IETF RFC 5952. For example:

```
fe80::230:48ff:fe7b:1bf7
```

For more information, see:

- http://tools.ietf.org/html/rfc5952

- http://tools.ietf.org/html/rfc4291

IPv6: Define `failover_net` to Order Addresses for Automatic CXFS Configuration

If you are using automatic CXFS configuration in an IPv6 environment, you must use the `failover_net` attribute to the `cxfs_admin`(8) command to reprioritize the list of available IP addresses, so that the preferred IPv6 address is chosen. (By default, IPv4 addresses are listed first.) See:

- "Automatically Configure a Client-Only Node" on page 250

- "Create or Modify a Node with `cxfs_admin`" on page 253

- "Network Failover Modification Tasks with `cxfs_admin`" on page 286

Use RAID Mirroring Not XVM Mirroring

In practice, XVM mirroring is not appropriate in a CXFS cluster because the mirror must revive every time a node drops out of the cluster without first unmounting the filesystem. A revive will be triggered by any server or client recovery.

If you require mirroring, you should use the mirroring capability provided by the hardware, such as RAID 1 or RAID 10.

Avoid Dropping Out-Of-Order Frames

To avoid dropping out-of-order frames when using Brocade switches, ensure that the in-order-delivery (IOD) feature `route.delayReroute` is set to `1` as displayed by the

Brocade `configShow` output. (This is the default for firmware releases greater than 6.0.0.) See "Enabling In-Order-Delivery of Packets" on page 107.

Suppress Change Notification for Switch Ports Connected to Nodes

To prevent unnecessary interruptions in the cluster, enable the suppression of change notification if the port is connected to a host HBA and disable the suppression for all other ports. See:

* "Suppressing RSCN" on page 108

* "QLogic® Fibre Channel Switch" on page 108

Administration Best Practices

This section discusses the following administration topics:

* "Back Up the Cluster Database" on page 78

* "Change the Brocade™ Password when Prompted" on page 78

* "Do Not Run User Jobs on Server-Capable Administration Nodes" on page 78

* "Do Not Run Backups on a Client Node" on page 78

* "Use `cron` Jobs Properly" on page 78

* "Repair Filesystems with Care" on page 79

* "Defragment Filesystems with Care" on page 79

* "Use Relocation Properly" on page 80

* "Shut Down Nodes Unobtrusively" on page 80

* "Remove Unused Cluster Components" on page 81

* "Use `fam` Properly" on page 81

* "Upgrade the Software Properly" on page 82

* "Use Fast Copying for Large CXFS Files" on page 83

* "Do Not Change Log File Names " on page 83

- "Rotate Log Files" on page 84
- "Use System Capacity Wisely" on page 84
- "Reboot Before Changing Node ID or Cluster ID" on page 84
- "Reboot a Removed Node Before Returning it to the Cluster Definition" on page 85
- "Restart CXFS on a Node after an Administrative CXFS Stop" on page 85
- "Bring Up the Cluster In an Orderly Fashion" on page 86
- "Disable Nodes that Affect Membership Before Maintenance" on page 86
- "Disable Reset If You Remove Reset Capability" on page 86
- "Avoid Performance Problems with Unwritten Extent Tracking" on page 87
- "Avoid Performance Problems with Exclusive Write Tokens" on page 87
- "Set System-Tunable Kernel Parameters Appropriately" on page 87
- "Keep the `telnet` or `ssh` Port Open On the Switch" on page 88
- "Solve Problems Efficiently" on page 88
- "Do Not Overfill CXFS Filesystems" on page 88
- "Use a Time Synchronization Application" on page 88
- "Turn Off Local XVM on Linux Nodes if Unused" on page 88
- "After Restart, Verify that All Nodes Use the Preferred XVM Path" on page 89
- "Use Mount Options Appropriately" on page 90
- "Use Improved `mkfs` Options" on page 90
- "Modify `updatedb` to Avoid Unnecessary Load" on page 91
- "Preallocate Space for Directories when Appropriate" on page 91
- "Use a Large Value for the XFS `probe_limit` Parameter" on page 93
- "Do Not Change Debugging Parameters" on page 93
- "Set Up Passwordless `root` `ssh` Access" on page 93
- "Unmount a CXFS Filesystem Before Growing It" on page 94

- "Use a Warm Start for `rpcbind`" on page 94

Back Up the Cluster Database

Before and after making cluster configuration changes, you should back up the cluster database so that you can return to an earlier configuration in case of configuration problems or database corruption. See "Backing Up the Cluster Database with `cdbBackup`" on page 368.

Change the Brocade™ Password when Prompted

If you are prompted to change the Brocade switch password, you should do so in order to make logins to the switch faster.

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 "Dedicate Server-Capable Administration Nodes to CXFS Work" on page 53.

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 the 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, the 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 amount (10% or more) of free space available on the filesystem.

⚠️ **Caution:** You should use xfs_fsr **manually**, and only on the active metadata server for the filesystem; the bulkstat system call has been disabled for CXFS clients. Make sure that the filesystem is idle because xfs_fsr may create a temporary filesystem-full condition and fail other applications that otherwise would succeed.

Use Relocation Properly

Relocation is disabled by default. You should enable relocation only during the time when you intend to perform relocations; otherwise, leave relocation disabled. See "CXFS Relocation Capability" on page 309.

Shut Down Nodes Unobtrusively

Rebooting the metadata server without first shutting down CXFS services can cause the metadata server to panic. Use the proper procedures for shutting down nodes. See "Cluster Member Removal and Restoration" on page 336.

When you shut down a server-capable administration node, it is unable to unmount the filesystems for which it is the active metadata server and is therefore unable to shut down gracefully. See "Relocation Error" on page 434.

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 the 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 204

- "Delete a Node with `cxfs_admin`" on page 262

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

In a production CXFS cluster, 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. For details, see the platform-specific release notes and "CXFS Release Versions and Rolling Upgrades" on page 301.

Do the following when upgrading the software:

- Save the current CXFS configuration as a precaution before you start an upgrade. See Chapter 13, "Cluster Database Management" on page 365.

- Read the release notes and any late-breaking caveats on Supportfolio before installing or upgrading CXFS. These contain important information needed for a stable install/upgrade.

- Acquire new CXFS licenses (if required). See Chapter 5, "CXFS Licensing" on page 113.

  Note: Keep your prior licenses available in the event that you must downgrade. If /etc/lk/keys.dat contains licenses from a prior release, they will be ignored after you upgrade.

- Upgrade all server-capable administration nodes before any client-only nodes (including any DMF parallel data-mover nodes.)

  ⚠ **Caution:** Relocation to a server-capable administration node that is running an older CXFS version is not supported.

  In some cases, improper upgrading can result in a loss of functionality. For example, if CXFS client-only nodes, CXFS edge-serving nodes, or DMF parallel data-mover nodes are updated first (before the active metadata server), those nodes might not be able to mount the CXFS filesystems.

- Verify that the upgrade of all nodes is complete and the cluster is running normally before you make any other configuration changes to the cluster (such as adding new nodes or filesystems).

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 7.0.3) on all server-capable administration nodes and client-only nodes.

⚠ **Caution:** Operating a cluster with client-only nodes running a mixture of older and newer CXFS versions may result in a performance loss. Although client-only nodes 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.

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`.

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.

Note: Some `cxfscp` options are platform-specific, and other limitations apply.

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 331.

Rotate Log Files

Periodically, you should rotate log files to avoid filling the disk space; see "Log File Management" on page 331. 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 209.

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 51

- "Dedicate Server-Capable Administration Nodes to CXFS Work" on page 53

- "Out of Logical Swap Space" on page 436

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 468.

Reboot a Removed Node Before Returning it to the Cluster Definition

If you remove a node from the cluster definition (the list of nodes that are eligible to be members of the cluster), you must reboot it before adding it back into the cluster definition in order to avoid cell-ID issues.

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> enable cmsd
```

See:

- "Disable a Node with cxfs_admin" on page 263
- "Enable a Node with cxfs_admin" on page 263
- "Enable or Disable CXFS Kernel Membership (cmsd) for the Local Node" on page 267
- "Forced CXFS Shutdown: Revoke Membership of Local Node" on page 328

Bring Up the Cluster In an Orderly Fashion

SGI recommends that you do the following to bring up 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. Enable 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. Enable CXFS services on the client-only tiebreaker node:

   cxfs_admin:*clustername*> **enable node:*tiebreaker_nodename***

3. Enable 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 Nodes that Affect Membership Before Maintenance

You should disable a CXFS server-capable administration node or a client-only tiebreaker node before shutting it down for maintenance or otherwise taking it offline because these types of nodes will affect the CXFS kernel membership quorum calculation as long as they are configured as enabled in the cluster database. See "Disable a Node with `cxfs_admin`" on page 263.

Disable Reset If You Remove Reset Capability

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

• "Modify a Node Definition with the GUI" on page 198

- "Create or Modify a Node with `cxfs_admin`" on page 253

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.

Set System-Tunable Kernel 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 D, "CXFS System Tunable Kernel Parameters" on page 493.

Keep the `telnet` or `ssh` Port Open On the Switch

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

> ⚠️ **Caution:** **Do not** perform a login to the switch and leave the session connected.

Solve Problems Efficiently

To solve problems efficiently, understand the information in "Troubleshooting Strategies" on page 393 and gather the information discussed in "Reporting Problems to SGI" on page 472.

Do Not Overfill CXFS Filesystems

For best performance, keep the CXFS filesystems under 98% full. This is a best practice for a local filesystem, but is even more important for a CXFS filesystem because of fragmented files and increased metadata traffic.

Use a Time Synchronization Application

Some CXFS operations apply a client timestamp to a file and other operations apply a metadata server timestamp, even if a file is only accessed by a single client. If the system clocks on nodes in a CXFS cluster are not synchronized, timestamps on shared files may appear to change erratically and be unreliable. Therefore, SGI highly recommends that you use a time synchronization application such as Network Time Protocol (NTP).

Turn Off Local XVM on Linux Nodes if Unused

If you do not have a local XVM volume on the Linux system, you should turn off the `boot.xvm` script to avoid unnecessarily probing all of the disks and LUNs to which the machine has access. Do the following:

```
# chkconfig boot.xvm off
```

After Restart, Verify that All Nodes Use the Preferred XVM Path

After restarting a node or after any failures in the fabric, you should verify that all nodes in the cluster are using the preferred XVM path. To perform the verification, you must use the following command on each node:

```
xvm show -v phys | grep preferred
```

If the preferred path is being used, the string `<currentpath>` is displayed (line breaks shown here for readability):

```
# xvm show -v phys | grep preferred
/dev/disk/by-path/pci-0002:00:01.0-fc-0x202400a0b85a42c2:0x0000000000000000
  <dev 2176> affinity=1 preferred <current path>

/dev/disk/by-path/pci-0002:00:01.1-fc-0x202500a0b85a42c2:0x0001000000000000
  <dev 2288> affinity=2 preferred <current path>

/dev/disk/by-path/pci-0002:00:01.1-fc-0x200500a0b8184876:0x0001000000000000
  <dev 2256> affinity=2 preferred <current path>

/dev/disk/by-path/pci-0002:00:01.0-fc-0x200400a0b8184876:0x0000000000000000
  <dev 2080> affinity=1 preferred <current path>
```

The following shows that no LUN is using the preferred path because `<current path>` is not displayed (line breaks shown here for readability):

```
# xvm show -v phys | grep preferred
/dev/disk/by-path/pci-0002:00:01.0-fc-0x202400a0b85a42c2:0x0000000000000000
  <dev 2176> affinity=1 preferred

/dev/disk/by-path/pci-0002:00:01.1-fc-0x202500a0b85a42c2:0x0001000000000000
  <dev 2288> affinity=2 preferred

/dev/disk/by-path/pci-0002:00:01.1-fc-0x200500a0b8184876:0x0001000000000000
  <dev 2256> affinity=2 preferred

/dev/disk/by-path/pci-0002:00:01.0-fc-0x200400a0b8184876:0x0000000000000000
  <dev 2080> affinity=1 preferred
```

Use Mount Options Appropriately

To see the mount options supported by CXFS, see the appendix in *CXFS 7 Client-Only Guide for SGI InfiniteStorage.* This section discusses the following:

- "Do Not Use Both `dmi` and `filestreams` Mount Options" on page 90

- "Use `filestreams` for Disk Layout Optimization (Approved Media Customers Only)" on page 90

Do Not Use Both `dmi` and `filestreams` Mount Options

Do not use both the `dmi` mount option for DMF and the `filestreams` mount options for the filestreams allocator. DMF is not able to arrange file extents on disk in a contiguous fashion when restoring offline files. This means that a DMF-managed filesystem most likely will not maintain the file layouts or performance characteristics normally associated with filesystems using the `filestreams` mount option.

Use `filestreams` for Disk Layout Optimization (Approved Media Customers Only)

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 356.

Use Improved `mkfs` Options

A number of `mkfs` options have been introduced that improve performance. For more information, see the `mkfs.xfs`(8) man page.

However, these options are not compatible with all CXFS platforms. RHEL 4 and RHEL 5 clients cannot mount filesystems built with `lazy-count=1`. For these clients, you must build the filesystems with the following required options:

- RHEL 4 (any update):

  ```
  server# mkfs -t xfs -l lazy-count=0 -i attr=1
  ```

- RHEL 5 (any update):

  ```
  server# mkfs -t xfs -l lazy-count=0
  ```

Depending upon version of the `mkfs.xfs` program that is installed, these options may or may not be the default. The parameters used are printed by `mkfs.xfs` when run, and can also be obtained later by using the following command on SLES systems:

```
sles# xfs_info mountpoint
```

Modify `updatedb` to Avoid Unnecessary Load

CXFS filesystems are mounted on each node in the cluster. Therefore, running the default `updatedb` or `slocate` on each Linux client will cause extra unnecessary load in the cluster. To avoid this situation, add CXFS mount points to the parameter in the following files:

- RHEL: `PRUNEPATHS` in `/etc/updatedb.config`
- SLES: `UPDATEDB_PRUNEPATHS` in `/etc/sysconfig/locate`

Preallocate Space for Directories when Appropriate

Applications that use small I/O (such as Windows copy and Windows Explorer drag-and-drop) have a high ratio of metadata transfer and can cause the filesystem to become fragmented quickly. Heavy I/O traffic and fragmentation can degrade performance.

To improve the performance of these applications, you can use the `xfs_io extsize` command after you create a new directory that will contain files written with small I/O; the `extsize` command causes the filesystem to preallocate the specified disk space when writing files.

Note: This setting affects new allocations only. When the application closes the file, the extra space is trimmed and returned to the filesystem.

The appropriate `extsize` value for a given directory depends upon the RAID configuration and the expected file sizes in that directory. Different directories within the same filesystem can have different `extsize` values, to accommodate differences in workflow.

Note: Preallocation may cause a delay when a file is opened for writing. If you set the size to a value that is far larger than the expected file size, there will be an unnecessary delay each time the file is opened for writing.

The `extsize` value is copied to subdirectories when they are created. If you set `extsize` initially for a new directory, you can later change it for that directory at any time without having to reboot or remount the filesystem. This change by default does not affect existing subdirectories, but new subdirectories will be set to the new `extsize` value.

To set the preallocation size for a new directory, run the following command on the CXFS metadata server:

MDS# **xfs_io -c 'extsize** *preallocation_size directory***'**

where:

| *preallocation_size* | Specifies the size of the block to be preallocated |
| *directory* | Specifies the directory that will use preallocation |

For more information, see the `xfs_io`(8) man page.

As an illustration, consider the following sequence of events and results:

1. Create the new directory `/small`.

2. Set preallocation to a 4-MB block for the `/small` directory:

   MDS# **xfs_io -c 'extsize 4m /small'**

3. Create the subdirectory `/small/subdir1`. This sets `extsize` to 4-MB for `/small/subdir1`.

4. Change preallocation for `/small` to 8-MB:

   MDS# **xfs_io -c 'extsize 8m /small'**

5. Create the subdirectory `/small/subdir2`. This sets `extsize` to 8-MB for `/small/subdir2`. (However, `/small/subdir1` remains at 4-MB.)

6. Change preallocation for /small and all of its subdirectories to 10-MB:

   MDS# **xfs_io -c 'extsize -R 2m /small'**

   This sets extsize to 2-MB for both /small/subdir1 and /small/subdir2.

Use a Large Value for the XFS `probe_limit` Parameter

On a CXFS Linux client, a large value for the XFS probe_limit system-tunable kernel parameter can provide a significant improvement in streaming buffered-I/O performance. You should set probe_limit to the typical file size for the system's workload. (The value for probe_limit has minimal impact on local XFS filesystems and CXFS filesystems on the active metadata server.) For more information, see *XFS for Linux Administration*.

Do Not Change Debugging Parameters

You should only change the CXFS system tunable kernel parameters listed in "Site-Configurable Parameters" on page 497.

Do not change the parameters listed in "Debugging Parameters" on page 507. Debugging parameters are potentially dangerous and you should only change them at the direction of SGI Support

Set Up Passwordless `root ssh` Access

In order to run the cxfsdump(8) command across the cluster, there must be passwordless root ssh access from a server-capable administration node to all of the other nodes in the cluster. (Reverse access from the client-only nodes to the server-capable administration node is not needed.) See "Cluster Information Gathering Tool (cxfsdump)" on page 407.

Note: If your cluster includes Windows nodes, run cxfsdump manually on those nodes. See the Windows chapter of *CXFS 7 Client-Only Guide for SGI InfiniteStorage.*

Unmount a CXFS Filesystem Before Growing It

You should unmount a filesystem from the CXFS cluster and mount it locally before growing it. For more information, see "Growing Filesystems " on page 333.

Use a Warm Start for `rpcbind`

If you must restart the rpcbind(8) utility, you should execute a *warm start* by using the -w option in order to retain the registration of the services on which CXFS depends.

Also see "Issues after Restarting rpcbind" on page 426

SGI RAID for CXFS Clusters

This chapter discusses SGI RAID for CXFS clusters:

- "RAID Hardware" on page 95
- "RAID Firmware" on page 96
- "Number of LUNs Supported" on page 98
- "Installing InfiniBand RAID" on page 100
- "RAID Verification" on page 101

For additional updates, see the CXFS release notes.

RAID Hardware

CXFS supports the following RAID hardware:

- Fibre Channel RAID:

  SGI InfiniteStorage 17000
  SGI InfiniteStorage 16000
  SGI InfiniteStorage 15000
  SGI InfiniteStorage 11000
  SGI InfiniteStorage 10000
  SGI InfiniteStorage 6700
  SGI InfiniteStorage 6120
  SGI InfiniteStorage 5500
  SGI InfiniteStorage 5000
  SGI InfiniteStorage 4600
  SGI InfiniteStorage 4500
  SGI InfiniteStorage 4100
  SGI InfiniteStorage 4000
  SGI InfiniteStorage 220
  SGI RM660
  SGI RM610
  SGI TP9700
  SGI TP9500S (serial ATA)
  SGI TP9500

SGI TP9400
SGI TP9300S (serial ATA)
SGI TP9300
SGI TP9100
SGI S330

• InfiniBand RAID:

SGI InfiniteStorage 17000
SGI InfiniteStorage 16000
SGI InfiniteStorage 15000
SGI InfiniteStorage 10000
SGI InfiniteStorage 5500
SGI InfiniteStorage 5000
SGI InfiniteStorage 4600

• SAS RAID:

SGI InfiniteStorage 5600
SGI InfiniteStorage 5500
SGI InfiniteStorage 5000

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 InfiniteStorage 11000 and 15000 supports controller firmware version 6.02 or later.

• SGI InfiniteStorage 6700 supports controller firmware version V3.00 or later.

• SGI InfiniteStorage IS4600 supports controller firmware version 7.36 or later

• SGI InfiniteStorage IS4100 supports controller firmware version 7.60 or later

• SGI RM610 and RM660 running version 5.12b or later.

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

- 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 1Gb 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).

- TP9100 2Gb requires firmware 9.03 or later

- 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 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).

For more information, see "Using SGI Knowledgebase" on page 472.

See also "XVM Path Failover Overview" on page 145.

Number of LUNs Supported

By default, the RAID firmware and/or RAID controller supports a maximum number of logical units (LUNs), as shown in Table 3-1.

In some cases, you can obtain a separate software-enabling key that key will support a larger number of LUNs in separate partitions; this requires that the Fibre Channel ports be mapped to a partition. Contact your SGI sales representative for the SGI software partitioning key.

Table 3-1 Number of LUNs Supported

Firmware	Controller	Default LUN Maximum	Optional 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
06.15.*xx.xx*		1024	
06.16.*xx.xx*		1024	
06.17.*xx.xx*		62	2048
06.19.*xx.xx*		2048	
06.60.*xx.xx*		2048	

Firmware	Controller	Default LUN Maximum	Optional LUN Maximum with a Partitioning Key
07.10.*xx.xx*	IS4000	2048	
07.10.*xx.xx*	IS4500/IS4700	4096	
07.35.*xx.xx*		2048	
07.36.*xx.xx*	IS4000	2048	
07.36.*xx.xx*	IS4500, IS4600, IS4700	4096	
07.50.*xx.xx*	IS4000	2048	
07.50.*xx.xx*	IS4500, IS4600, TP9700	4096	
07.60.*xx.xx*	IS4100	4096	
07.60*xx.xx*	IS4600	4096	
	IS6120	2048	
	IS6700	512	
	IS11000	1024	
	IS15000	1024	
	RM610	256	
	RM660	512	

Installing InfiniBand RAID

This section discusses the following:

- "Installing InfiniBand RAID on RHEL" on page 100
- "Installing InfiniBand RAID on SLES" on page 100

Installing InfiniBand RAID on RHEL

To install InfiniBand RAID on RHEL, do the following:

1. Ensure that the required software is installed:

   ```
   rhel# yum groupinstall "Infiniband Support"
   rhel# yum install srptools.x86_64
   rhel# yum install srptools-debuginfo.x86_64
   ```

2. Edit the /etc/rdma/rdma.conf file to use the following setting:

   ```
   SRP_LOAD=yes
   ```

   For example:

   ```
   rhel# sed -i 's/SRP_LOAD=no/SRP_LOAD=yes/' /etc/rdma/rdma.conf
   ```

3. Set the rdma and srpd services to start at boot time:

   ```
   rhel# chkconfig rdma on
   rhel# chkconfig srpd on
   ```

4. Start the services:

   ```
   rhel# service rdma start
   rhel# service srpd start
   ```

Installing InfiniBand RAID on SLES

To install InfiniBand RAID on SLES, do the following:

1. Ensure that the required software is installed:

   ```
   sles# zypper in --type pattern OFED srptools
   ```

2. Edit the `/etc/sysconfig/infiniband` file to use the following settings:

```
SRPHA_ENABLE=yes
SRP_DAEMON_ENABLE=yes
```

For example:

```
sles# sed -i 's/SRPHA_ENABLE=no/SRPHA_ENABLE=yes/' /etc/sysconfig/infiniband
sles# sed -i 's/SRP_DAEMON_ENABLE=no/SRP_DAEMON_ENABLE=yes/' /etc/sysconfig/infiniband
```

3. Set the `openibd` service to start at boot time:

```
sles# chkconfig openibd on
```

4. Start the service:

```
sles# service openibd start
```

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.

Switch Configuration

This chapter discusses the following:

- "Brocade Switch" on page 103
- "QLogic® Fibre Channel Switch" on page 108

Note: No configuration specific to CXFS is required for InfiniBand and serial-attached storage (SAS) switches.

Brocade Switch

This section discusses the following:

- "Brocade Firmware" on page 103
- "Verifying the Brocade Switch Firmware Version" on page 104
- "Verifying the Brocade License" on page 105
- "Limiting Login Sessions" on page 105
- "Changing the Brocade FC Cable Connections" on page 107
- "Enabling In-Order-Delivery of Packets" on page 107
- "Suppressing RSCN" on page 108

See also "Change the Brocade™ Password when Prompted" on page 78.

Brocade Firmware

All Brocade switches contained within the SAN fabric must have the appropriate Brocade firmware. For the currently supported firmware, see the CXFS general release notes.

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 via the `telnet` (default) or `ssh` from all CXFS administration nodes. The fencing network connected to the Brocade switch must be physically separate from the private heartbeat network.

⚠ **Caution:** In order for I/O fencing to succeed, the `admin` user must not be logged in.

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

For more information, see "Using SGI Knowledgebase" on page 472.

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 examples:

```
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 Login Sessions

You must limit the maximum allowed simultaneous `telnet` or `ssh` login sessions:

- "Brocade 12000/24000/48000 Models" on page 105
- "Other Brocade Models" on page 106

Brocade 12000/24000/48000 Models

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

1. Connect to the switch 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` (for either `telnet` or `ssh`) 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
```

Other Brocade Models

To limit the maximum allowed simultaneous telnet or ssh login sessions for the admin user to 1 on the Brocade 200E/300/3250/3252/3850/3852/3900/4100/4900/5000/5100/5300 models, do the following:

1. Connect to the switch as root.

2. Issue the sync command to avoid filesystem corruption:

```
# sync
```

3. Edit the /etc/profile file to change the max_telnet_sessions (for either telnet or ssh) 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
```

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.

Enabling In-Order-Delivery of Packets

To avoid dropping frames, ensure that the in-order-delivery (IOD) feature route.delayReroute is enabled (set to 1) as displayed by the Brocade configShow output.

For example, the following output shows that the feature is disabled (set to 0):

```
brocade:admin> configshow
date = Mon Mar 11 08:30:26 2013
[Switch Configuration Begin : 0]
...
route.delayReroute:0
...
```

To change the setting to 1, enter the iodSet command:

```
brocade:admin> iodSet
```

Note: You can change this setting without rebooting the switch.

The output would then show the feature is enabled (set to 1):

```
brocade:admin> configshow
date = Mon Mar 11 08:30:26 2013
[Switch Configuration Begin : 0]
...
route.delayReroute:1
...
```

Suppressing RSCN

Enable (turn ON) registered state change notification (RSCN) suppression if the port is connected to a host HBA or disable (turn OFF) suppression for all other ports. Use the portcfgshow command to display the current settings.

Use the following command on the switch:

```
switch> portcfg rscnsupr [Slot/]Port[-Range] --enable|--disable
```

For example, suppose that ports 4 through 7 go from the switch to nodes in the cluster. You would enter the following to enable RSCN suppression for ports 4 through 7:

```
switch> portcfg rscnsupr 4-7 --enable
```

```
switch> portcfgshow
```

```
Locked L_Port      .. .. .. ..   .. .. .. ..   .. .. .. ..   .. .. .. ..
Locked G_Port      .. .. .. ..   .. .. .. ..   .. .. .. ..   .. .. .. ..
Disabled E_Port    .. .. .. ..   .. .. .. ..   .. .. .. ..   .. .. .. ..
ISL R_RDY Mode     .. .. .. ..   .. .. .. ..   .. .. .. ..   .. .. .. ..
RSCN Suppressed    .. .. .. ..   ON ON ON ON   .. .. .. ..   .. .. .. ..
Persistent Disable.. ON ON ON    .. ON ON ON   ON ON ON ON   ON .. .. ..
NPIV capability    ON ON ON ON   ON ON ON ON   ON ON ON ON   ON ON ON ON
```

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 the CXFS general release notes.

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 for all other ports. (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 UV® 100 system:

```
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] UV100
   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        UV100
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
```

CXFS Licensing

This section discusses the following:

- "Licensing Overview" on page 113
- "Installing the License Keys" on page 119
- "License Key Verification" on page 120

See also "Upgrading Licenses" on page 303.

Licensing Overview

This section discusses the following:

- "Licensing Requirements" on page 113
- "CXFS_CLIENT License Bundles" on page 114
- "Server-side Licensing Flexibility" on page 115
- "Adding Licenses" on page 115
- "Examples of License Requirements" on page 115

Licensing Requirements

CXFS 7.0 and later requires CXFS 7.0 licenses.

CXFS licensing uses the SGI License Key (LK) software to implement a simple node-count scheme. (There is no restriction on the CPU count for either server-capable administration nodes or client-only nodes.)

All license keys are node-locked and installed only on the server-capable administration nodes. Each server-capable administration node should have equivalent keys installed.

You must purchase the following:

- One CXFS_MDS license for each server-capable administration node.

- One set of CXFS_CLIENT licenses equal to the number of client-only nodes in your cluster (see "CXFS_CLIENT License Bundles" on page 114). This entitles you to generate a set of CXFS_CLIENT keys for each licensed server-capable administration node.

- *(Optional)* GRIO2_CLUSTER license if GRIO v2 is enabled

You must install keys for the above on each server-capable administration node; all of the nodes should have equivalent keys installed.

Note: No license keys are installed on the client-only nodes.

If you use an SGI UV 100, UV 1000, UV 2000, or Altix ia64 system as a CXFS client-only node, you must also purchase the CXFS 7.*x* Feature Enabler for SGI NUMAlink systems. (These systems are not supported as CXFS server-capable administration nodes.) A single Feature Enabler is required per system and it does not generate an LK key. For a partitioned NUMAlink system, each partition requires a client license but only one Feature Enabler is required for the entire system.

CXFS_CLIENT License Bundles

The CXFS_CLIENT licenses determine how many client-only nodes can join the cluster. The following bundles are available:

- 1 client
- 5 clients
- 10 clients
- 20 clients
- 50 clients

These licenses are cumulative and can be used in any combination. However, some combinations are more cost-effective than others; larger client bundles have a lower per-client cost.

For more information about the appropriate licenses for your site, contact your SGI Sales representative.

Server-side Licensing Flexibility

Server-side licensing provides flexibility when changing the CXFS cluster configuration. For example, a 5–client license will allow any five client-only nodes to join the CXFS membership. For example, the cluster could have five Windows nodes on Monday and then change to four Mac OS X nodes and a Linux node on Tuesday.

Adding Licenses

To add a new server-capable administration node, you must purchase a new CXFS_MDS license for it and then generate a set of CXFS_CLIENT keys for it. You must install the CXFS_MDS and CXFS_CLIENT keys on the new node.

To increase the number of client-only nodes, you must purchase additional CXFS_CLIENT licenses; you do not have to change existing keys. The purchase of a new CXFS_CLIENT license entitles you to generate a key for it on each licensed server-capable administration node. You must install a key for this new CXFS_CLIENT license on each server-capable administration node in the cluster. See " License Changes as a Cluster Grows" on page 117.

Note: If the license capability is not uniform across the cluster and the active metadata server fails over to a server that has fewer CXFS_CLIENT keys installed, 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.

Examples of License Requirements

This section discusses the following:

- " License Examples for Various Cluster Configurations" on page 115
- " License Changes as a Cluster Grows" on page 117

License Examples for Various Cluster Configurations

The following are the licenses needed for various example clusters:

- A 13–node cluster with 12 client-only nodes and a single server-capable administration node:

  - One CXFS_MDS

  - One CXFS_CLIENT for 10 clients

  - Two CXFS_CLIENT for 1 client

- A 14–node cluster with 12 client-only nodes and 2 server-capable administration nodes:

  - Two CXFS_MDS

  - One CXFS_CLIENT for 10 clients

  - Two CXFS_CLIENT for 1 client

- A 3–node cluster with 1 client-only node and 2 server-capable administration nodes, where the client is an SGI UV 2000 with a single partition:

  - Two CXFS_MDS

  - One CXFS_CLIENT for 1 client

  - One CXFS 7.*x* Feature Enabler for SGI NUMAlink systems

- A 6–node cluster with 4 client-only nodes and 2 server-capable administration nodes, where the client-only nodes are 4 partitions within a single SGI UV 2000:

  - Two CXFS_MDS

  - Four CXFS_CLIENT for 1 client

  - One CXFS 7.*x* Feature Enabler for SGI NUMAlink systems

- A 14–node cluster with 12 client-only nodes, 2 server-capable administration nodes, and GRIO V2 support:

  - Two CXFS_MDS

  - One CXFS_CLIENT for 10 clients

  - Two CXFS_CLIENT for 1 client

  - One GRIO2_CLUSTER license

License Changes as a Cluster Grows

The following figures illustrate the license changes required as a cluster grows from 4 to 6 client-only nodes. Figure 5-1 shows that there is one CXFS_CLIENT license that is not being used by the current configuration. .
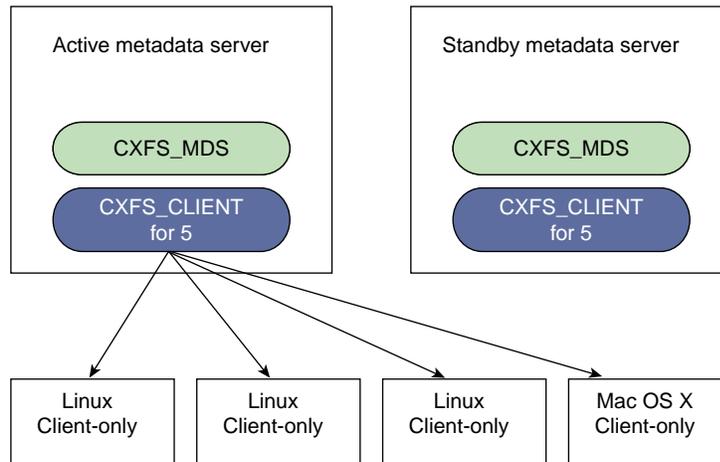
Figure 5-1 Cluster with a 5–Client License and Four Client-only Nodes

Figure 5-2 shows that a fifth client-only node can be added without any change in the CXFS_CLIENT licenses (which are now maximized). It also shows that the licenses can apply to client-only nodes running any supported OS.
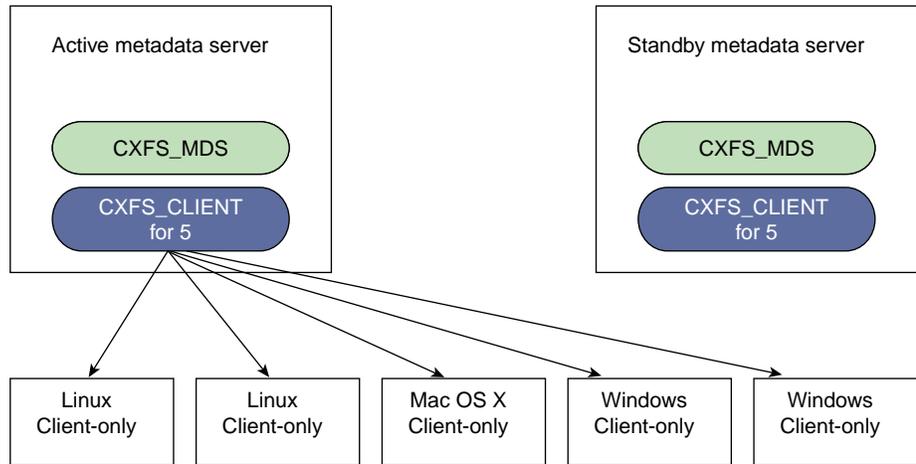
Figure 5-2 Adding a Fifth Client-only Node and Changing the OS Composition

Figure 5-3 shows that a new CXFS_CLIENT license must be purchased and a key installed on each server-capable administration node.
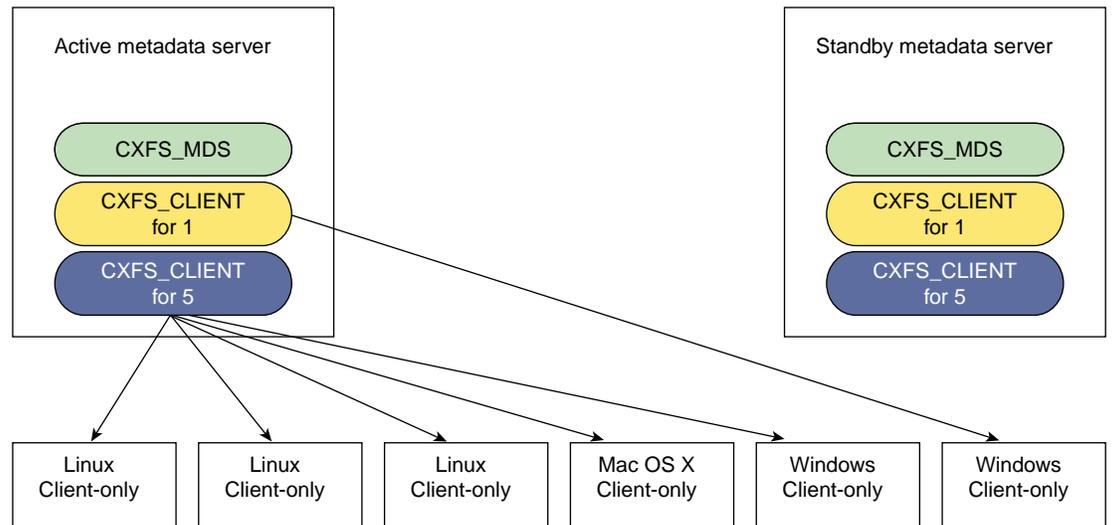
Figure 5-3 Adding a Sixth Client Requires a New License

Installing the License Keys

When you order CXFS, you will receive an entitlement ID for the licenses you purchased (one entitlement ID for each CXFS_MDS, each CXFS_CLIENT bundle, and each GRIO2_CLUSTER license; the CXFS_CLIENT and GRIO2_CLUSTER entitlements can generate multiple keys, one set per CXFS_MDS license). You must submit the system host ID, host name, and entitlement IDs when requesting your permanent CXFS license keys.

This section discusses the following:

- "Gathering the Host Information" on page 119
- "Obtaining the Keys from SGI" on page 120
- "Copying the Keys to the /etc/lk/keys.dat File" on page 120
- "Restarting fs2d After Installing or Upgrading Licenses" on page 120

Gathering the Host Information

To obtain the host information for a server-capable administration node, execute the following command:

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

For example, the following shows that the serial number is 000423d5fd92 and the license ID is 23d5fd92:

```
server-admin# lk_hostid
000423d5fd92 23d5fd92 socket=1 core=2 processor=2
#----------------------------------------------------------------------------
#The above is the default selected by lk_hostid. See below for additional
#hostid pairs.
#----------------------------------------------------------------------------
#Interface  SN                  LI          Driver ( Comment )
#----------------------------------------------------------------------------
 eth0       000423d5fd92        23d5fd92    e1000
 eth1       000423d5fd93        23d5fd93    e1000
```

Obtaining the Keys from SGI

To obtain your CXFS license keys, see information provided in your customer letter and 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.

Copying the Keys to the `/etc/lk/keys.dat` File

To install the license keys, copy them into the `/etc/lk/keys.dat` file on the server-capable administration nodes.

Note: CXFS will generate warnings in the system log if the license keys are not equivalent on all server-capable administration nodes in the cluster.

Restarting `fs2d` After Installing or Upgrading Licenses

After you install or upgrade licenses on an existing CXFS server-capable administration node, you must restart the `fs2d` daemon:

```
server-admin# service cxfs_cluster restart
```

License Key Verification

This section discusses the following:

- "Displaying the Keys with `lk_verify`" on page 121
- "Displaying the Keys with `cxfslicense` After Installing CXFS" on page 121
- "Displaying the Keys with `cxfs_admin` After Installing CXFS" on page 124

See also "License Key Error" on page 437.

Displaying the 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:

```
serveradmin# lk_verify -A -VVV
LK(License Keys) - Version SGI_SFS_2.3 2.2.0- Built Jan 12 2012 23:00:28-Vendor SGI- Silicon Graphics International

  1 /etc/lk/keys.dat:187    product=CXFS_CLIENT, version=7.000, count=0, begDate=1355516270, \
    expDate=1363323599, licenseID=201e9247, key=bLEHGcIya4N5icWKeLOQyFBjM7jT66Ci, \
    info='CXFS CLIENT 1 NODE',attr='1', vendor='Silicon Graphics International', \
    ref_id='267783'
       Verdict:     SUCCESS.

  2 /etc/lk/keys.dat:192    product=CXFS_CLIENT, version=7.000, count=0, begDate=1355516339, \
    expDate=1363323599, licenseID=201e9247, key=OSgtcSMJfQFN7LpQbavxz4Sm3BXx+ieK, \
    info='CXFS CLIENT 5 NODE',attr='5', vendor='Silicon Graphics International', \
    ref_id='267784'
       Verdict:     SUCCESS.

  3 /etc/lk/keys.dat:197    product=CXFS_CLIENT, version=7.000, count=0, begDate=1355516393, \
    expDate=1363323599, licenseID=201e9247, key=cqBOStuzITV85TwdhrgAOH4fPu0US9j/, \
    info='CXFS CLIENT 10 NODE',attr='10', \
    vendor='Silicon Graphics International',ref_id='267785'
       Verdict:     SUCCESS.

  4 /etc/lk/keys.dat:202    product=CXFS_CLIENT, version=7.000, count=0, begDate=1355516454, \
    expDate=1363323599, licenseID=201e9247, key=w3AYBzkqvQXj9VnMK2GUZ5otkgNoOy81, \
    info='CXFS CLIENT 20 NODE',attr='20', \
    vendor='Silicon Graphics International',ref_id='267786'
       Verdict:     SUCCESS.

  5 /etc/lk/keys.dat:218    product=CXFS_MDS, version=7.000, count=0, begDate=1355770754, \
    expDate=1363582799, licenseID=201e9247, key=PShg1XGVwu+d6YAx0RyN/FzLx6LwWoli, \
    info='CXFS 7.0 MDS', vendor='Silicon Graphics International',ref_id='268052'
       Verdict:     SUCCESS.
```

Displaying the Keys with `cxfslicense` After Installing CXFS

To verify that the license keys have been installed properly, use the `cxfslicense` `-d` command **after installing the CXFS software** (see Chapter 7, "Server-Capable

Administration Node Installation" on page 131). Licensing errors will be reported to the fs2d log.

This section contains sample output:

- "Valid Licenses" on page 122
- "No Metadata Server License" on page 122
- "Valid Metadata Server License without Client Licenses" on page 123
- "No Licenses Found" on page 123

Valid Licenses

The following output shows that the licenses are correctly in place:

```
serveradmin# cxfslicense -d

License(s) found: 1
Found CXFS 7.0 MDS version 7.0 license for CXFS_MDS serial 268052
Server-side licensing is available

License(s) found: 4
Found license for 1 of CXFS_CLIENT 7.0 serial = 267783.
Found license for 5 of CXFS_CLIENT 7.0 serial = 267784.
Found license for 10 of CXFS_CLIENT 7.0 serial = 267785.
Found license for 20 of CXFS_CLIENT 7.0 serial = 267786.
```

No Metadata Server License

The following output shows that there is no metadata server license, and therefore the client licenses that are installed are ignored:

```
serveradmin# cxfslicense -d

Cannot find valid version 7.0 LK license for CXFS_MDS

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

License(s) found: 4
Found license for 1 of CXFS_CLIENT 7.0 serial = 267783.
```

```
Found license for 5 of CXFS_CLIENT 7.0 serial = 267784.
Found license for 10 of CXFS_CLIENT 7.0 serial = 267785.
Found license for 20 of CXFS_CLIENT 7.0 serial = 267786.


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

Valid Metadata Server License without Client Licenses

The following output shows that the metadata server license was found, but there are
no client licenses:

```
serveradmin# cxfslicense -d

License(s) found: 1
Found CXFS 7.0 MDS version 7.0 license for CXFS_MDS serial 268052
Server-side licensing is available

No licenses available for CXFS_CLIENT 7.0

No client licenses for server-side licensing are available,
CXFS clients will need a client-side license.
```

No Licenses Found

The following output shows that no licenses were found:

```
serveradmin# cxfslicense -d

Cannot find valid version 7.0 LK license for CXFS_MDS

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

No licenses available for CXFS_CLIENT 7.0

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

Displaying the Keys with `cxfs_admin` After Installing CXFS

You can use the `cxfs_admin` command to display license information **after installing the CXFS software** (see Chapter 7, "Server-Capable Administration Node Installation" on page 131). For example, the following is output from the `show licenses` command for the `clusterOne` cluster:

```
serveradmin# cxfs_admin -i cluster1 -c "show licenses"
Connecting to the CXFS server for the "cluster1" cluster...
Event at [ Mar 05 11:59:25 ]
status:licenses:
    cxfs_client:
        allocated=1
        valid=36
```

For example, the following is truncated output from the `status` command for the `cluster1` cluster:

```
serveradmin# cxfs_admin -i cluster1 -c status
Connecting to the CXFS server for the "cluster1" cluster...
Event at [ Mar 05 11:56:32 ]
Cluster        : cluster1
Tiebreaker     :
Client Licenses : allocated 1 of 36
...
```

Also see:

- "`cxfs_admin` and Status" on page 384
- "License Key Error" on page 437

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 125

- "Adding a Private Network" on page 126

- "Verifying the Private and Public Networks" on page 129

- "Modifications Required for CXFS GUI Connectivity Diagnostics" on page 130

- "Configuring `SuSEfirewall2`" on page 130

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 network 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 all nodes 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:

```
10.0.3.1 priv-server1
10.0.3.2 priv-server2
10.0.3.3 priv-client1
190.0.2.1 server1.example.com server1
190.0.2.2 server2.example.com server2
190.0.2.3 client1.example.com client1
```

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.

007–5618–009   127

For example:

```
server-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 "Modifications for CXFS Connectivity Diagnostics" on page 134.

Verifying the Private and Public Networks

For each private network on each server-capable administration node in the pool, verify access with the `ping` command:

1. Execute a `ping` using the private network. Enter the following, where *nodeIPaddress* is the IP address of the node:

   ping *nodeIPaddress*

   For example:

   ```
   server-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
   ```

2. Execute a `ping` using the public network.

3. If `ping` fails, repeat the following procedure on each node:

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

   ```
   server-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.

   b. Verify that the cables are correctly seated.

4. Repeat this procedure on each node.

Modifications Required for CXFS GUI Connectivity Diagnostics

In order to test node connectivity by using the GUI, 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. (This test is not required when using cxfs_admin because it verifies the connectivity of each node as it is added to the cluster.)

There are several ways of accomplishing this, depending on the existing settings in the pluggable authentication modules (PAMs) and other security configuration files.

The following method works with default settings. Do the following on **all nodes** in the cluster:

1. Install the rsh-server RPM.

2. Enable rsh.

3. Restart xinetd.

4. Add rsh to the /etc/securetty file.

5. 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 Linux operating system documentation about PAM and the hosts.equiv man page.

Configuring SuSEfirewall2

Linux uses SuSEfirewall2 or iptables for IP filtering for the CXFS private network. SuSEfirewall2 must either be turned off or configured appropriately. For more information, see the SUSE documentation.

Server-Capable Administration Node Installation

This chapter discusses the following:

- "Limitations and Considerations for Server-Capable Administration Nodes" on page 131
- "Installation Overview for Server-Capable Administration Nodes" on page 133
- "Installation Verification" on page 134
- "Modifications for CXFS Connectivity Diagnostics" on page 134

Limitations and Considerations for Server-Capable Administration Nodes

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

Server-capable administration nodes contain the cluster administration daemons (fs2d, crsd, cad, and cmond), the CXFS control daemon (clconfd), and the cluster database.

The following limitations and considerations apply to server-capable administration nodes in addition to the information in "CXFS Restrictions" on page 9:

- "DMAPI Requirement" on page 132
- "Reuse of a Disk with GPT Labels" on page 132
- "XFS Version 1" on page 132
- "O_EXCL Limitation" on page 132
- "Weak-Updates Messages" on page 133

See also client-only node information in *CXFS 7 Client-Only Guide for SGI InfiniteStorage.*

DMAPI Requirement

DMAPI is required to mount CXFS filesystems. For those nodes that are also DMF servers or DMF parallel data-mover nodes, DMAPI is automatically enabled when installing the `dmf` or `dmf-mover` packages. However, you must manually enable DMAPI for SLES and RHEL client-only nodes that are not running DMF software. See the Linux chapter of the *CXFS 7 Client-Only Guide for SGI InfiniteStorage*.

Reuse of a Disk with GPT Labels

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 Guide* and http://support.sgi.com/content_request/838562/index.html on Supportfolio.

XFS Version 1

CXFS filesystems with XFS version 1 directory format cannot be mounted on Linux nodes.

O_EXCL Limitation

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 on multiple nodes may not get the O_EXCL behavior they requested (creates of the same file from two or more separate nodes may all succeed).

Weak-Updates Messages

If you are installing the CXFS server package on a system that is currently running standalone XVM, you may see messages similar to the following:

```
WARNING: /lib/modules/NNN-smp/weak-updates/xvm/sgi-xvm-cell.ko needs unknown symbol xvm_trace_enter
WARNING: /lib/modules/NNN-smp/weak-updates/xvm/sgi-xvm-cell.ko needs unknown symbol xvm_physlab_ver_to_cur
```

You should ignore these messages and reboot the system as documented in the installation instructions.

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, "CXFS Best Practices" on page 47
Chapter 3, "SGI RAID for CXFS Clusters" on page 95
Chapter 4, "Switch Configuration" on page 103
Chapter 5, "CXFS Licensing" on page 113
Chapter 6, "Preinstallation Steps" on page 125
Chapter 7, "Server-Capable Administration Node Installation" on page 131
Chapter 8, "Postinstallation Steps" on page 137
Chapter 9, "Initial Setup of the Cluster" on page 149

Also see the *CXFS 7 Client-Only Guide for SGI InfiniteStorage*.

Following is the order of installation and configuration steps:

1. Install the operating system. See the CXFS release notes for supported levels.

2. Install and verify the RAID. See Chapter 3, "SGI RAID for CXFS Clusters" on page 95.

3. Install and verify the switch. See Chapter 4, "Switch Configuration" on page 103.

4. Obtain and install the CXFS license keys. See Chapter 5, "CXFS Licensing" on page 113.

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

6. Install the CXFS software. See the *SGI InfiniteStorage Software Platform* release note. The ISSP release note is located in the /docs/README.txt on the DVD and is available on the Supportfolio download page. For a complete list of the RPM packages and their location on the media, see the /docs/RPMS.txt file.

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 186.

8. Install the client-only nodes, as described in *CXFS 7 Client-Only Guide for SGI InfiniteStorage*.

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 sgi-sysadm_base-lib package has installed:

```
# rpm -qa | grep sgi-sysadm_base-lib
sgi-sysadm_base-lib-3.0-sgi200a6.sles11
```

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

To verify licenses, see "License Key Verification" on page 120.

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.

2. Enable `rsh` by changing `disable yes` to `disable no` in the `/etc/xinetd.d/rsh` file.

3. Restart `xinetd`:

   ```
   server-capable# 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.

Postinstallation Steps

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:

- "Configuring /etc/exports on All Nodes" on page 137
- "Configuring Server-Capable Administration Node System Files" on page 138
- "Enabling GRIOv2 *(Optional)*" on page 144
- "Configuring XVM Path Failover" on page 145

After completing the steps discussed in this chapter, see Chapter 9, "Initial Setup of the Cluster" on page 149. For information about upgrades, see "CXFS Release Versions and Rolling Upgrades" on page 301.

Configuring /etc/exports on All Nodes

The optional /etc/exports file on each node describes the filesystems that are being exported to NFS clients. For optimal performance, use the mountpoint (mp) option for CXFS filesystems in /etc/exports that are exported with NFS. For more information, see the Linux exports(5) man page.

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 318.

Configuring Server-Capable Administration Node System Files

This section discusses system files on administration nodes:

- "`/etc/services` on Server-Capable Administration Nodes" on page 138
- "`cad.options` on Server-Capable Administration Nodes" on page 138
- "`fs2d.options` on Server-Capable Administration Nodes" on page 139
- "`clconfd.options` on Server-Capable Administration Nodes" on page 142

`/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         5435/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 (which provides cluster information) is started. The file is located as follows:

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

You can set the following options in the `cad.options` file:

`--append_log`	Appends `cad` logging information to the `cad` log file instead of overwriting it.
`--log_file` *file*	Specifies the `cad` log filename. Alternately, this can be specified as `-lf` *file.*
`--tcp_port`	Specifies a TCP port number to be used by `cad`. The default is port 5435.

-vvvv   Specifies the 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/log/cxfs/cad_log --append_log

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

-vv -lf /var/log/cxfs/cad_log --append_log

The default log file is /var/log/cxfs/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:

server-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
-logdest *log destination*	Sets the log destination: `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`.
-logevents *event name*	Logs selected events: `all`, `internal`, `args`, `attach`, `chandle`, `node`, `tree`, `lock`, `datacon`, `trap`, `notify`, `access`, `storage`. The default is `all`.
-logfile *filename*	Sets the log filename. The default is `/var/log/cxfs/fs2d_log`.
-logfilemax *maximum size*	Sets the 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*	Sets the 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	[Does not] trace to the 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`.

Option	Description
-[no]parent_timer	[Does not] exit when the parent exits. The default is -noparent_timer.
-[no]daemonize	[Does not] run as a daemon. The default is -daemonize.
-l	Does not run as a daemon.
-h	Prints usage message.
-o help	Prints 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/log/cxfs/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:

server-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.

fs2d.options 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/log/cxfs/fs2d_ops1.

  When the size of this file exceeds 100,000,000 bytes, this file is renamed to /var/log/cxfs/fs2d_ops1.old and a new file /var/log/cxfs/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/log/cxfs/fs2d_ops1
-tracefilemax 100000000
```

`fs2d.options` Example 2

> The following example shows an `/etc/config/fs2d.options` file that directs all log and trace messages into one file, `/var/log/cxfs/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/log/cxfs/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 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 209.
-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:

```
server-admin# service cxfs restart
```

Enabling GRIOv2 *(Optional)*

This section discusses enabling and disabling GRIOv2 the following for server-capable administration nodes:

- "Enabling GRIOv2 After Reboot" on page 144
- "Enabling GRIOv2 for the Current Session" on page 144
- "Disabling GRIOv2 After Reboot" on page 144
- "Disabling GRIOv2 for the Current Session" on page 145

Note: GRIOv2 requires a license key on the server-capable administration nodes.

Enabling GRIOv2 After Reboot

To enable GRIOv2 on a server-capable administration node, do the following:

1. Turn the grio2 flag on with the chkconfig(8) command:

   server# **chkconfig grio2 on**

2. Reboot the system.

Enabling GRIOv2 for the Current Session

To enable GRIOv2 for the current session, enter the following:

server# **services grio2 start**

Disabling GRIOv2 After Reboot

To prevent GRIOv2 from being enabled after reboot on server-capable administration node, do the following:

1. Turn the grio2 flag on with the chkconfig(8) command:

   server# **chkconfig grio2 on**

2. Reboot the system.

Disabling GRIOv2 for the Current Session

To disable GRIOv2 for the current session, enter the following:

```
server# services grio2 stop
```

Configuring XVM Path Failover

This section discusses the following:

- "XVM Path Failover Overview" on page 145
- "/etc/failover2.conf File" on page 146
- "xvm Commands Related to XVM Failover" on page 146
- "RAID Units and XVM Failover" on page 147

XVM Path Failover Overview

XVM path failover creates an infrastructure for the definition and management of multiple paths to a single disk device or logical unit (LUN). XVM selects a path for I/O from the host through the fabric to the RAID, using a particular HBA at the host end and a particular controller at the RAID end. XVM can be controlled to select paths that result in the best I/O performance. The key choices are:

- RAID controller selection
- HBA selection

Unnecessary switching between RAID controllers to access a LUN can degrade performance considerably. If more selected paths go through one HBA than through another, resource contention in the first HBA will slow down I/O while the other one is underused.

To avoid these problems, XVM uses path failover and the path manager feature. You must specify appropriate controller groupings, preferred controller paths, and/or HBA usage in the /etc/failover2.conf file on each node in the CXFS cluster as needed.

> **Note:** SGI requires SGIAVT or ALUA mode. See "Changing SGIRDAC Mode to SGIAVT Mode for SGI RAID" on page 528.

/etc/failover2.conf File

For information about generating and editing the /etc/failover2.conf file for Linux and Mac OS X platforms, see the chapter about XVM failover in *XVM Volume Manager Administrator Guide*. For information about generating a failover2.conf file for Windows, see the *CXFS 7 Client-Only Guide for SGI InfiniteStorage*.

Also see:

• The example file installed in /etc/failover2.conf.example

• The comments in the failover2.conf file

xvm Commands Related to XVM Failover

> **Note:** The xvm command is provided on all CXFS platforms. You can use xvm on a client-only node to display configuration information. To make administrative changes, you must run xvm as root on a server-capable administration node.

The following are useful xvm commands related to XVM failover:

• Get information about a command:

```
# xvm help -verbose foconfig
# xvm help -verbose foswitch
# xvm help -verbose show
```

• Display information about physical volumes (*physvols*):

```
# xvm show -verbose physvol
# xvm show -verbose physvol | fgrep affinity > templatefile
```

• Push the information in the /etc/failover2.conf file to the kernel:

```
# xvm foconfig -init
```

- Switch paths for physical volumes:

  ```
  # xvm foswitch -cluster -preferred phys
  # xvm foswitch -cluster -preferred phys/physvolname
  # xvm foswitch -cluster -affinity affinityvalue phys
  # xvm foswitch -cluster -setaffinity affinityvalue phys/physvolname
  ```

- Show current paths for physical volumes not on preferred paths:

  ```
  # xvm show -verbose physvol | fgrep current | fgrep -v preferred
  ```

- Switch to the specified device number for the local node:

  ```
  # xvm foswitch -dev newdevice
  ```

For details, see the *XVM Volume Manager Administrator Guide* and the xvm(8) man page.

RAID Units and XVM Failover

For considerations regarding specific models of SGI RAID hardware, see *XVM Volume Manager Administrator Guide*.

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 installation 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, "CXFS Best Practices" on page 47
Chapter 3, "SGI RAID for CXFS Clusters" on page 95
Chapter 4, "Switch Configuration" on page 103
Chapter 5, "CXFS Licensing" on page 113
Chapter 6, "Preinstallation Steps" on page 125
Chapter 7, "Server-Capable Administration Node Installation" on page 131
Chapter 8, "Postinstallation Steps" on page 137
Chapter 9, "Initial Setup of the Cluster" on page 149

Also see the *CXFS 7 Client-Only Guide for SGI InfiniteStorage*.

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 150

- "Initial Setup" on page 152

- "Configuring a Large Cluster" on page 162

- "Testing the System" on page 163

You should also refer to the information in "Configuration Best Practices" on page 47 and you may wish to use the worksheet provided in Appendix J, "Initial Configuration Checklist" on page 543.

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

For information about licenses, see Chapter 5, "CXFS Licensing" on page 113.

Preliminary Cluster Configuration Steps

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

- "Verify the License" on page 150

- "Verify that the `chkconfig` Arguments are On" on page 150

- "Verify that the Cluster Daemons are Running" on page 151

- "Gather the Required Information" on page 151

- "Configure for `nsd` Use *(Optional)*" on page 152

During the course of configuration, you will see various information-only messages in the log files. See "Normal Messages" on page 430.

Verify the License

Verify that you have the appropriate CXFS licenses by using the `cxfslicense -d` command on server-capable administration nodes after installing the CXFS software. See "Displaying the Keys with `cxfslicense` After Installing CXFS" on page 121.

Verify that the `chkconfig` Arguments are On

Ensure that the appropriate `chkconfig` arguments are `on`. For more information, see "`chkconfig` Arguments" on page 315.

Use `chkconfig --list` to verify that `cxfs_cluster` and `cxfs` are set to `on` for the site's normal run levels. For example, if the normal run levels are 3 and 5:

```
server-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:

```
server-admin# /sbin/chkconfig cxfs_cluster on
server-admin# /sbin/chkconfig cxfs on
```

```
server-admin# /sbin/chkconfig grio2 on (if running GRIOv2)
server-admin# reboot
```

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:

```
cad
cmond
crsd
fs2d
clconfd
```

To determine which daemons are running on a server-capable administration node, enter the following:

```
server-admin# service cxfs_cluster status
fs2d is running.
cmond is running.
cad is running.
crsd is running.
server-admin# service cxfs status
clconfd is running.
```

If you have previously enabled GRIOv2, the ggd2 daemon should also be running on a server-capable administration node.

If you do not see these processes on a server-capable administration node, see "Verify that the chkconfig Arguments are On" on page 150.

For more information, see:

- "Stopping and Restarting Cluster Administration Daemons" on page 468
- "Kernel Threads" on page 475

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" on page 152), you should use

the hostname when defining the first node. (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 you 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).

Initial Setup

You can create the cluster and its components using either of the following tools, which provide similar functionality:

- "Initial Setup with the CXFS GUI" on page 153
- "Initial Setup with the cxfs_admin Command" on page 157

⚠️ **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 39 and Chapter 10, "CXFS GUI" on page 167.

To initially configure the cluster with GUI, do the following:

- "Start the GUI" on page 153
- "Set Up a New Cluster with the GUI" on page 154
- "Set Up a New CXFS Filesystem with the GUI" on page 156

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:

```
server-admin# /usr/bin/cxfsmgr
```

You can also start the GUI from your web browser. For more information, see "Starting the GUI via the Command Line" on page 168.

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 GUI.
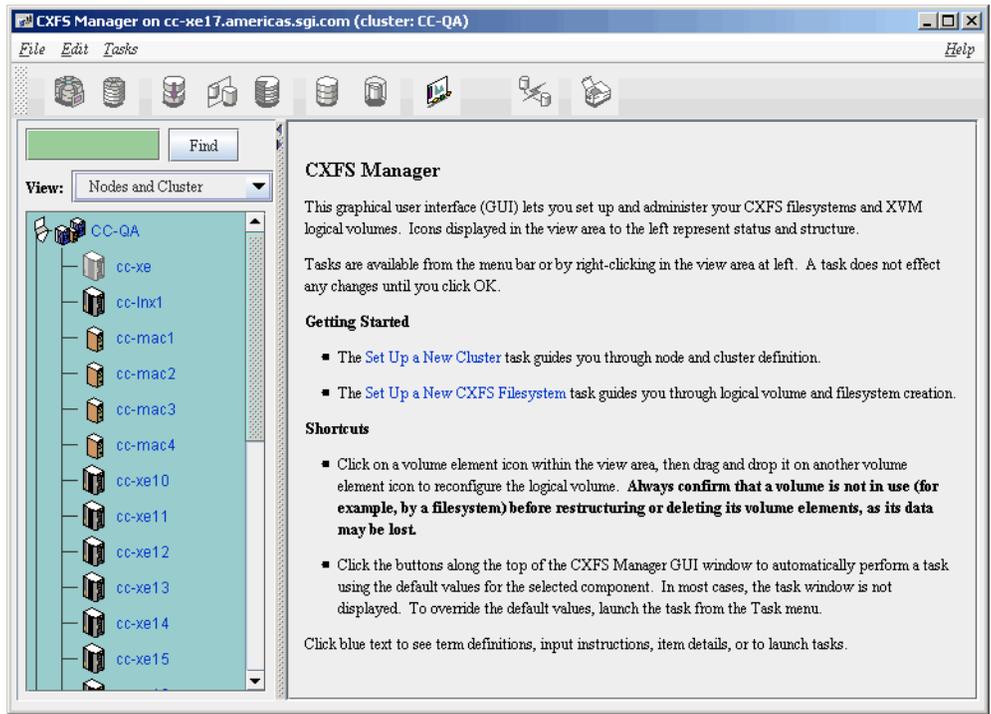
Figure 9-1 CXFS Manager

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 253.

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 188.

   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 Modification Tasks with cxfs_admin" on page 286. See "Define a Node with the GUI" on page 188.

   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 162.

3. Click **Define a Cluster** to create the cluster definition. See "Define a Cluster with the GUI" on page 203. Verify that the cluster appears in the view area. Choose **View: Nodes and Cluster**.

4. After the cluster icon appears in the view area:

   a. Click **Add/Remove Nodes in Cluster** to add the server-capable administration nodes to the new cluster. Select the server-capable administration node names and click **Add** to add them to the list and click **OK** to complete the task. See "Add or Remove Nodes in the Cluster with the GUI" on page 196.

> **Note:** SGI recommends that you form an initial cluster that consists of all of the server-capable administration nodes, to ensure that these nodes will have lower cell ID numbers than any client-only nodes. See "Create an Initial Cluster of All Server-Capable Administration Nodes" on page 54.

  b. Click **Add/Remove Nodes in Cluster** to add the client-only nodes to the new cluster. Select the client-only node names and click **Add** to add them to the list and click **OK** to complete the task. See "Add or Remove Nodes in the Cluster with the GUI" on page 196.

  > **Note:** For large clusters, SGI recommends that you build up the remainder of the cluster in small groups of client-only nodes. This makes it easier to locate and fix problems, should any occur. See "Configuring a Large Cluster" on page 162.

5. (*Optional*) Click on **Test Connectivity** to verify that the nodes are physically connected. See "Test Node Connectivity with the GUI" on page 202. (This test requires the proper configuration; see "Modifications for CXFS Connectivity Diagnostics" on page 134.)

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 206.

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 206.

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 Linux parted command to initialize the disk. For more information, see the *XVM Volume Manager Administrator Guide*.

3. Create slices, which define the physical storage, on the labeled disk. Click **Slice Disks**.

4. Create the type of volume you want (such as stripe or concat).

   Note: Mirrors do not perform well in a CXFS cluster. See "Use RAID Mirroring Not XVM Mirroring" on page 75.

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 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 219.

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 223.

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 153. For complete details about using cxfs_admin, see "CXFS Tools" on page 39 and Chapter 11, "cxfs_admin Command" on page 233.

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 150.

2. Initialize the cluster database and start cxfs_admin:

   server-admin# **/usr/cluster/bin/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:

   ```
   server-admin# service grio2 stop (if running GRIOv2)
   server-admin# service cxfs stop
   server-admin# service cxfs_cluster stop
   server-admin# service cxfs_cluster start
   server-admin# service cxfs start
   server-admin# service grio2 start (if running GRIOv2)
   ```

6. Restart cxfs_admin:

   ```
   server-admin# /usr/cluster/bin/cxfs_admin [-i clustername]
   ```

   Note: If you have multiple clusters connected to the same public network, use the -i option to identify the cluster name.

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, for IPv4:

```
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
```

   For example, for IPv6:

```
cxfs_admin:mycluster> create failover_net network=2001:db8::2:1 mask=64
cxfs_admin:mycluster > create failover_net network=fe80:: mask=64
```

   Note: For IPv4, you can use either an IP address or the number of bits for the *netmask*. For IPv6, you must supply the *netmask* as the prefix number of bits.

8. Create the switches:

```
cxfs_admin:cluster> create switch name=switchname [vendor=brocade|qlogic|voltaire|lsi] [user=username password=password]
```

   For example:

```
cxfs_admin:mycluster> create switch name=myswitch vendor=qlogic
```

9. Create another server-capable administration node:

```
cxfs_admin:mycluster> create node name=nodename os=OStype private_net=IPaddress type=server_admin
```

   For example:

```
cxfs_admin:mycluster> create node name=server2 os=Linux private_net=10.11.20.115 type=server_admin
```

   Repeat this step for each server-capable administration node.

> **Note:** SGI recommends that you form an initial cluster that consists of all of the server-capable administration nodes to ensure that these nodes will have lower cell ID numbers than any client-only nodes and thereby avoid potential problems. See "Create an Initial Cluster of All Server-Capable Administration Nodes" on page 54.

10. Create the client-only nodes:

```
cxfs_admin:mycluster> create node name=nodename [os=OStype] private_net=IPaddress [type=client_only]
```

For example:

```
cxfs_admin:mycluster> create node name=client1 os=Windows private_net=10.11.20.116
```

11. *(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
    ```

12. Obtain a shell window for one of the server-capable administration nodes in the cluster and use the Linux `parted` command to create a volume header on the disk drive. For information, see the `parted`(8) man page and *Linux Configuration and Operations Guide*.

13. Create the XVM logical volumes. In the shell window, use the `xvm` command line interface. For information, see the *XVM Volume Manager Administrator Guide*.

14. Make the XFS filesystems. In the shell window, use the `mkfs` command. For information, see:

    - *XFS for Linux Administration*

    - *XVM Volume Manager Administrator Guide*

    the .

15. 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
```

16. *(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
```

17. View the cluster status:

```
cxfs_admin:cluster> status
```

Following is a simple example using prompting mode:

```
cxfsopus14:~ # /usr/cluster/bin/cxfs_admin -s
Event at [ Oct 26 11:38:00 ]
Connecting to the local CXFS server...
cxfs_admin:(no cluster defined)> create cluster
Specify the attributes for create cluster:
 name? mycluster
Event at [ Oct 26 11:38:04 ]
cxfs_admin:mycluster> create node
Specify the attributes for create node:
 name? cxfsopus14
 type? server_admin
 private_net? 10.11.20.114
Event at [ Oct 26 11:38:08 ]
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.
Event at [ Oct 26 11:38:10 ]
cxfs_admin:mycluster> create filesystem
Specify the attributes for create filesystem:
 name? thump
 options?
 forced_unmount? false
 mountpoint? /mnt/thump
 mounted? true
```

```
 grio_managed? false
Event at [ Oct 26 11:39:04 ]
Filesystem "thump" has been created, waiting for it to be mounted on all
assigned nodes...
```

For more information, see Chapter 11, "cxfs_admin Command" on page 233 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 any
client-only nodes. By building up a large cluster with small groups of client-only
nodes (step 3), you will minimize concurrent operational issues and use the database
most efficiently.

Following is an overview of the procedure:

1. Create the initial cluster with just the server-capable administration nodes and
   test it:

   a. Define the cluster name.

   b. Define the initial server-capable administration node.

   c. *(Optional)* Create the failover networks.

   d. Create the switches.

   e. Define all of the other the server-capable administration nodes.

   f. If using the GUI, add all of the server-capable administration nodes to the
      cluster. (In cxfs_admin, nodes that are created are automatically added to
      the cluster.)

   g. Verify that the nodes are all part of the cluster membership.

2. Add the client-only nodes to the database:

   a. Define **all** client-only nodes.

   b. Add **all** client-only nodes to the cluster. (In cxfs_admin, nodes that are
      created are automatically added 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.

4. Repeat step 3 as needed to complete the cluster membership.

5. Create the CXFS filesystems.

6. *(Optional)* Manually mount the CXFS filesystems.

7. Verify that the nodes are all part of the cluster membership and that all of the filesystems are mounted and fully functional.

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:

Note: When using a script, SGI strongly recommends using `mounted=false` and then mounting the filesystems manually.

```
create cluster name=clustername
create node name=nodename  private_net=IPaddress [type=server_admin]
create failover_net network=IPaddress1 mask=netmask
create switch name=switchname [vendor=brocade|qlogic|voltaire|lsi] [user=username password=password] [copy and repeat]
create node name=nodename type=server_admin private_net=IPaddress [copy and repeat]
create node name=nodename [os=OS] private_net=IPaddress [type=client_only]    [copy and repeat]
create filesystem name=filesystemname forced_unmount=false mountpoint=/mnt/nodename mounted=false [copy and repeat]
```

Testing the System

This section discusses the following:

- "Private Network Interface" on page 164
- "System Reset Connection for Server-Capable Administration Nodes" on page 165

Private Network Interface

For each private network on each node in the pool, enter the following, where *IPaddress* is the IP address of the destination node:

- IPv4:

  # **ping -c 3 *IPaddress***

  For example:

  ```
  # ping -c3 128.162.240.71
  PING bert.americas.sgi.com (128.162.240.71): 56 data bytes
  64 bytes from 128.162.240.71: icmp_seq=0 ttl=62 time=2.181 ms
  64 bytes from 128.162.240.71: icmp_seq=1 ttl=62 time=1.327 ms
  64 bytes from 128.162.240.71: icmp_seq=2 ttl=62 time=1.189 ms

  ----bert.americas.sgi.com PING Statistics----
  3 packets transmitted, 3 packets received, 0.0% packet loss
  round-trip min/avg/max = 1.189/1.566/2.181 ms
  ```

- IPv6:

  # **ping6 -I *interface IPaddress***

  For example:

  ```
  # ping6 -I eth1 -c 3 fe80::225:90ff:fe21:3087
  PING fe80::225:90ff:fe21:3087(fe80::225:90ff:fe21:3087) from
  fe80::225:90ff:fe20:359 eth1: 56 data bytes
  64 bytes from fe80::225:90ff:fe21:3087: icmp_seq=1 ttl=64 time=0.155 ms
  64 bytes from fe80::225:90ff:fe21:3087: icmp_seq=2 ttl=64 time=0.179 ms
  64 bytes from fe80::225:90ff:fe21:3087: icmp_seq=3 ttl=64 time=0.175 ms

  --- fe80::225:90ff:fe21:3087 ping statistics ---
  3 packets transmitted, 3 received, 0% packet loss, time 1999ms
  rtt min/avg/max/mdev = 0.155/0.169/0.179/0.018 ms
  ```

If `ping` fails, follow these steps:

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

```
# 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 are powered on.

2. Start the `cxfs_admin` command as `root` in write mode (`-A`) on one of the server-capable administration nodes:

   server-admin# **/usr/cluster/bin/cxfs_admin -A [-i *clustername*]**

3. Test the connections by entering of the following for each node:

   cxfs_admin:*clustername*> **control *target*-*nodename* operation=ping**

4. If a command fails, make sure all the cables are seated properly and rerun the command.

CXFS GUI

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 Guide*.)

This chapter contains the following sections:

- "GUI Overview" on page 167
- "Guided Configuration Tasks" on page 186
- "Node Tasks with the GUI" on page 187
- "Cluster Tasks with the GUI" on page 202
- "Cluster Services Tasks with the GUI" on page 205
- "Switches and I/O Fencing Tasks with the GUI" on page 211
- "Filesystem Tasks with the GUI" on page 216
- "Privileges Tasks with the GUI" on page 227

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 errors when trying to use the CXFS Manager GUI. For more information about licensing, see Chapter 5, "CXFS Licensing" on page 113.

GUI Overview

The GUI lets you configure and administer CXFS filesystems and XVM logical volumes. It also provides icons representing status and structure.

This section discusses the following:

- "Starting the GUI via the Command Line"
- "Starting the GUI from the Web" on page 168
- "Summary of GUI Platforms" on page 169

- "Logging In" on page 170
- "Making Changes Safely" on page 170
- "GUI Windows" on page 171
- "GUI Features" on page 173
- "Key to Icons and States" on page 182

Starting the GUI via the Command Line

To start the GUI on a server-capable administration node where the CXFS GUI-client software (`sgi-sysadm_cxfs-client`) is installed, do the following:

1. Ensure that the following line is not commented out in the file `/etc/ld.so.conf`:

   ```
   /usr/lib64/sysadm/lib
   ```

2. Enter the following command line:

   ```
   linux# /usr/bin/cxfsmgr
   ```

Starting the GUI from the Web

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) for performing administrative operations:

   ```
   sgi-sysadm_xvm-web
   sgi-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. Enable and restart the required web service:

- RHEL:

```
rhel# chkconfig httpd on
rhel# service httpd restart
```

- SLES:

```
sles# chkconfig apache2 on
sles# service apache2 restart
```

4. Close all browser windows and restart the browser.

5. Enter the URL http://*server*/CXFSManager/ where *server* is the name of a server-capable administration node in the pool

6. At the resulting webpage, click the **CXFS Manager** icon.

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	A server-capable administration node system with sgi-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
Web	Any system with a web browser and Java2 1.6 or 1.6 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 227.

A dialog box will prompt 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 (xinetd must be turned on via chkconfig and running)

- **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 171.) 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 383.
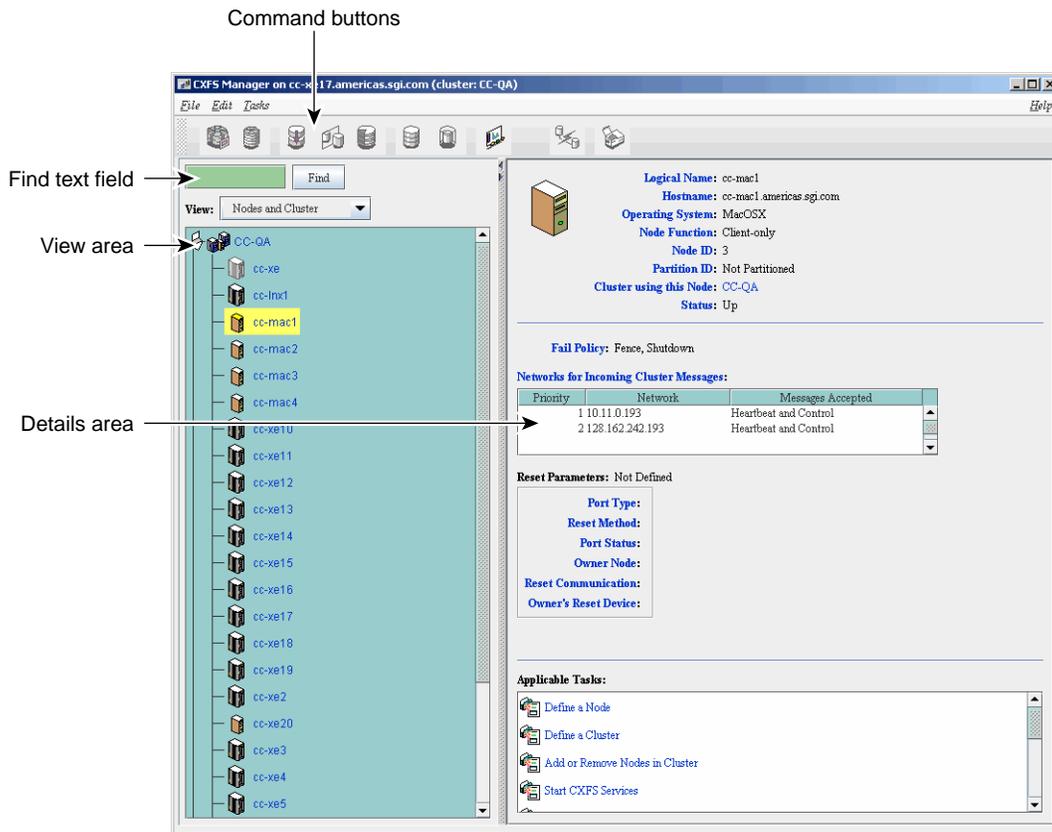
Figure 10-1 CXFS Manager GUI Showing Details for a Node

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 `cc-xe` displays a list of applicable tasks.
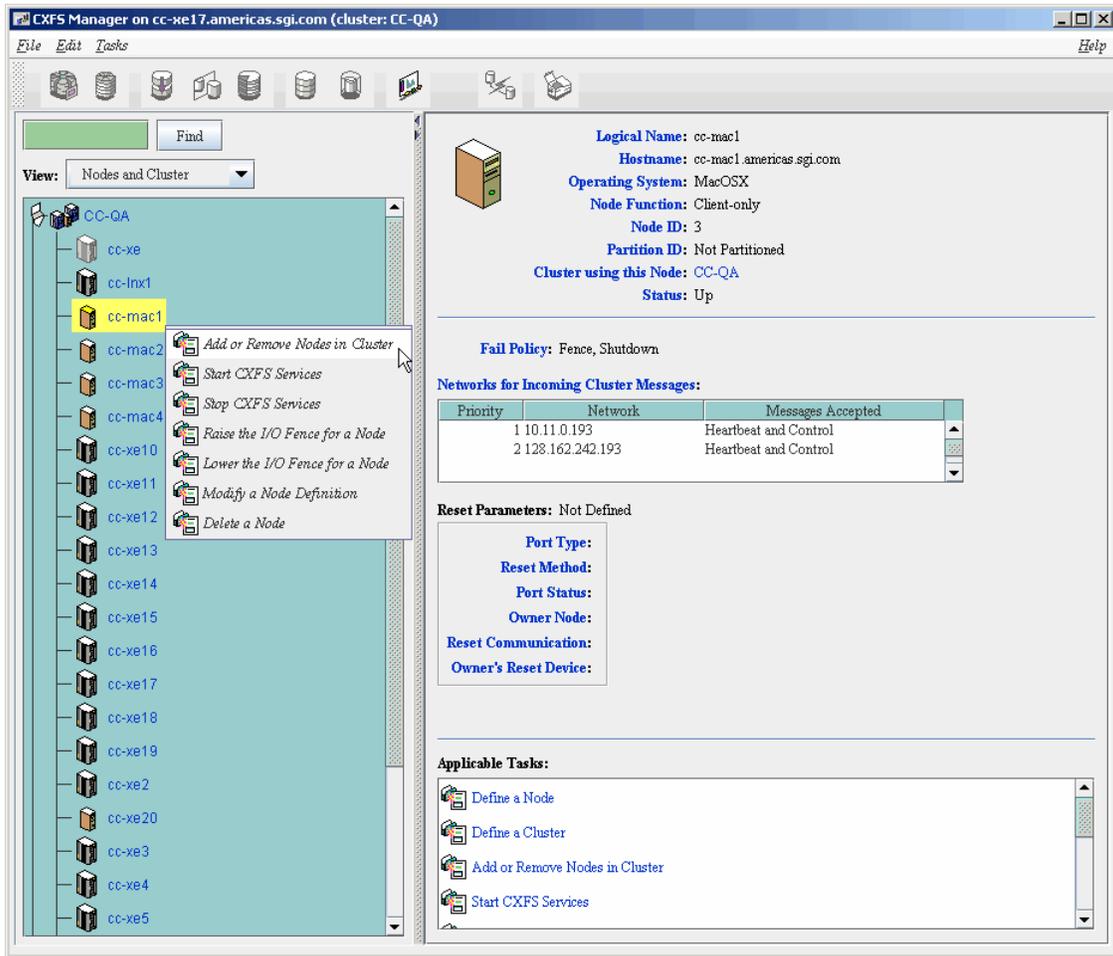
Figure 10-2 Pop-up Menu that Appears After Clicking the Right Mouse Button

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 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 174

- "File Menu" on page 174

- "Edit Menu" on page 174

- "Tasks Menu" on page 174

- "Help Menu" on page 175

- "Shortcuts Using Command Buttons" on page 175

- "View Menu" on page 177

- "Performing Tasks" on page 178

- "Using Drag-and-Drop" on page 179

- "Structuring Volume Topologies" on page 179

- "Configuring Disks" on page 180

- "Displaying State" on page 181

- "Getting More Information" on page 181

- "Important GUI and xvm Command Differences" on page 181

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 171). 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 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
	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

Button	Task
	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 177.

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 items onto shortcut command buttons.

See the *XVM Volume Manager Administrator 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 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 29.

- 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
	64–bit Linux systems
	Client-only nodes other than 64–bit Linux
	Cluster
	Expanded tree in view area
	Collapsed tree in view area

Icon	Entity
	Switch
	XVM disk
	Unlabeled disk
	Foreign disk
	Slice
	Volume
	Subvolume
	Concat
	Mirror

Icon	Entity
	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 186
- "Fix or Upgrade Cluster Nodes" on page 187

Also see "Set Up a New Cluster with the GUI" on page 154, "Set Up a New CXFS Filesystem with the GUI" on page 156, and "CXFS GUI and Status" on page 383. For information about XVM guided configuration tasks, see the *XVM Volume Manager Administrator 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 204
- "Set Up a New CXFS Filesystem with the GUI" on page 156
- "Modify a CXFS Filesystem with the GUI" on page 222
- "Define a Node with the GUI" on page 188

- "Test Node Connectivity with the GUI" on page 202

- "Add or Remove Nodes in the Cluster with the GUI" on page 196

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 206.

- Perform the necessary maintenance on the node. If required, see "Reset a Node with the GUI " on page 198.

- "Start CXFS Services with the GUI" on page 206.

- Monitor the state of the cluster components in the view area. See "CXFS GUI and Status" on page 383.

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 188

- "Examples of Defining a Node with the GUI" on page 194

- "Add or Remove Nodes in the Cluster with the GUI" on page 196

- "Reset a Node with the GUI " on page 198

- "Modify a Node Definition with the GUI" on page 198

- "Delete a Node with the GUI" on page 201

- "Test Node Connectivity with the GUI" on page 202

- "Display a Node with the GUI" on page 202

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 154.

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.

For Linux, you must specify whether the node is a server-capable administration node or a client-only node. Other nodes are always client-only nodes.

If you select a fail policy that includes reset for a Linux node, 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 a 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 Linux 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 23.

     Note: CXFS Manager will display IPv6 address in the shortest recommended form. See "IPv6: Use Any Legitimate Address Representation" on page 75.

   - **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, "cxfs_admin Command" on page 233.

6. **Node ID**: *(Optional)* 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 will calculate an ID for you.

7. **Partition ID**: *(Optional)* Uniquely defines a partition in a partitioned Altix 3000 series system or Altix 4700 system. If your system is not partitioned, leave this field empty. Use the Linux 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:

- **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 Linux 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 25. 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 71. To specify a fail policy without **Shutdown** you must define or modify the node with `cxfs_admin`. See Chapter 11, "`cxfs_admin` Command" on page 233.

- **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.

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:

    - **BMC** (SGI x86_64 systems)

    - **L2** (Any SGI ia64 system with an L2, Silicon Graphics Prism®)

  - **Reset Method**: The type of reset to be performed:

    - **Reset** simulates the pressing of the reset button on the front of the machine (recommended)

    - **Power Cycle** shuts off power to the node and then restarts it

    - **NMI** (nonmaskable interrupt) performs a core-dump of the operating system kernel, which may be useful when debugging a faulty machine

      Note: NMI depends upon kernel support, which may not be present on all SGI ia64 systems; if the kernel support is not provided, the **NMI** setting will not work.

  - **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 server-capable administration node in CXFS before you run the node connectivity diagnostics task or start CXFS services.

– **Reset Communication**: The type of reset communication used by the owner node:

- **ipmi** (required for BMC)

- **network** (optional for L2)

- **tty** (optional for L2)

– **Reset Device**: The name of the reset device file or network address used by the owner node:

- For BMC and network L2 devices, this must be the IP address or hostname of the device. An ethernet cable must connect the device to the private network.

- For TTY L2 devices, this must be a pathname that begins with the `/dev` prefix. `/dev/ttyd2` is the most commonly used port, except on 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.) A serial cable must connect the owner node's serial port to this node's system controller port so that the owner node can reset this node.

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 find the partition ID by reading the `proc` file. For example:

```
linux# 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.

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.
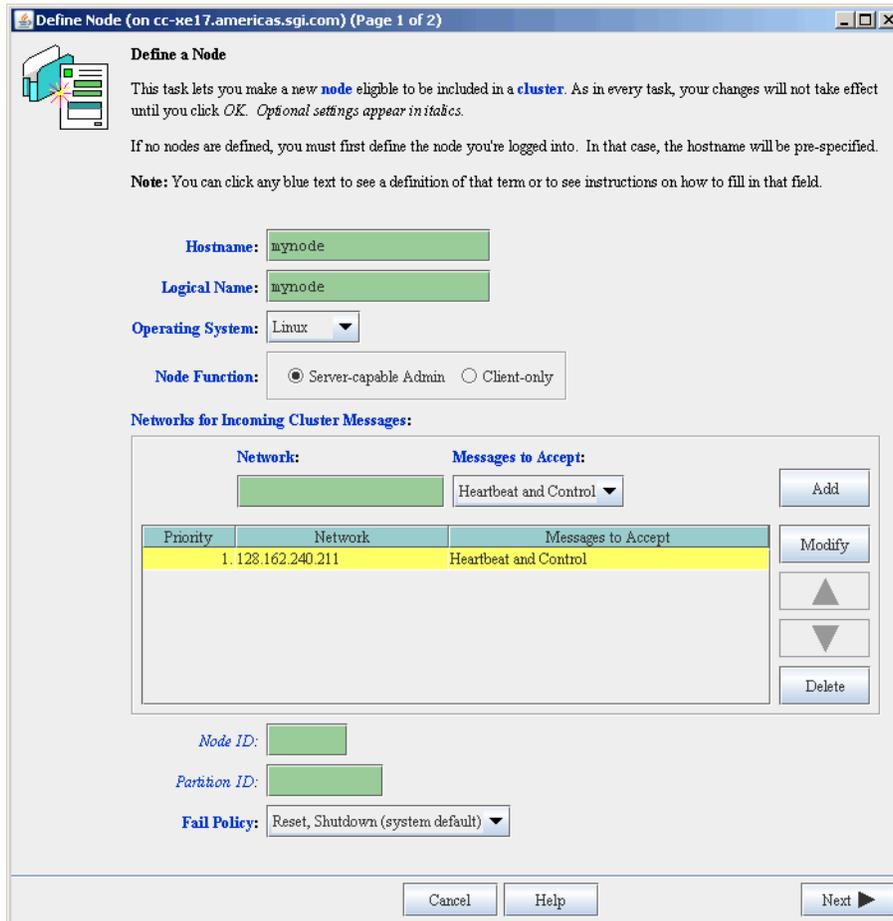
Figure 10-3 Example Node Definition

Figure 10-4 Example System Reset Settings

Add or Remove Nodes in the Cluster with the GUI

To avoid potential problems, you should add all server-capable administration nodes to the cluster before adding any client-only nodes. See "Create an Initial Cluster of All Server-Capable Administration Nodes" on page 54. After you have added nodes to the pool and defined the cluster, you can indicate which nodes to include in the cluster.

> **Note:** Before adding or removing a server-capable administration node, you must first unmount any filesystems for which the node will be a potential metadata server.
>
> 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**.

     > **Note:** SGI recommends that you form an initial cluster that consists of all of the server-capable administration nodes, to ensure that these nodes will have lower cell ID numbers than any client-only nodes. See "Create an Initial Cluster of All Server-Capable Administration Nodes" on page 54.

   - 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 Linux nodes with system controllers. 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, "`cxfs_admin` Command" on page 233.

Click **Next** to move to the next page.

3. **Partition ID**: *(Optional)* Uniquely defines a partition in a partitioned Altix 3000 series system or Altix 4700 system. If your system is not partitioned, leave this field empty. You can use the Linux `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:

- **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 Linux 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 25. 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 71. To specify a fail policy without **Shutdown** you must define or modify the node with `cxfs_admin`. See Chapter 11, "`cxfs_admin` Command" on page 233.

- **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.

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:

       - **BMC** (SGI x86_64 systems

       - **L2** (Any SGI ia64 system with an L2, Silicon Graphics Prism)

     – **Reset Method**: The type of reset to be performed:

       - **Reset** simulates the pressing of the reset button on the front of the machine (recommended)

       - **Power Cycle** shuts off power to the node and then restarts it

       - **NMI** (nonmaskable interrupt) performs a core-dump of the operating system kernel, which may be useful when debugging a faulty machine

         > **Note:** NMI depends upon kernel support, which may not be present on all SGI ia64 systems; if the kernel support is not provided, the **NMI** setting will not work.

     – **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. 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 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.

6. Click **OK**.

On Linux, you can find the partition ID by reading the `proc` file. For example:

```
linux# 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.

You could enter one of the following for the **Partition ID** field:

```
1
001.10
```

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 204.

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 203
- "Modify a Cluster Definition with the GUI" on page 204
- "Delete a Cluster with the GUI" on page 205

- "Display a Cluster with the GUI" on page 205

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 154.

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 203, for information about `Experimental` mode.

  - **Notify Administrator**: Select the desired notification. For more information, see "Define a Cluster with the GUI" on page 203.

  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 196.

> **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 417.

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 196.

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 206

- "Stop CXFS Services with the GUI" on page 206

- "Set Tiebreaker Node with the GUI" on page 207

- "Set Log Configuration with the GUI" on page 207

- "Revoke Membership of the Local Node with the GUI" on page 210

- "Allow Membership of the Local Node with the GUI" on page 210

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 337

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.

⚠️ **Caution:** If one of the server-capable administration nodes is the CXFS tiebreaker in a cluster with two server-capable administration nodes, 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 382.

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 331
   "`cad.options` on Server-Capable Administration Nodes" on page 138
   "`fs2d.options` on Server-Capable Administration Nodes" on page 139

   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 325).

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.

The 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 325.

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 210.

- 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 211
- "Modify a Switch Definition with the GUI" on page 214
- "Update Switch Port Information with the GUI" on page 214
- "Delete a Switch Definition with the GUI" on page 215
- "Raise the I/O Fence for a Node with the GUI" on page 215
- "Lower the I/O Fence for a Node with the GUI" on page 215

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`.

> **Note:** The telnet protocol is used by default. To use the ssh protocol for a Brocade or InfiniBand switch, you must use cxfs_admin. See "Create or Modify a Switch with cxfs_admin" on page 288.

- **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 SAN 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 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 211.

2. Click **OK** to complete the task.

Note: The telnet protocol is used by default. To use the ssh protocol for a Brocade or InfiniBand switch, you must use cxfs_admin. See "Create or Modify a Switch with cxfs_admin" on page 288.

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, SAS, or InfiniBand 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 (or the `ssh` protocol if set via `cxfs_admin` for Brocade switches) 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 156.

This section discusses the following:

- "Make Filesystems with the GUI" on page 216
- "Grow a Filesystem with the GUI" on page 218
- "Define CXFS Filesystems with the GUI" on page 219
- "Modify a CXFS Filesystem with the GUI" on page 222
- "Mount CXFS Filesystems with the GUI" on page 223
- "Unmount CXFS Filesystems with the GUI" on page 224
- "Mount a Filesystem Locally" on page 224
- "Unmount a Local Filesystem" on page 225
- "Delete a CXFS Filesystem with the GUI" on page 225
- "Remove Filesystem Mount Information" on page 225
- "Relocate a Metadata Server for a CXFS Filesystem with the GUI" on page 226

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. If the volume contains an `rt` subvolume, the real-time size will be set to the size of the `rt` 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**: *(Obsolete)*

   - **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. 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.

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 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.

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 for the initial mount of a filesystem.

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 153.

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.

     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.

   - **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 224.

- **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. See "Relocation" on page 26.

  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 216 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 225.

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 for the initial mount of a filesystem.

- **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 206.

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 224.

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 26). 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 206.

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 Users" on page 227
- "Revoke Task Access from Users" on page 230

Grant Task Access to 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 `abuild` can now execute. This screen is displayed when you select **View: Users** and click `abuild` to display information in the details area of the GUI window. The privileged commands listed are the underlying commands executed by the GUI tasks.
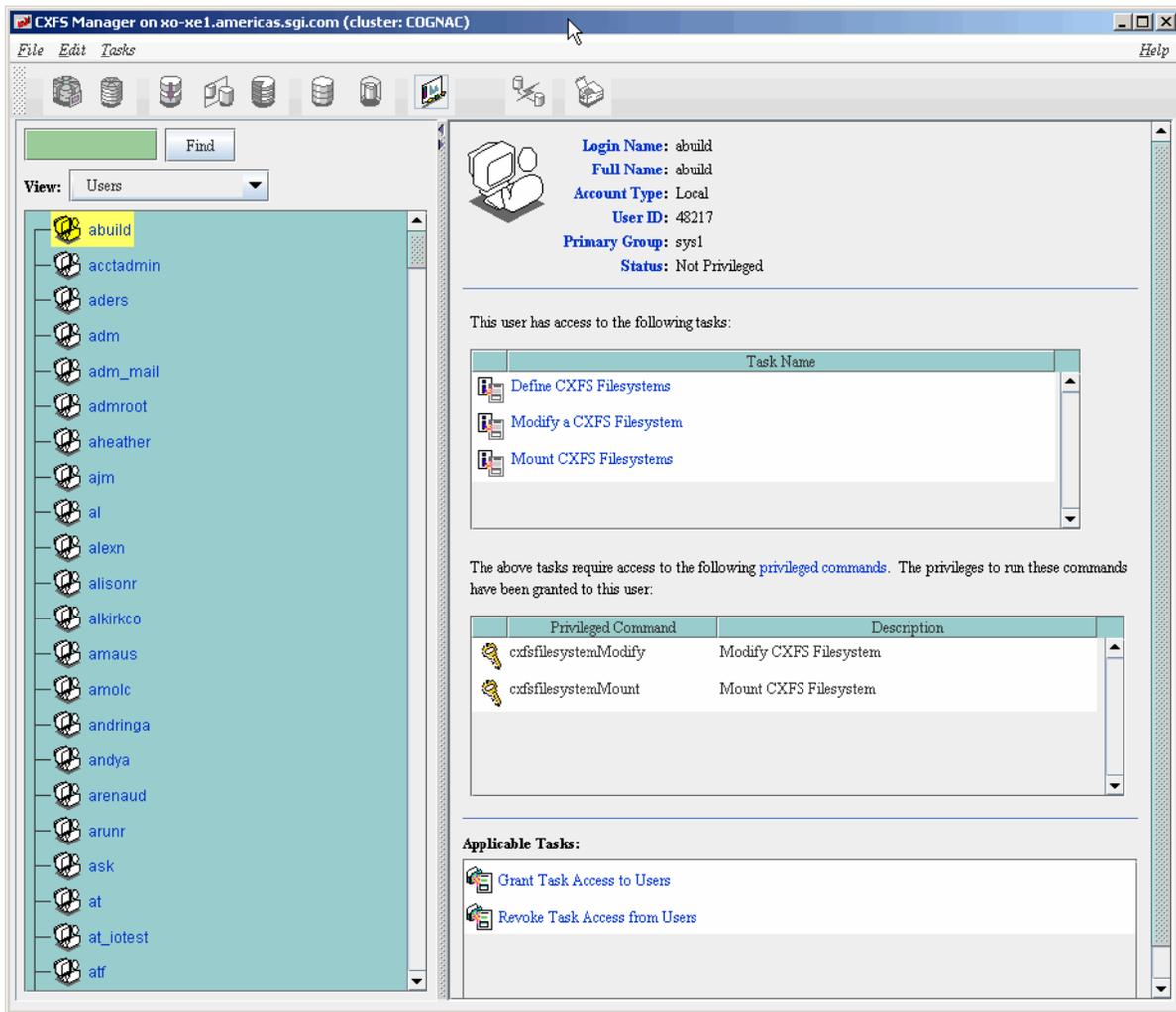
Figure 10-6 Task Privileges for a Specific User

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 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.

cxfs_admin **Command**

This chapter discusses the following:

- "cxfs_admin Overview" on page 233
- "Node Tasks with cxfs_admin" on page 250
- "Cluster Tasks with cxfs_admin" on page 270
- "CXFS Filesystem Tasks with cxfs_admin" on page 276
- "Network Failover Modification Tasks with cxfs_admin" on page 286
- "Switch Tasks with cxfs_admin" on page 287
- "Fencing Tasks with cxfs_admin" on page 293
- "Saving and Recreating the Current Configuration with cxfs_admin" on page 295

cxfs_admin **Overview**

To use the cxfs_admin command, you must be logged in to a node that has permission to access the CXFS cluster database. See "Setting cxfs_admin Access Permissions" on page 241.

To make configuration changes using cxfs_admin, you must be logged in as root and execute cxfs_admin on a node with admin permission and in write mode. To perform some operational tasks, you must in addition execute cxfs_admin directly on a server-capable administration node. See "Making Changes Safely" on page 240

Note: For the steps to create a cluster for the first time, see "Initial Setup with the cxfs_admin Command" on page 157.

This section discusses the following:

- "Starting cxfs_admin" on page 234
- "Syntax of Commands within cxfs_admin" on page 235
- "Setting cxfs_admin Defaults" on page 238

- "Making Changes Safely" on page 240

- "Setting `cxfs_admin` Access Permissions" on page 241

- "Accessing the Correct Cluster at a Multiple-Cluster Site" on page 243

- "Getting Help" on page 243

- "Basic and Advanced Mode" on page 245

- "Using Prompting Mode" on page 247

- "Displaying Command History" on page 248

- "Waiting for Commands to Complete" on page 248

- "Entering `cxfs_admin` Commands on the Command Line" on page 249

- "Using `cxfs_admin` Script Files" on page 249

- "Exiting from `cxfs_admin`" on page 250

Starting `cxfs_admin`

To execute `cxfs_admin` in the default read-only mode, enter the following:

/usr/cluster/bin/cxfs_admin [-i *clustername*]

To execute `cxfs_admin` in write mode, enter the following:

/usr/cluster/bin/cxfs_admin -A [-i *clustername*]

Note: If you have multiple clusters connected to the same public network, use the `-i` option to identify the cluster name.

If no other user already holds the `cxfs_admin` lock, you will be given the lock by default when executing `cxfs_admin` in write mode. You can make executing `cxfs_admin` in write mode the default by setting the `write_mode` parameter in the `$HOME/.cxfs_admin` file; see "Setting `cxfs_admin` Defaults" on page 238.

See also:

- "Making Changes Safely" on page 240

- "Entering `cxfs_admin` Commands on the Command Line" on page 249

- "Setting cxfs_admin Access Permissions" on page 241

- "Accessing the Correct Cluster at a Multiple-Cluster Site" on page 243

Syntax of Commands within `cxfs_admin`

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 command syntax is:

command [[class:]object] [attributes]

where *class* can be one of the following:

```
autoconf
cluster
cmsd
fence
failover_net
filesystem
log
node
poolnode
switch
```

where *attributes* takes a number of formats depending on the context, such as the following:

attribute
attribute=value
attribute=value1,value2,value3...

The actual syntax components for any given command vary, 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 with the `show` and `config` commands in place of the *object* to apply the command to the entire cluster. if you do not specify any attributes, you can even omit the `*` character.

For commands that will change the cluster database, however, SGI recommends that you also include at least one other character in the object's name to ensure that you act upon only the intended objects. For example, if you wanted to unmount both `myfileA` and `yourfileB`, you could enter `unmount *file*`.

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")
```

```
grio_managed=    : 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 the complexity mode; see "Basic and Advanced Mode" on page 245.

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>
  Linux     MacOSX    Unknown    Windows
```

Use $ to refer to the object in the last command that you entered at the cxfs_admin prompt. 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 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.

Some attributes can take a comma-separated list of values. These attributes also allow you to use the + and – symbols to specify items to add or remove. The symbol must precede the list. For example, for the nodes attribute to the modify command:

- nodes=*entirelist* to specify the entire list of nodes that can mount the filesystem, given as a comma-separated list; you must include the server-capable administration nodes in this list.

- nodes=+*additionalnodes* to specify additional (+) nodes that can mount the filesystem, which will be added to the current list.

- nodes=–*removednodes* to specify which nodes that were previously allowed to mount the filesystem but should now be prevented (–) from mounting the filesystem.

> **Note:** The `+`/`-` symbols are not applicable in prompting mode.

For an example, see "Create or Modify a CXFS Filesystem with `cxfs_admin`" on page 277.

Setting `cxfs_admin` Defaults

You can use one of the following methods to set the defaults for the way that `cxfs_admin` behaves, shown in the order of their 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 prompting and output will show only those values that are required (`basic`) or all values (`advanced`). The default is `basic`. See "Basic and Advanced Mode" on page 245.

   - `stop_on_error` will abort a command upon encountering an error (`true`) or keep going (`false`). The default is `true`.

2. Set the environment variables discussed in Table 11-1.

Table 11-1 `cxfs_admin` 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 connected to the same public network, or want to use cxfs_admin from a node that is not part of the cluster. 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.
CXFS_ADMIN_WRITE_MODE	true or false (default), which specifies whether cxfs_admin will attempt to obtain the lock (enter write mode) by default (similar to entering -A on the cxfs_admin command line). To obtain the lock, the host on which cxfs_admin is executed must also have admin access permission and you must be logged in as root. (See "Setting cxfs_admin Access Permissions" on page 241.)

3. Use the `.cxfs_admin` file in your home directory (as defined by the HOME environment variable) to set any of the following:

```
cluster_name=clustername
editor=emacs|vi
line_wrap=true|false
mode=basic|advanced
stop_on_error=true|false
write_mode=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 by default, include the following in your (or `root`) `$HOME/.cxfs_admin` file:

```
# My settings for cxfs_admin:
cluster=mycluster
mode=advanced
editor=vi
```

Making Changes Safely

This section discusses the following safety features of `cxfs_admin`:

- "Read-Only Mode by Default" on page 240
- "Making Changes Requires Both Write Mode and `admin` Permission" on page 240
- "Stealing the Lock" on page 241
- "Giving Up the Lock" on page 241

Read-Only Mode by Default

By default, `cxfs_admin` executes in read-only mode, in which the `cxfs_admin` lock is not set. The lock permits only one user at a time can make changes via `cxfs_admin`.

⚠ **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 `cxfs_admin` command at any given time.

In read-only mode, you can view the cluster status but you cannot make changes. The `cxfs_admin` prompt identifies that you are in read-only mode.

Making Changes Requires Both Write Mode and `admin` Permission

To make changes to the cluster database, you must be logged in as `root` to a host that has `admin` access permission (see "Setting `cxfs_admin` Access Permissions" on page 241) and be in write mode. Do one of the following as `root` when `cxfs_admin` is connected to a node with `admin` access:

- Use the `-A` option on the `cxfs_admin` command line to override the default setting for this instance of the `cxfs_admin` process.

- Use the `.cxfs_admin` file or the `CXFS_ADMIN_WRITE_MODE` environment variable to permanently override the default setting (making it unnecessary to enter `-A` on the command line). See "Setting `cxfs_admin` Defaults" on page 238.

- Invoke `cxfs_admin` as normal but then request the lock:

  ```
  cxfs_admin:mycluster> lock
  ```

Stealing the Lock

If you request the lock but it is already held by another user, you will enter in read-only mode. 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
Event at [ Dec 21 15:58:02 ]
The administration lock is already held by root@node2 (pid=48449)
cxfs_admin:mycluster (read only) > lock steal=true
Event at [ Dec 21 15:58:03 ]
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, you can obtain the lock by entering `lock` (there is no need to steal the lock at this point).

Giving Up the Lock

To give up the lock, use the `unlock` command. For example:

```
cxfs_admin:mycluster> unlock
Event at [ Dec 21 15:58:02 ]
cxfs_admin:mycluster (read only) >
```

Setting `cxfs_admin` Access Permissions

The `access` command lets you 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
Event at [ Dec 21 15:58:02 ]
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 263)

  - `any` denotes any system that is on the private network

- permission specifies:

  - admin provides the ability to view and change the cluster database and perform administrative actions (if cxfs_admin was started in write mode and you are logged in as root). This is the default for all server-capable administration nodes.

  - monitor provides the ability only to view the current status of the cluster and the current definitions in the cluster database. This is the default for all client-only nodes.

- 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 connected to the same public network, use the -i option to identify the cluster name:

cxfs_admin -i *clustername*

For example, for the cluster named mycluster:

server-admin# **/usr/cluster/bin/cxfs_admin -i mycluster**

On each client-only node, add -i *clustername* to the cxfs_client.options file. For more information about the cxfs_client.options file, see *CXFS 7 Client-Only Guide for SGI InfiniteStorage.*

You can also use the CXFS_ADMIN_CLUSTER_NAME environment variable. See "Setting cxfs_admin Defaults" on page 238.

Note: CXFS does not support multiple clusters on the same **private** network.

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` (operations) command:

ops *object*

For example:

```
cxfs_admin:mycluster> ops filesystem
Event at [ Dec 21 15:58:02 ]
Commands for "filesystem":
    create, ops, show
cxfs_admin:mycluster> ops myfs
Event at [ Dec 21 15:58:03 ]
Commands for "filesystem:myfs":
    delete, modify, mount, ops, show, unmount
```

You can also use the `clconf_info` tool to view status. See Chapter 14, "Monitoring Status" on page 381.

For help with error messages, see "`cxfs_admin` Errors" on page 457.

Basic and Advanced Mode

The cxfs_admin command operates in two complexity modes:

- **Basic mode**, which:

    - Shows only the required options and attributes in show output

    - Provides a list of possible choices when using the <TAB> key

    - Uses prompting for only the required fields

- **Advanced mode**, which:

    - Provides a list of possible choices when using the <TAB> key

    - Prompts for all possible fields, displays all attributes

    - Includes debugging information in output

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.

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-mode commands and attributes are noted in their help topics.

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 for the node named cxfsxe5:

```
cxfs_admin:clusterOne> set mode=basic
Event at [ Dec 20 12:32:44 ]
Mode is set to basic
cxfs_admin:clusterOne> show cxfsxe5
```

```
Event at [ Dec 20 12:32:54 ]
node:cxfsxe5:
    cellid=1
    enabled=true
    os=Linux
    private_net:
        10.0.199.1
    status:
        age=23
        connected=true
        fencing=Stable
        license:
            have_license=true
        summary=Stable
        version=7.0.0.3
        wwns:
            210000e08b123d95
    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 275.

For example, the following output shows all information that is available in advanced mode for `cxfsxe5`:

```
cxfs_admin:clusterOne > set mode=advanced
Event at [ Dec 20 12:33:48 ]
Mode is set to advanced
cxfs_admin:clusterOne> show cxfsxe5
Event at [ Dec 20 12:33:53 ]
node:cxfsxe5:
    autoconf=false
    cellid=1
    clustername=clusterOne
    enabled=true
    failpolicy=Fence,Shutdown
    hostname=cxfsxe5.americas.sgi.com
    nodeid=1
    os=Linux
```

```
private_net:
    10.0.199.1
status:
    age=23
    build=07:39:08 Sep 22 2009
    connected=true
    fencing=Stable
    license:
        have_license=true
    member=true
    stable=true
    summary=Stable
    version=7.0.0.3
    wwns:
        210000e08b123d95
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:clusterOne> create <RETURN>
What do you want to create? The following can be used:
    autoconf, cluster, failover_net, filesystem, node, switch
```

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 245.

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 `type` value is `client-only`, `cxfs_admin` will prompt you for the operating system.

To exit from prompt mode, send an interrupt signal (typically, press `Ctrl-C`).

Displaying 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 (full pathname or relative pathname):

  `history output=`*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 -c option, as follows:

```
server-admin# /usr/cluster/bin/cxfs_admin -c "cxfs_admin_commands"  [-i clustername]
```

Using `cxfs_admin` Script Files

You can execute a series of cxfs_admin commands by using the -f option and specifying an input file:

```
server-admin# /usr/cluster/bin/cxfs_admin -f command_file  [-i clustername]
```

For example, suppose the file /tmp/showme contains the following:

```
cxfs6# more /tmp/showme
show access
set
```

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 CXFS server for the "clusterOne" cluster...
Event at [ Dec 20 12:49:17 ]
access:
    admin=server
    monitor=cluster
Event at [ Dec 20 12:49:17 ]
mode=basic
editor=emacs
line_wrap=false
stop_on_error=true
```

Note: To make changes, you must execute cxfs_admin in write mode, such as by using the -A option. See "Making Changes Requires Both Write Mode and admin Permission" on page 240.

Exiting from `cxfs_admin`

To exit from prompt mode, send 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:

- "Automatically Configure a Client-Only Node" on page 250
- "Create or Modify a Node with `cxfs_admin`" on page 253
- "Delete a Node with `cxfs_admin`" on page 262
- "Enable a Node with `cxfs_admin`" on page 263
- "Disable a Node with `cxfs_admin`" on page 263
- "Show Node Information with `cxfs_admin`" on page 264
- "Enable or Disable CXFS Kernel Membership (`cmsd`) for the Local Node" on page 267
- "Control and Contact a Node with `cxfs_admin`" on page 267
- "Move a Node Between the Cluster and the Pool with `cxfs_admin`" on page 269

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.

Automatically Configure a Client-Only Node

The `autoconf` class creates a rule that determines whether the specified new client-only nodes can be automatically configured into the CXFS cluster database. This is known as *easy client configuration*.

Note: The `autoconf` creation action only applies to hosts that are not currently defined in the cluster database.

Before deleting a node created by `autoconf`, you must set the `autoconf policy` to `disallowed`. If the policy remains in `allowed` mode, the node will be automatically re-created after the node is deleted.

You can specify a single hostname and/or a range of IP addresses to be allowed (*whitelisted*) or disallowed (*blacklisted*). Only those nodes specified in a whitelist and not specified in a blacklist can be automatically configured. (That is, the blacklist has higher priority than the whitelist.)

Each `autoconf` rule must have a unique name. By default, only nodes that are defined by an `autoconf` rule set to `enable_node=true` will be defined as enabled and will be allowed to join the cluster. (If you set `enable_node=false`, the node will be defined in the database but disabled/inactive.)

- To create a new rule (line breaks shown here for readability):

  ```
  create autoconf
       rule_name=rulename
       policy=allowed|disallowed                  (allowed)
       hostname=hostname
       startIP=startingIPaddress
       endIP=endingIPaddress
       enable_node=true|false                     (true)
  ```

- To change an existing rule:

  ```
  modify [allowed:|disallowed:]rulename
  ```

- To delete an existing rule:

  ```
  delete [allowed:|disallowed:]rulename
  ```

- To show an existing rule:

  ```
  show [allowed:|disallowed:]rulename
  ```

- To show all existing rules:

  ```
  show autoconf
  ```

- To show all allowed rules, which displays the rules for client-only nodes that are allowed to be automatically configured (whitelisted):

  ```
  show allowed
  ```

- To show all disallowed rules, which displays the rules for client-only nodes that are explicitly not allowed to be automatically configured (blacklisted):

  ```
  show disallowed
  ```

For example, suppose you want nodes NodeA and NodeB to be automatically configured and enabled in the cluster. You would create two rules:

```
cxfs_admin:mycluster> create autoconf rule_name=ruleNodeA policy=allowed hostname=NodeA enable_node=true
cxfs_admin:mycluster> create autoconf rule_name=ruleNodeB policy=allowed hostname=NodeB enable_node=true
```

If you then wanted to change the rule so that NodeA could not be automatically configured, you would modify the autoconf rule named ruleNodeA:

```
cxfs_admin:mycluster> modify autoconf rule_name=ruleNodeA policy=disallowed hostname=NodeA
```

Note: If NodeA was already defined in the cluster database, this would not affect the existence of that node but would prevent it from being recreated automatically in the future.

In this case, you could also delete ruleNodeA, because not listing a node in an autoconf allow rule is the same as disallowing it.

Suppose all of your client-only nodes have IP addresses within the range 192.168.0.1 through 192.168.0.20. To let all of the nodes be configured automatically except for the node named bert, you would need to create two rules:

```
cxfs_admin:mycluster> create autoconf rule_name=allowall
policy=allowed startIP=192.168.0.1 endIP=192.168.0.20 enable_node=true
cxfs_admin:mycluster> create autoconf rule_name=rulebert policy=disallowed
```

If there is a node that is not within the IP address range that you also want to allow, you can include it in the same rule. For example, the following would allow all nodes in the range 192.168.0.1 through 192.168.0.20 plus the node named ernie (that is in a different IP range):

```
cxfs_admin:mycluster> create autoconf rule_name=allowall policy=allowed startIP=192.168.0.1
endIP=192.168.0.20 hostname=ernie enable_node=true
Event at [ Dec 20 12:53:50 ]
```

```
"allowall" is created. After creating all desired autoconf rules, unlock all cxfs_admin sessions to allow
 the defined nodes to then be automatically configured.
```

After you have finished creating or modifying all of the desired autoconf rules, you must unlock all cxfs_admin sessions in order for nodes to be automatically configured. (The automatic configuration process must have access to the cxfs_admin lock.) If the node that is created or reconfigured via autoconf is unable to join cluster membership, you must reboot the node.

When you create a new node, you can use the failover_net attribute to specify the preferred selection order of the private IP addresses (private_net), which is particularly important in an IPv6 environment. See:

- "Create or Modify a Node with cxfs_admin" on page 253

- "Network Failover Modification Tasks with cxfs_admin" on page 286

Create or Modify a Node with `cxfs_admin`

Note: Before adding a server-capable administration node, you must first unmount any filesystems for which the node will be a potential metadata server.

You should create all server-capable administration nodes before creating any client-only nodes to ensure that the server-capable administration nodes get the lowest cell IDs. The cell ID is assigned by CXFS according to the order in which the nodes are created.

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=Linux|MacOSX|Windows|Unknown                        (Linux)
     private_net=private_network_IPaddress_list|hostname_list

   Advanced-mode:
     enabled=true|false                                     (true)
     hostname=hostname                                      (FQDN_of_nodename)
     nodeid=nodeID                                          (assigned by cxfs_admin)
     partition_id=partition_number
```

```
     failpolicy=FenceReset,Fence,Reset,Shutdown              (Fence,Shutdown)
          reset_method=nmi|powerCycle|reset                  (reset)
          reset_port= bmc|l2
          reset_password=password                            (password)
          reset_user=username                                (admin)
          reset_status=enabled|disabled                       (enabled)
          reset_node=server_admin_node_sending_reset_command
          reset_comms=ipmi|network|tty
          reset_device=port|IP_address_or_hostname_of_device
```

You will use a similar set of subcommands to modify a node via the `modify` command. (To modify a node, you must first disable it.)

When adding the first server-capable administration node, you must restart it or restart CXFS services and cluster services on the node:

```
server-admin# service cxfs stop
server-admin# service cxfs_cluster stop

server-admin# service cxfs_cluster start
server-admin# service cxfs start
```

When you create additional nodes, they will automatically be enabled and join the cluster.

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>
 Linux     MacOSX     Unknown     Windows
 os? macosx
 private_net? 192.168.0.178
Event at [ Dec 21 15:58:02 ]
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
```

> 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
Event at [ Dec 21 15:58:02 ]
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? bmc
 reset_password?
 reset_user?
 reset_status? enabled
 reset_node? mds2
 reset_comms? ipmi
 reset_device? newnode-bmc.mycompany.com
Event at [ Dec 21 15:58:02 ]
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.
```

You will use a similar set of subcommands to modify a node via the `modify` command. (To modify a node, you must first disable it.)

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.

  - `server_admin` is a `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:

  ```
  Linux
  MacOSX
  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 23.

  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. Unless you specify the `failover_net` attribute, the interfaces must be specified in the same order according to subnet on each node in the cluster.

  To limit the networks chosen from this list or reprioritize the list, use the `failover_net` attribute. See "Network Failover Modification Tasks with `cxfs_admin`" on page 286.

For more information about using the hostname, see "Hostname Resolution and Network Configuration Rules" on page 125.

You can use `private_net` for networks using both IPv6 and IPv4. For IPv6, the address must be a link-local address. For example:

```
cxfs_admin:mycluster> create node name=mynode private_net=192.168.0.5,fe80::204:23ff:fee2:d861
```

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 263.

- `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 25.

    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 "Node Shutdown" on page 62.

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 25.

For example, to perform a reset only if a fencing action fails, specify the following:

```
failpolicy=Fence,Reset
```

Note: 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`(8) man page.

To perform a fence and an asynchronous reset, specify the following:

```
failpolicy=FenceReset
```

- `hostname` is the fully qualified domain name. 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 an Altix 3000 series system or
  Altix 4700 system. For a non-partitioned system, this attribute is not required (the
  default unassigned). To unset the partition ID, use a value of `0`. You can find the
  partition ID by reading the `proc` file. For example, for an Altix 3000:

  ```
  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` specifies the action that will automatically be taken by CXFS if `reset_status=enabled`. It can be one of the following:

  - `powerCycle` shuts off power to the node and then restarts it

    Note: If you specify `reset_comms=ipmi`, `reset_method=powerCycle` and `reset_method=reset` have the same effect, which is to perform a hard reset (that is, perform a `power reset` with `ipmitool`).

  - `reset` simulates the pressing of the reset button on the front of the machine (recommended)

  - `nmi` (nonmaskable interrupt) performs a core-dump of the operating system kernel, which may be useful when debugging a faulty machine

    Note: NMI depends upon kernel support, which may not be present on all SGI ia64 systems; if the kernel support is not provided, the `nmi` setting will not work.

    NMI is not available on systems containing a baseboard management controller (BMC), such as SGI x86_64 systems.

  The default is `reset`.

- `reset_port` is the system controller port type based on the node hardware, as shown in Table 11-2 on page 262.

- `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. The default is `password`.

- `reset_user` is the user name for the node's system controller port, which might not apply to all systems. If you wish to set or change the system controller user, consult the hardware manual for your node. The default is `admin`.

- `reset_status` specifies if the automatic system reset capability is turned on (`enabled`) or turned off (`disabled`). Using `disabled` lets you 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`, the default for nodes without system controllers is `disabled`; see

"Hardware and Software Requirements for Server-Capable Administration Nodes" on page 34.

- `reset_node` specifies the name of the server-capable administration node (the *owner node*) that will send the reset command to the target node. It can be the logical name, hostname, or fully qualified domain name. If you use `reset_comms=tty`, serial cables must physically connect the target node and the owner node through the system controller port. The owner 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 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`

Altix 3000 Bx2 system:

`reset_comms=network reset_device=nodename-l2.mycompany.com`

Altix 350 system without an L2:

`reset_comms=tty reset_device=/dev/ttyIOC0`

SGI x86_64 system with a BMC:

`reset_comms=ipmi reset_device=nodename-bmc.mycompany.com`

Table 11-2 System Controller Types

bmc	l2
Any SGI x86_64 system	Any SGI ia64 system with an L2 Prism

The following is an example of a node created for BMC reset, where the default for `reset_password` is `password` as explained in Appendix C, "BMC System Controller" on page 487 (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=reset reset_port=bmc reset_status=enabled
reset_node=mds2 reset_comms=ipmi reset_device=node1-bmc.mycompany.com
```

Delete a Node with `cxfs_admin`

Note: Before deleting a node that was created by `autoconf`, you must set the `autoconf policy` to `disallowed`. If the policy remains in `allowed` mode, the node will be automatically re-created after the node is deleted.

Before deleting a server-capable administration node, you must first unmount any filesystems on that node and disable the node.

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
Node "mynode" has been disabled, waiting for it to leave the cluster...
Waiting for node mynode, current status: Disabled and unmounting
Operation completed successfully
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 240.

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 strongly recommended for stopping CXFS services on CXFS server-capable administration nodes or on a client-only tie-breaker node that must be shutdown for maintenance or otherwise taken offline. These types of nodes will affect the CXFS kernel membership quorum calculation as long as they are configured as enabled in the cluster database.

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 263.

Show Node Information with `cxfs_admin`

You can display a node's parameters with the following command:

show [node:]*nodename*

For example, if `cxfsxe5` is a unique name in the cluster database:

```
cxfs_admin:clusterOne> show cxfsxe5
Event at [ Dec 21 13:35:25 ]
node:cxfsxe5:
    cellid=1
    enabled=true
    os=Linux
    private_net:
        10.0.199.1
    status:
        age=44
        connected=true
        fencing=Stable
        license:
            have_license=true
        summary=Stable
        version=7.0.0.3
        wwns:
            210000e08b123d95
    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 275.

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:clusterOne > show node
Event at [ Dec 21 13:36:02 ]
node:
    bert:
        cellid=0
        enabled=true
        os=Linux
        private_net:
            10.0.199.3
        status:
            age=33
            connected=true
            fencing=Stable
            license:
                have_license=true
            summary=Stable
            version=7.0.0.3
            wwns:
                100000062b126ff8, 100000062b126ff9
        type=server_admin
    cxfsxe5:
        cellid=1
        enabled=true
        os=Linux
        private_net:
            10.0.199.1
        status:
            age=44
            connected=true
            fencing=Stable
            license:
                have_license=true
            summary=Stable
            version=7.0.0.3
            wwns:
```

```
                    210000e08b123d95
          type=server_admin
     cxfsxe10:
          cellid=3
          enabled=false
          os=Linux
          private_net:
               10.0.199.4
          status:
               age=-
               connected=false
               summary=Disabled
          type=client_only
     penguin17:
          cellid=4
          enabled=true
          os=Linux
          private_net:
               10.0.199.17
          status:
               age=54
               client=stable
               connected=true
               fencing=Stable
               filesystems=up
               license:
                    allocated=1
                    have_license=true
                    oem=none
                    os=SGI_Software_Foundation
                    membership=up
               summary=Stable
               version=7.0.0.3
               wwns:
                    210000e08b1a07d8
               xvm=up
          type=client_only
...
```

See also "Displaying the Keys with `cxfs_admin` After Installing CXFS" on page 124.

Enable or Disable CXFS Kernel Membership (`cmsd`) for the Local Node

The `cmsd` daemon controls CXFS kernel membership and heartbeat.

You must use the following command to allow CXFS kernel membership for the local server-capable administration node (the node on which `cxfs_admin`is running) after fixing the problems that required a forced CXFS shutdown (see "Shutdown of the Database and CXFS" on page 325):

```
cxfs_admin:mycluster> enable cmsd
```

You should revoke CXFS kernel membership of the local node only if an error requires you to perform a forced CXFS shutdown:

```
cxfs_admin:mycluster> disable cmsd
```

The result of the `disable cmsd` 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.

Note: The `enable cmsd` and `disable cmsd` commands are only effective when `cxfs_admin` is executed directly on a server-capable administration node. They only affect the local node (the node on which `cxfs_admin` is running).

Control and Contact a Node with `cxfs_admin`

Note: The `control` command must be run when `cxfs_admin` is executed directly on a server-capable administration node.

You can use the `control` command from a server-capable administration node to manually control and contact a target node that has a system controller. The target node must have the following attributes defined in the cluster database (see "Create or Modify a Node with `cxfs_admin`" on page 253):

```
reset_port
reset_node
reset_comms
reset_device
```

The node may also have the following attributes defined in the cluster database:

```
reset_password
reset_user
```

You can do the following:

- "Reset a Node with `cxfs_admin`" on page 268 (recommended)

- "Power-Cycle a Node with `cxfs_admin`" on page 268

- "NMI a Node with `cxfs_admin`" on page 268

- "Ping a Node System Controller with `cxfs_admin`" on page 269

Reset a Node with `cxfs_admin`

SGI recommends the reset method for system control on server-capable administration nodes. To reset the target node, which simulates the pressing of the reset button on the front of the machine (recommended), enter the following:

```
cxfs_admin:mycluster> control target-nodename operation=reset
```

For example, to reset the node named `mds1`:

```
cxfs_admin:mycluster> control mds1 operation=reset
```

When the target node is reset, other nodes in the cluster will detect the change and remove the target node from the active cluster. When the target node reboots, it will rejoin the CXFS kernel membership.

Power-Cycle a Node with `cxfs_admin`

To shut off power to the target node and then restart it, enter the following:

```
cxfs_admin:mycluster> control target-nodename operation=powerCycle
```

NMI a Node with `cxfs_admin`

To send a nonmaskable interrupt to the target node, which performs a core-dump of the operating system kernel that may be useful when debugging a faulty machine, enter the following:

```
cxfs_admin:mycluster> control target-nodename operation=nmi
```

> **Note:** NMI depends upon kernel support, which may not be present on all SGI ia64 systems; if the kernel support is not provided, the `nmi` setting will not work.
>
> NMI is not available on systems containing a baseboard management controller (BMC), such as SGI x86_64 systems.

Ping a Node System Controller with `cxfs_admin`

To contact the system controller for the target node via a `ping` command, enter the following:

```
cxfs_admin:mycluster> control target-nodename operation=ping
```

Move a Node Between the Cluster and the Pool with `cxfs_admin`

> **Note:** Prompting for the `detach` and `attach` commands is available only in advanced mode.

The `detach` command does the following:

- Removes the node from the cluster definition
- Releases the node's cell ID
- Moves the node to the pool (making it part of the `poolnode` class)

The `attach` command does the following:

- Adds the node to the cluster definition
- Creates a new cell ID for the node
- Removes the node from the `poolnode` class

> **Note:** Normally, you should only use the detach and attach commands if you want to release/establish a node's cell ID.
>
> If you want to change a node's cluster membership. disable and enable commands. (A disabled node still keeps its cell ID.) See:
>
> - "Disable a Node with `cxfs_admin`" on page 263
> - "Enable a Node with `cxfs_admin`" on page 263

To detach a node, enter the following:

detach *nodename*

To return the node to the cluster, enter the following:

attach *nodename*

See also:

- "Adding a New Server-Capable Administration Node to an Existing Cluster" on page 339

- "Adjusting the Cell ID Numbers" on page 344

Cluster Tasks with `cxfs_admin`

This section discusses the following:

- "Create a Cluster with `cxfs_admin`" on page 271

- "Modify a Cluster with `cxfs_admin`" on page 272

- "Specify a Tiebreaker with `cxfs_admin`" on page 273

- "Delete a Cluster with `cxfs_admin`" on page 274

- "Display a Cluster with `cxfs_admin`" on page 274

- "Show License Information with `cxfs_admin`" on page 275

- "Show CXFS Software Version with `cxfs_admin`" on page 276

Create 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:
```
     id=clusterID
     heartbeat_monitor=dynamic|static     (static)
```

For example:

```
cxfs_admin:> create cluster name=mycluster
```

Basic-mode usage notes:

* *clustername* specifies 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.

Advanced-mode usage notes:

* `id` is a unique number within your network in the range `1` through `255`. The default is `1`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.

  Note: Clusters must have unique IDs. See "Use a Private Network and Unique Cluster Name/ID" on page 50.

* `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.

Modify a Cluster with `cxfs_admin`

To modify the cluster, use the following command (line breaks shown here for readability, defaults in parentheses):

```
modify clustername
      tiebreaker=client_only_nodename
      notify_addr=[+/-]email_addresses
      notify_cmd=email_program
```

Advanced-mode:
      `heartbeat_monitor=dynamic|static`   *(static)*

Note: You cannot change the cluster's name or ID.

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`**

To email status changes to the users `joe` and `sally` using the `/usr/bin/mail` program, enter the following (line breaks shown here for readability):

`cxfs_admin:mycluster>` **`modify mycluster notify_cmd=/usr/bin/mail`**
`notify_addr=joe@MyCompany.com,sally@MyCompany.com`

Basic-mode usage notes:

- *clustername* identifies the object for which values will be modified (you cannot change the name of the cluster).

- `tiebreaker` specifies the CXFS tiebreaker. See "Specify a Tiebreaker with `cxfs_admin`" on page 273.

- `notify_addr` specifies the email address that CXFS will contact when there are cluster and node status changes. To specify multiple addresses, separate them with commas.

- `notify_addr` specifies the email program to use, such as `/usr/bin/mail`.

Advanced-mode usage notes:

- `heartbeat_monitor` specifies how CXFS kernel membership is monitored. See the description in "Create a Cluster with `cxfs_admin`" on page 271.

Specify 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.
>
> To ensure data integrity for all nodes, you must use reset or I/O fencing. 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.

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 an inactive 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:clusterOne > show cluster
Event at [ Dec 21 14:08:26 ]
cxfs:cluster:
    clusterOne:
        access:
            admin=server
            monitor=cluster
        autoconf:
            allowed:
                pg-27
            disallowed:
                (none)
        clusterid=9
        failover_net:
            (none)
        filesystem:
```

```
      zj01s0, zj01s1
heartbeat_monitor=static
log:
    clconfd, cli, crsd, diags
node:
    bert, cxfsxe5, cxfsxe10, penguin17, pg-27
status:
    filesystems:
        stable=true
        summary=Stable
    licenses:
        cxfs_client
    nodes:
        stable=true
        summary=Stable
    stable=true
    summary=stable
switch:
    brocade26cp1
tiebreaker=
```

Show License Information with `cxfs_admin`

To show the number of CXFS licenses available for the cluster, use the following command:

```
show licenses
```

For example:

```
cxfs_admin:clusterOne > show licenses
Event at [ Dec 21 16:07:25 ]
status:licenses:
    cxfs_client:
        allocated=1:
        valid=20
```

Show CXFS Software Version with `cxfs_admin`

To show the version of CXFS software that is running on each node defined in the cluster, use the following command:

```
show version
```

For example:

```
cxfs_admin:clusterOne > show version
Event at [ Dec 22 10:57:19 ]
node:bert:status:version=7.0.0.3
node:cxfsxe5:status:version=7.02.0.3
node:cxfsxe10:status:version=7.0.0.3
node:penguin17:status:version=7.0.0.3
node:penguin19:status:version=7.0.0.3
node:pg-27:status:version=7.0.0.3
```

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 277
- "Mount a CXFS Filesystem with `cxfs_admin`" on page 282
- "Unmount a CXFS Filesystem with `cxfs_admin`" on page 283
- "Relocate the Metadata Server for a Filesystem with `cxfs_admin`" on page 283

- "Delete a CXFS Filesystem with cxfs_admin" on page 284

- "Show a CXFS Filesystem" on page 285

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=fsname
    options=mount_options
    forced_unmount=true|false              (false)
    mountpoint=mountpoint                  (/mnt/fsname)
    mounted=true|false                     (true)
    grio_managed=true|false                (false)
    grio_qual_bandwidth=qualified_bandwidth
```

Advanced-mode:
```
    device=volumename                      (fsname)
    servers=server_list                    (all servers are potential MDS)
    nodes=nodelist                         (all nodes can mount)
    mount_new_nodes=true|false             (true)
```

You will use a similar set of subcommands to modify a CXFS filesystem via the `modify` command. To modify the following attributes, you must first unmount the filesystem:

```
device
grio_qual_bandwidth
mountpoint
options
```

Note: You must create XVM volumes with `xvm` and `mkfs` before you set it up using `cxfs_admin`.

The `device` value must be the XVM volume name. In basic mode, you are not prompted for a device name; `cxfs_admin` uses the value for `name` and prepends `/dev/cxvm/` to it. Therefore, in basic mode, the value you specify for `name` must be the XVM volume name. If you want to use a `name` value other than the XVM volume name, then you must use advanced mode and specify the `device` attribute.

Basic-mode usage notes:

- `name` specifies the name of the filesystem:

    - 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**

> **Note:** A filesystem name that is longer than 18 characters will be truncated in the `status` output. For more information about this command, see "`cxfs_admin` and Status" on page 384.

    - If you do not specify a value for `device`, then `name` must be the name of the XVM volume name following `/dev/cxvm`. For example:

`cxfs_admin:mycluster>` **create filesystem name=concat1**

- `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 7 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/`*fsname*.

  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`**

- `grio_managed` specifies whether a filesystem is managed by GRIOv2 (`true`) or not (`false`) if GRIOv2 has already been enabled on the node. The default is `false`. Setting `grio_managed` to `false` disables GRIO management for the specified filesystem, but it does not reset the `grio_qual_bandwidth` value. 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. To modify the qualified bandwidth, the filesystem must be unmounted.

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
```

Advanced-mode usage notes:

- `device` is the name of the XVM volume (minus `/dev/cxvm`). The default is the filesystem name specified by `name`.

  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.

  Note: Relocation is disabled by default. See "Relocation" on page 26.

  The default is all server-capable administration nodes in the cluster. 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 the initial mount of a filesystem.

  Note: Before deleting a server-capable administration node, you must first unmount any filesystems for which the node is a potential 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 nodes that can mount the filesystem. If you do not specify the `nodes` attribute on the `create` command, all nodes can mount the filesystem.

You can later change the list of nodes allowed to mount the filesystem by using the `modify` command. For example, to restrict mounting the `myfs` filesystem to just nodes `node1` and `node2`:

```
cxfs_admin:mycluster> create myfs nodes=node1,node2
```

To then add `node3` (thereby allowing `node1`, `node2`, and `node3` to mount the `myfs` filesystem):

```
cxfs_admin:mycluster> modify myfs nodes=+node3
```

To then prevent `node1` from mounting (thereby allowing just `node2` and `node3` to mount the `myfs` filesystem):

```
cxfs_admin:mycluster> modify myfs nodes=-node1
```

• `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
```

For example, using prompting in basic mode:

```
cxfs_admin:clusterOne> create filesystem
Specify the attributes for create filesystem:
 name? zj01s0
 options?
 forced_unmount? false
 mountpoint? /mnt/zj01s0
 mounted? true
 grio_managed? false
Event at [ Dec 21 15:04:15 ]
Filesystem "zj01s0" has been created, waiting for it to be mounted on all assigned nodes...
Waiting for filesystem zj01s0, current status: A server is trying to mount
Waiting for filesystem zj01s0, current status: cxfsxe5 trying to mount, penguin17 trying to mount, pg-27 trying to
mount
```

```
Waiting for filesystem zj01s0, current status: pg-27 trying to mount
Operation completed successfully
```

For example, using prompting in advanced mode:

```
cxfs_admin:clusterOne> create filesystem
Specify the attributes for create filesystem:
 name? zj01s0
 options?
 forced_unmount? false
 mountpoint? /mnt/zj01s0
 device? zj01s0
 servers? bert,cxfsxe5
 nodes? bert,cxfsxe10,cxfsxe5,penguin17,pg-27
 mounted? true
 mount_new_nodes? true
 grio_managed? false
Event at [ Dec 21 15:02:28 ]
Filesystem "zj01s0" has been created, waiting for it to be mounted on all assigned nodes...
Waiting for filesystem zj01s0, current status: A server is trying to mount
Waiting for filesystem zj01s0, current status: penguin17 trying to mount, pg-27 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 284.

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
```

To then add `node4` (thereby allowing `node2`, `node3`, and `node4` to mount the `myfs` filesystem):

```
modify myfs nodes=+node3
```

To then prevent `node4` from mounting (thereby allowing just `node2` and `node3` to mount the `myfs` filesystem):

```
modify myfs nodes=-node4
```

Note: When a node is configured to mount a filesystem, the node will automatically mount that filesystem when the node joins the cluster membership.

Unmount a CXFS Filesystem with `cxfs_admin`

To unmount a filesystem from all nodes in the cluster:

```
unmount filesystem
```

Note: As a convenience, `cxfs_admin` also accepts this command spelled as `umount`.

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: When a node is configured to unmount a filesystem, the node will automatically unmount that filesystem when the node joins the cluster 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 that 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 26).

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:clusterOne> unmount zj01s0
Event at [ Dec 21 15:09:13 ]
Waiting for filesystem "zj01s0" to be unmounted on all nodes
Waiting for filesystem zj01s0, current status: bert trying to unmount, cxfsxe5 trying to unmount, penguin17 trying
to unmount, pg-27 trying to unmount
Waiting for filesystem zj01s0, current status: bert trying to unmount, penguin17 trying to unmount, pg-27 trying
to unmount
Waiting for filesystem zj01s0, current status: Unmounted
Operation completed successfully
cxfs_admin:clusterOne> delete zj01s0
Event at [ Dec 21 15:09:31 ]
```

Show a CXFS Filesystem

To show information about all filesystems:

```
show filesystem
```

To show information about a specific filesystem:

```
show [filesystem:]fsname
```

For example, for a filesystem named `zj0ds0`:

```
cxfs_admin:clusterOne> show zj01s0
Event at [ Dec 21 15:11:19 ]
filesystem:zj01s0:
    device=zj01s0
    forced_unmount=false
    grio_managed=false
    grio_qual_bandwidth=0B
    mount=true
    mount_new_nodes=true
    mountpoint=/mnt/zj01s0
    nodes:
        bert, cxfsxe10, cxfsxe5, penguin17, pg-27
    options=
    servers:
        bert, cxfsxe5
    status:
        blocksize=4.00KB
        free=8.28GB
        nodes:
            bert=mounted
            cxfsxe10=disabled
            cxfsxe5=mounted
            penguin17=mounted
            pg-27=mounted
        server=bert
        size=8.37GB
        stable=true
        summary=Mounted
        utilization=1%
```

The filesystem status for a node can have one the following conditions:

- `disabled`: the filesystem is configured to be mounted, but the node is disabled and therefore the filesystem cannot mount

- `inactive`: the filesystem is configured to be mounted, but although the node is enabled it is not currently in membership (such as would be the case if CXFS services were stopped) and therefore the filesystem currently cannot mount

- `mounted`: the filesystem is configured to be mounted and is currently mounted

- `trying to mount`: the filesystem is configured to be mounted and is attempting to mount, but it may be having problems

- `unmounted`: the filesystem is not configured to be mounted and is therefore not mounted

Network Failover Modification Tasks with `cxfs_admin`

To limit the networks chosen from the `private_net` list or to reprioritize the list without making changes to the definition of each node, you can use the `failover_net` attribute.

!\ **Caution:** Do not configure `failover_net` while CXFS services are active; doing so can lead to cluster malfunction.

Command syntax:

```
create failover_net network=IPaddress
mask=netmask
```

For example, suppose that you already have the following definitions for `private_net` for a three-node cluster:

- nodeA:

  ```
  10.0.0.1
  10.1.0.1
  192.168.0.1
  ```

- nodeB:

  ```
  10.0.0.2
  10.1.0.2
  ```

```
                             192.168.0.2
```

- `nodeC:`

```
                             10.0.0.3
                             10.1.0.3
                             192.168.0.3
```

If you want to limit the networks chosen to those using the `192.168.0.`*x* subnet as the first priority and the `10.0.0.`*x* subnet as the second priority, and not use the `10.1.0.`*x* network at all, you would use the following commands:

```
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
```

Note: For IPv6, you must supply the *netmask* as the prefix number of bits. For example:

```
cxfs_admin:mycluster> create failover_net network=2001:db8::2:1 mask=64
cxfs_admin:mycluster> create failover_net network=fe80:: mask=64
```

For IPv4, you can use either an IP address or the number of bits for the *netmask*.

If the network specified for `failover_net` does not match any values specified for `private_net`, or if there are multiple interfaces specified for one node on a given network, `cxfs-config` will report an error.

Switch Tasks with `cxfs_admin`

This section discusses the following:

- "Create or Modify a Switch with `cxfs_admin`" on page 288
- "Delete a Switch Definition with `cxfs_admin`" on page 291
- "Show Switches with `cxfs_admin`" on page 291

For general information, see "I/O Fencing" on page 58.

> **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, see "Fencing Tasks with `cxfs_admin`" on page 293. For all of the options to update port information for switches, use the `hafence` command.

Create or Modify 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|voltaire|lsi      (brocade)
```

Advanced-mode:
```
  [mask=ports_that_will_not_be_fenced]
  [protocol=ssh|telnet]                      (ssh)
```

> **Note:** You must define all of the switches within your fabric to which a CXFS client or server is connected.

You will use a similar set of subcommands to modify a switch via the `modify` command.

Basic-mode usage notes:

- `name` specifies the hostname of the Fibre Channel, SAS, or InfiniBand switch; this is used to determine the IP address of the switch.

- `password` specifies the password for the specified *username*. The default is `password`.

> **Note:** For passwordless authentication, the following public and private keys should be stored on each server-capable administration node:
>
> ```
> /root/.ssh/id_rsa.pub
> /root/.ssh/id_rsa
> ```

- `user` specifies the user name to use when sending a message to the switch. The default is `admin`.

- `vendor` specifies the vendor of the switch. It can be one of the following values:

  ```
  brocade (default)
  lsi
  qlogic
  voltaire
  ```

  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 through 15 for `myswitch` will never be fenced by CXFS:

  ```
  cxfs_admin:mycluster> create switch name=myswitch mask=0,4,12-15
  ```

Note: For non-bladed switches, the `port` column of the `switchShow` output is the port number for CXFS.

For bladed switches, where the port number is not unique, the `port` column of `switchShow` output means the port number of one slot. However, CXFS requires a unique number identified by the combination of slot and port for bladed switch, which is usually represented by the `Index` or `Area` value in the `switchShow` output (depending upon the switch version). For example, the `switchShow` output below for the Brocade 48000 switch 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
...
```

For more details, see the switch documentation.

Server-capable administration nodes automatically discover the available HBAs and, when fencing is triggered, fence off all of the SAN HBAs when the `Fence` or `FenceReset` fail policy is selected. However, masked HBAs will not be fenced. Masking lets you 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.

- `protocol` specifies the protocol to use when sending a message to the switch, either `ssh` or `telnet`. The `ssh` option is only valid with Brocade and InfiniBand switches. The default is `telnet`.

Note: There are issues with using `telnet`. See "I/O Fencing" on page 58.

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_of_output_file*]

For example, in basic mode (truncated):

```
cxfs_admin:clusterOne> show switch
Event at [ Dec 21 15:20:59 ]
switch:
    brocade26cp1:
        hostname=brocade26cp1
        num_ports=192
        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, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
                    ...
        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_of_output_file*]

To display mask values, use advanced mode (see "Basic and Advanced Mode" on page 245.) For example, if myswitch is a unique name in the cluster database (truncated):

```
cxfs_admin:clusterOne> show switch
Event at [ Dec 04 09:34:31 ]
switch:
    brocade26cp0:
        hostname=brocade26cp0
        num_ports=256
        port:
        0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ...
        249, 250, 251, 252, 253, 254, 255
        protocol=ssh
        vendor=brocade
```

To display the switches and ports each host is connected to:

```
show wwns
```

For example:

```
cxfs_admin:clusterOne> show wwns
Event at [ Dec 21 15:26:11 ]
node:bert:status:wwns:
    100000062b126ff8:
        order=1
    100000062b126ff9:
        order=0
        switch=brocade26cp1
        switch_port=62
node:cxfsxe5:status:wwns:
    210000e08b123d95:
        order=0
        switch=brocade26cp1
        switch_port=167
node:penguin17:status:wwns:
    210000e08b1a07d8:
        order=0
        switch=brocade26cp1
        switch_port=41
node:pg-27:status:wwns:
    210000e08b1284c6:
        order=0
        switch=brocade26cp1
        switch_port=170
```

```
        210100e08b3284c6:
                order=1
```

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 `brocade26cp1`:

```
cxfs_admin:clusterOne> show brocade26cp1:port
Event at [ Dec 21 15:27:01 ]
switch:brocade26cp1:port:
    0:
        status=Enabled
        wwn=100000062b117c8c
    1:
        status=Enabled
        wwn=100000062b117954
    2:
        status=Enabled
        wwn=100000062b117c7c
    3:
        status=Enabled
        wwn=100000062b11796c
...
```

Fencing Tasks with `cxfs_admin`

This section discusses the following fencing tasks:

- "Query Fence Status" on page 294
- "Raise a Fence" on page 294
- "Lower a Fence" on page 294

Query Fence Status

To query the fencing status of all nodes, enter the `query fence` command when `cxfs_admin` is executed directly on a server-capable administration node. For example:

```
cxfs_admin:clusterOne> query fence
Event at [ Dec 21 15:29:07 ]
Waiting for shell command to end
  Switch[0] "brocade26cp1" has 192 ports
    Port 41 type=FABRIC status=enabled hba=210000e08b1a07d8 on host penguin17
    Port 62 type=FABRIC status=enabled hba=100000062b126ff9 on host bert
    Port 167 type=FABRIC status=enabled hba=210000e08b123d95 on host cxfsxe5
    Port 170 type=FABRIC status=enabled hba=210000e08b1284c6 on host pg-27
Operation completed successfully
```

Raise a Fence

To raise the I/O fence for a node, enter the following when `cxfs_admin` is executed directly on a server-capable administration node:

```
cxfs_admin:mycluster> raise fence node=nodename
```

To be prompted for required attributes, you must be in advanced mode.

Raising an I/O fence isolates the node from the SAN; CXFS sends a messages via the `telnet` (default) or `ssh` protocol to the switch and disables the port. After the node is isolated, it cannot corrupt data in the shared CXFS filesystem.

For example, to isolate node `pg-27`:

```
cxfs_admin:clusterOne> raise fence node=pg-27
Event at [ Dec 21 15:29:47 ]
Waiting for shell command to end
Operation completed successfully
```

Lower a Fence

To lower the I/O fence for a node, enter the following when `cxfs_admin` is executed directly on a server-capable administration node:

```
cxfs_admin:mycluster> lower fence node=nodename
```

To be prompted for required attributes, you must be in advanced mode.

Lowering the fence reenables the port and allows the node to reconnect to the SAN and access the shared CXFS filesystem.

For example, to reconnect node pg-27:

```
cxfs_admin:clusterOne> lower fence node=pg-27
Event at [ Dec 21 15:30:37 ]
Waiting for shell command to end
Operation completed successfully
```

Saving and Recreating the Current Configuration with `cxfs_admin`

This section discusses the following:

- "Creating `cxfs_admin` Scripts" on page 295
- "Recreating the Cluster Using `cxfs_admin` Scripts" on page 296

Creating `cxfs_admin` Scripts

The `cxfs_admin 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 or relative pathname of the file to contain the information:

```
config node output=pathname
```

You can use the generated file with the `-f` command line option to recreate the configuration at a later time.

Note: For a more-readable configuration output (without the related commands), use the `show` command rather than the `config` command.

To create `cxfs_admin` scripts, do the following:

1. Dump the entire cluster configuration to a file, such as `/tmp/config1`:

   `cxfs_admin:mycluster>` **`config * output=/tmp/config1`**

2. Edit the `/tmp/config1` file:

   a. 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.)

   b. Cut the `create filesystem` command lines at the end of the file and paste them into a second script file, such as `/tmp/config2`.

Recreating the Cluster Using `cxfs_admin` Scripts

To recreate the cluster using `cxfs_admin` scripts, do the following:

1. Verify that the cluster configuration is empty by using the `cxfs-config` command. For example:

```
server-admin# /usr/cluster/bin/cxfs-config
Global:
     cluster: <not defined> (id <n/a>)
     tiebreaker: <n/a>
     dynamic heartbeat: <n/a>

Networks:
     <none>

Machines:
     <none>

Autoconf:
     <none>

Filesystems:
     <none>

Switches:
```

```
     <none>

cxfs-config warnings/errors:
     no cluster configured
     no machines defined
```

2. Initialize the cluster and recreate the base cluster (except for the CXFS filesystems) by using the first script file (`/tmp/config1`):

```
server-admin# /usr/cluster/bin/cxfs_admin -s -f /tmp/config1
```

Note: Do not reboot or restart CXFS or cluster services before this script finishes.

3. Reboot the server-capable administration node:

```
server-admin# reboot
```

Repeat the `reboot` command on each server-capable administration node in the cluster.

4. Recreate the CXFS filesystems by using the second script file (`/tmp/config2`), providing the cluster name that was established as a result of step 2:

```
server-admin# /usr/cluster/bin/cxfs_admin -f /tmp/config2
```

Administration and Maintenance

This chapter discusses the following topics:

- "Administrative Tools" on page 300
- "Precedence of Configuration Options" on page 300
- "CXFS Release Versions and Rolling Upgrades" on page 301
- "CXFS Relocation Capability" on page 309
- "CXFS and Cluster Administration `service` Commands" on page 309
- "Using `hafence`" on page 310
- "Firewalls and CXFS Port Usage" on page 313
- "`chkconfig` Arguments" on page 315
- "Granting Task Execution Privileges for Users" on page 316
- "Transforming a Server-Capable Administration Node into a Client-Only Node" on page 317
- "CXFS Mount Scripts" on page 318
- "Using DMF" on page 319
- "Discovering the Active Metadata Server" on page 320
- "Shutdown of the Database and CXFS" on page 325
- "Avoiding a CXFS Restart at Reboot" on page 330
- "Log File Management" on page 331
- "Filesystem Maintenance" on page 331
- "Dump and Restore" on page 333
- "Hardware Changes and I/O Fencing" on page 334
- "Private Network Failover" on page 335
- "Cluster Member Removal and Restoration" on page 336

- "XVM Volume Mapping to Storage Targets" on page 354
- "Generation of Streaming Workload for Video Streams" on page 354
- "Frame Files Defragmentation and Analysis" on page 355
- "Disk Layout Optimization for Approved Media Customers" on page 356
- "Creating a Case-Insensitive CXFS Filesystem" on page 361

See also Chapter 13, "Cluster Database Management" on page 365.

Administrative Tools

You will use one of the following tools for CXFS administration:

- CXFS GUI connected to any server-capable administration node. See Chapter 10, "CXFS GUI" on page 167.

- cxfs_admin: for most commands, a cxfs_admin instance running on any node in the cluster network that has admin access permission (see "Setting cxfs_admin Access Permissions" on page 241). See Chapter 11, "cxfs_admin Command" on page 233.

  Note: A few cxfs_admin commands, such as stop_cxfs, must be run from a cxfs_admin instance running on a server-capable administration node.

Precedence of Configuration Options

Figure 12-1 shows the order in which CXFS programs take their configuration options.
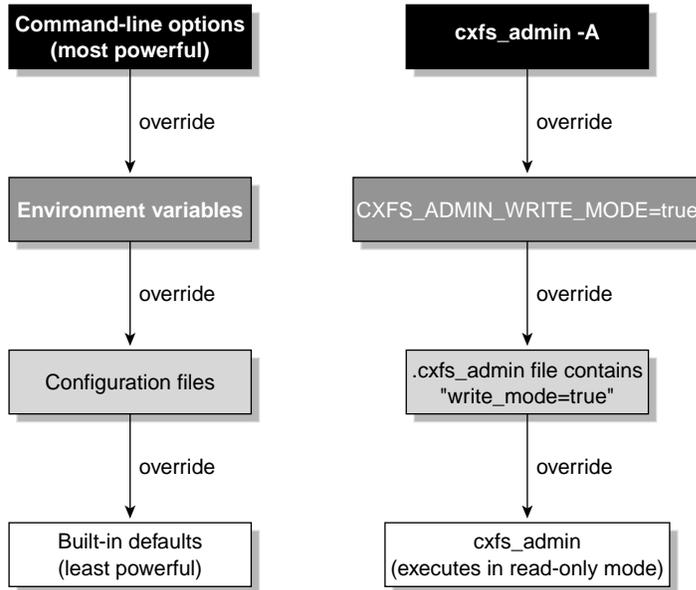
Figure 12-1 Precedence of Configuration Options

CXFS Release Versions and Rolling Upgrades

This section discusses the following:

- "Definition of a Rolling Upgrade" on page 302
- "Importance of Upgrading All Servers First" on page 302
- "Importance of An Accurate Cluster Database Across the Cluster Before Upgrading" on page 303
- "Upgrading Licenses" on page 303
- "General Upgrade Procedure" on page 303
- "Example Upgrade Process" on page 306

Definition of a Rolling Upgrade

SGI lets you upgrade of a subset of nodes from CXFS *X.anything* to CXFS *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.

Note: A rolling upgrade is not supported between major releases CXFS (*X.anything* to CXFS *Y.anything*). For a major release, you must upgrade all nodes in the cluster at the same time.

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 release.

Importance of Upgrading All Servers First

You must upgrade all server-capable administration nodes to a given CXFS release before upgrading any client-only nodes (including DMF parallel data-mover nodes, which are CXFS client-only nodes); server-capable administration nodes must run the same or later release as client-only nodes. Operating a cluster with client-only nodes running a mixture of older and newer CXFS versions may 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.

⚠ **Caution:** In some cases, improper upgrading can also result in a loss of functionality. For example, if CXFS client-only nodes, CXFS edge-serving nodes, or DMF parallel data-mover nodes are updated first (before the active metadata server), those nodes might not be able to mount the CXFS filesystems.

Importance of An Accurate Cluster Database Across the Cluster Before Upgrading

It is important that every server-capable administration node in the cluster has an accurate cluster database, in order to avoid inadvertently implementing an old database during the rolling upgrade process.

Upgrading Licenses

CXFS 7.0 and later releases require 7.0 licenses. For details about licenses, see Chapter 5, "CXFS Licensing" on page 113.

To upgrade to a 7.0 or later release, do the following:

1. Obtain 7.0 licenses from SGI. See "Obtaining the Keys from SGI" on page 120.

2. Edit the /etc/lk/keys.dat file on all server-capable administration nodes:

   a. Add the new 7.0 license keys.

   b. *(Optional)* Delete the prior licenses.

   > **Note:** Prior versions of CXFS will not function with the CXFS 7.0 license keys. If you are upgrading, you should keep your prior licenses available in the event that you must downgrade. If /etc/lk/keys/dat contains licenses from a prior release, they will be ignored after you upgrade to CXFS 7.0.

   To verify the licenses, see "License Key Verification" on page 120.

3. Follow the steps in "General Upgrade Procedure" on page 303.

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. Ensure that the configuration database is accurate on every server-capable administration node.

Note: If a given node has an inaccurate database, delete it by using
cdbreinit(**8**) and then follow the instructions in "Restoring a Database from
Another Node" on page 366.

3. Upgrade the potential metadata server (say admin2) for a given filesystem. (See
   the release notes and Chapter 7, "Server-Capable Administration Node
   Installation" on page 131.) Then reboot the potential metadata server.

4. For the first server-capable administration node that is an active metadata server
   (say admin1), move all CXFS filesystems running on it to the potential metadata
   server (admin2), making the potential metadata server 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.

5. Upgrade the server-capable administration node (admin1). See the release notes
   and Chapter 7, "Server-Capable Administration Node Installation" on page 131.

6. 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 7, 8, and 9 if your cluster has only two server-capable
   administration nodes.

7. For the next server-capable administration node that is an active metadata server
   (say admin3), move all CXFS filesystems running on it to the potential metadata

server (making the it 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
```

8. Upgrade the server-capable administration node (admin3). See the release notes and Chapter 7, "Server-Capable Administration Node Installation" on page 131.

9. 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 7 through 9 for each remaining server-capable administration node.

10. 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). Do the following:

   a. Enable relocation on the current active metadata server (admin2) by using the cxfs_relocation_ok system tunable parameter. See "Relocation" on page 26.

   b. For each filesystem for which admin2 is the now the active metadata server, manually relocate the metadata services back to the original metadata server (admin1) by using the CXFS GUI or cxfs_admin. For example:

   ```
   cxfs_admin:mycluster> relocate fs1 server=admin1
   ```

   c. Disable relocation. See "Relocation" on page 26.

11. 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.

12. Upgrade the client-only nodes. See the release notes for each platform and the *CXFS 7 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 6.0 at the beginning and Node2 is the potential metadata server , and the cluster is upgrading to the 6.1.

1 Starting configuration, all nodes running 6.0:

server-admin1 6.0	server-admin2 6.0	server-admin3 6.0	client4 6.0
/fs1 active server	/fs1 potential server	/fs1 client	/fs1 client
/fs2 potential server	/fs2 client	/fs2 active server	/fs2 client
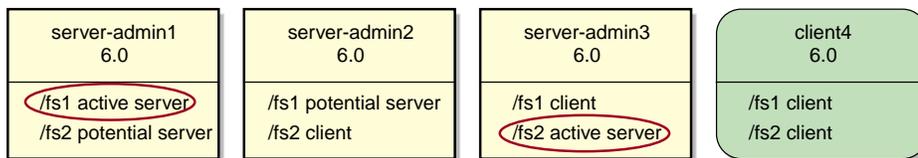
2 Upgrade server-admin2 to 6.1 and then reboot server-admin2:

server-admin1 6.0	server-admin2 6.1	server-admin3 6.0	client4 6.0
/fs1 active server	/fs1 potential server	/fs1 client	/fs1 client
/fs2 potential server	/fs2 client	/fs2 active server	/fs2 client
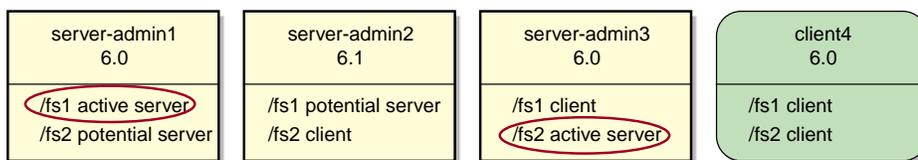
3 On server-admin1, run chkconfig *parameter* off and then reset and then reboot server-admin1 to force recovery of /fs1 onto server-admin2:

server-admin1 6.0	server-admin2 6.1	server-admin3 6.0	client4 6.0
	/fs1 active server	/fs1 client	/fs1 client
	/fs2 client	/fs2 active server	/fs2 client
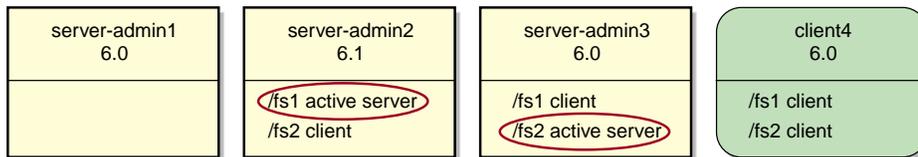
4 Upgrade server-admin1 to 6.1:

server-admin1 6.1	server-admin2 6.1	server-admin3 6.0	client4 6.0
	/fs1 active server	/fs1 client	/fs1 client
	/fs2 client	/fs2 active server	/fs2 client
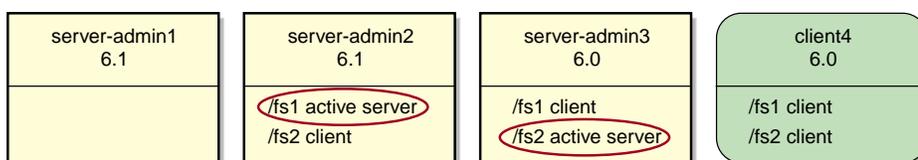
Figure 12-2 Example Rolling Upgrade Procedure (part 1)

5 On server-admin1, run chkconfig *parameter* on and then reset server-admin1:

Note: Ensure that there will be no I/O that will be restarted from server-admin1 to /fs1 or /fs2 after server-admin1 is reset.

server-admin1 6.1	server-admin2 6.1	server-admin3 6.0	client4 6.0
/fs1 potential server /fs2 potential server	/fs1 active server /fs2 client	/fs1 client /fs2 active server	/fs1 client /fs2 client
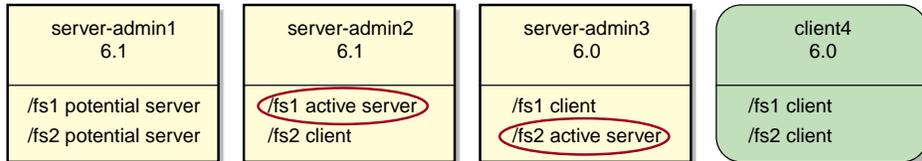
6 On server-admin3, run chkconfig *parameter* off and then reset server-admin3 to force recovery of /fs2 onto server-admin1:

server-admin1 6.1	server-admin2 6.1	server-admin3 6.0	client4 6.0
/fs1 potential server /fs2 active server	/fs1 active server /fs2 client		/fs1 client /fs2 client
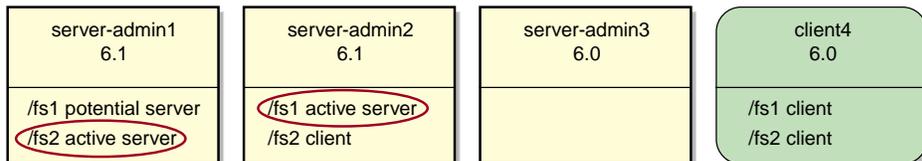
7 Upgrade server-admin3 to 6.1:

server-admin1 6.1	server-admin2 6.1	server-admin3 6.1	client4 6.0
/fs1 potential server /fs2 active server	/fs1 active server /fs2 client		/fs1 client /fs2 client
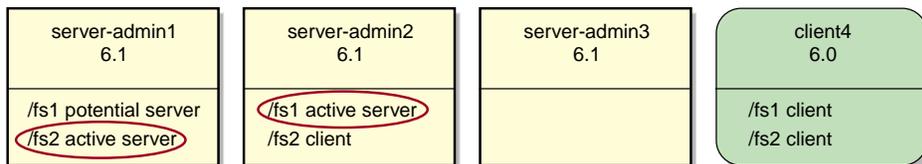
8 On server-admin3, run chkconfig *parameter* on and then reset server-admin3:

Note: Ensure that there will be no I/O that will be restarted from server-admin3 to /fs2 after server-admin3 is reset.

server-admin1 6.1	server-admin2 6.1	server-admin3 6.1	client4 6.0
/fs1 potential server /fs2 active server	/fs1 active server /fs2 client	/fs1 client /fs2 potential server	/fs1 client /fs2 client
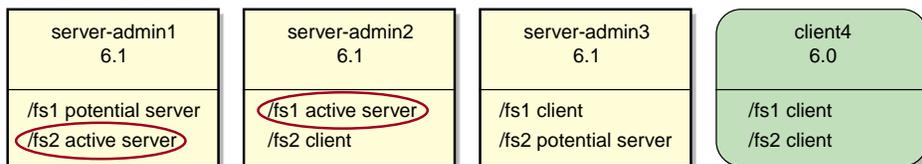
Figure 12-3 Example Rolling Upgrade Procedure (part 2)

9 To return the active metadata server for /fs2 to server-admin3, reset server-admin1:

Note: Ensure that there will be no I/O that will be restarted from server-admin1 to /fs2 after server-admin1 is reset.

server-admin1 6.1	server-admin2 6.1	server-admin3 6.1	client4 6.0
/fs1 potential server /fs2 potential server	/fs1 active server /fs2 client	/fs1 client /fs2 active server	/fs1 client /fs2 client
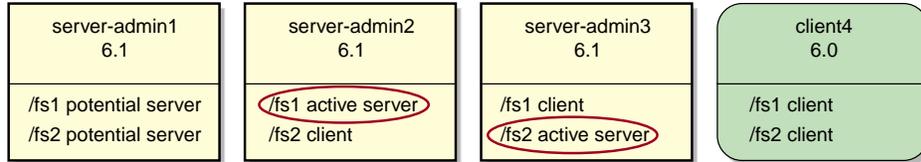
10 To return the active metadata server for /fs1 to server-admin1, reset server-admin2:

server-admin1 6.1	server-admin2 6.1	server-admin3 6.1	client4 6.0
/fs1 active server /fs2 potential server	/fs1 potential server /fs2 client	/fs1 client /fs2 active server	/fs1 client /fs2 client
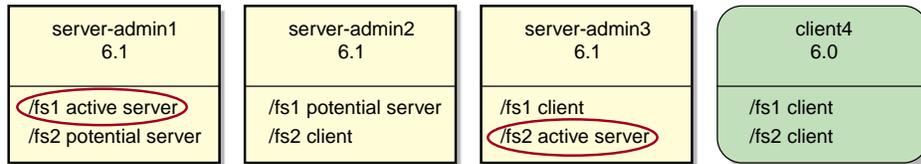
11 Upgrade the client-only server-admin4 to 6.1 (repeat for all other client-only nodes):

server-admin1 6.1	server-admin2 6.1	server-admin3 6.1	client4 6.1
/fs1 active server /fs2 potential server	/fs1 potential server /fs2 client	/fs1 client /fs2 active server	/fs1 client /fs2 client
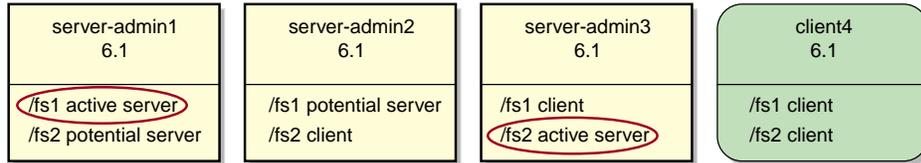
Figure 12-4 Example Rolling Upgrade Procedure (part 3)

CXFS Relocation Capability

Note: Relocation is disabled by default.

Relocating the metadata server for a given filesystem requires that the relocation feature is enabled on the filesystem's active metadata server. If you want to use relocation, SGI recommends that you enable the feature on all of the filesystem's potential metadata servers by resetting the cxfs_relocation_ok parameter as follows:

- Enable at run time:

  potentialMDS# **sysctl -w fs.cxfs.cxfs_relocation_ok=1**

- Enable at reboot by adding the following line to
  /etc/modprobe.d/sgi-cxfs-xvm.conf:

  options sgi-cxfs cxfs_relocation_ok=1

To disable relocation, do the following:

- Disable at run time:

  potentialMDS# **sysctl -w fs.cxfs.cxfs_relocation_ok=0**

- Disable at reboot by adding the following line to
  /etc/modprobe.d/sgi-cxfs-xvm.conf:

  options sgi-cxfs cxfs_relocation_ok=0

See also "Relocation" on page 26.

CXFS and Cluster Administration `service` Commands

Table 12-1 summarizes the service(8) services used for CXFS.

The commands to start, stop, restart (stop and then start), and give status (running or stopped) are as follows:

```
service servicename start
service servicename restart
service servicename stop
service servicename status
```

Table 12-1 CXFS and Cluster Administration `service` Services

Service	Description
cxfs	Controls the `clconfd` daemon (the CXFS server-capable administration node control daemon) on the local node, which in turn controls CXFS cluster services in the kernel
cxfs_client	Controls the `cxfs_client` daemon (the client daemon) on the local node, which in turn controls CXFS cluster services in the kernel
cxfs_cluster	Controls `fs2d`, `cmond`, `cad`, and `crsd` (the cluster administration daemons) on the local node
grio2	Controls `ggd2` (the GRIOv2 daemon) on the local node

Note: CXFS cluster services may also be stopped by other events, such as loss of membership.

For more information, see the `service`(8) man page and "CXFS Tools" on page 39.

Using `hafence`

To query the fencing status of a node, or raise or lower the fence for a node, you will normally use commands within `cxfs_admin`. See "Fencing Tasks with `cxfs_admin`" on page 293.

You can also run the following command on a server-capable administration node:

```
server-admin# /usr/cluster/bin/hafence -a -s switchname -u username -p password -m mask [-L vendorname]
```

To raise the fence for a node:

```
server-admin# /usr/cluster/bin/hafence -r nodename
```

To lower the fence for a node:

```
server-admin# /usr/cluster/bin/hafence -l nodename
```

To query switch status:

```
server-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 213.

  Server-capable administration nodes automatically discover the available HBAs and, when fencing is triggered, fence off all of the SAN 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, SAS, or InfiniBand switch; this is used to determine the IP address of the switch.

- -u specifies the user name to use when sending a message to the switch.

For example, the following defines a QLogic switch named myqlswitch and uses no masking:

server-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:

server-admin# **/usr/cluster/bin/hafence -a -s myqlswitch -u admin -p *** -m "#2,3" -L qlogic**

The following lowers the fence for client1:

server-admin# **/usr/cluster/bin/hafence -l client1**

The following raises the fence for client1:

server-admin# **/usr/cluster/bin/hafence -r client1**

The following queries port status for all switches defined in the cluster database:

server-admin# **/usr/cluster/bin/hafence -q**

For more information, see the hafence(8) man page. See the release notes for supported switches.

Firewalls and CXFS Port Usage

The CXFS private network should be restricted to CXFS use. If there are daemons other than CXFS daemons that are running on systems attached to the private network, you should configure them so that they listen for connections on the public networks only. A firewall on the CXFS private network is not required. However, if you use a firewall, be aware that CXFS uses the ports listed in Table 12-2 and Table 12-3.

Table 12-2 Ports Used by a Client-Only Node

Port/Protocol	Description
5449/tcp	cxfs_client connects to this port on a server-capable administration node
5449/udp	cxfs_client listens to this port for fs2d heartbeat traffic from server-capable administration nodes
5450/tcp	A client-only node connects to this port on the server-capable administration nodes for kernel messages (channel 0)
5451/tcp	A client-only node connects to this port on the server-capable administration nodes for kernel messages (channel 1)
5452/udp	Previously used for kernel discovery (prior to CXFS 6.6)
5453/udp	A client-only node listens for and sends multicast kernel heartbeat/discovery messages using this port

Table 12-3 Ports Used by a Server-Capable Administration Node

Port/Protocol	Description
22/tcp	`ssh` connections to the SAN switch for fencing
23/tcp	`telnet` connections to the SAN switch for fencing
111/tcp	TCP port, for more information see the Linux `rpcbind`(8) man page
111/udp	UDP port, for more information see the Linux `rpcbind`(8) man page
600-1023/tcp *(arbitrary assignment, typically in this range)*	`fs2d` registers its RPC service with the `rpcbind` utility with program number `391060` and version number `1`. The `rpcbind` utility then assigns an arbitrary port, typically in the range `600` through `1023`, for TCP RPC traffic. Various daemons running on the server-capable administration nodes (such as `cad`, `crsd`, `clconfd`, and `cmond`) will connect to the assigned port.
5435/tcp	CXFS GUI `cad` daemon, specified as `sgi-cad` in `/etc/services` (changeable by the site)
5449/tcp	`cxfs_admin` connects to this port on other server-capable administration nodes and `fs2d` accepts connections to this port from client-only nodes
5449/udp	`cxfs_admin` listens to this port for `fs2d` heartbeat traffic from other server-capable administration nodes and `fs2d` listens to this port for `fs2d` heartbeat traffic from other server-capable administration nodes
5450/tcp	A server-capable administration node accepts connections from all nodes and will itself connect to other server-capable administration nodes for kernel messages (channel 0)
5451/tcp	A server-capable administration node accepts connections from all cluster nodes and will itself connect to other server-capable administration nodes for kernel messages (channel 1)
5452/udp	Previously used for kernel discovery (prior to CXFS 6.6)

Port/Protocol	Description
5453/udp	A server-capable administration node listens for and sends multicast kernel heartbeat/discovery messages using this port
7500/udp	`crsd` daemon that handles node resets, specified as `sgi-crsd` in `/etc/services` (changeable by the site)

For example, suppose you have a cluster with two private networks and you want to move your Linux client-only nodes outside a firewall. To ensure that the `fs2d` and `cxfs_client` daemons only see port `5449` traffic on those networks, you could use the following `iptables`(8) commands on the Linux client-only nodes:

```
client-only# iptables -A INPUT --source 192.168.13.0/24 -p udp --dport 5449 -j RETURN
client-only# iptables -A INPUT --source 192.168.14.0/24 -p udp --dport 5449 -j RETURN
client-only# iptables -A INPUT -p udp --dport 5449 -j DROP
```

`chkconfig` Arguments

Table 12-4 summarizes the CXFS `chkconfig` arguments for server-capable administration nodes.

Table 12-4 `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 Linux 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.

The following shows the settings of the arguments on server-capable administration nodes:

```
server-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
```

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. For instructions, see "Privileges Tasks with the GUI" on page 227.

The cxfs_admin command lets you grant access permission to specific nodes using the access allow|deny subcommands. See "Setting cxfs_admin Access Permissions" on page 241.

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 55.

To transform a server-capable administration node into a client-only node, do the following:

1. Ensure that the node is not listed in the cluster database 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 37 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 7 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, "CXFS GUI" on page 167

* Chapter 11, "cxfs_admin Command" on page 233

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 storage 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 7 Client-Only Guide for SGI InfiniteStorage.*

Using DMF

DMF must make all of its DMAPI interface calls through the CXFS 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 DMF through the metadata server. This generally requires that DMF run on the CXFS metadata server. If DMF is managing a CXFS filesystem, DMF will ensure that the filesystem's CXFS metadata server is the DMF server and will use metadata server relocation if necessary to achieve that configuration.

Note: DMF data-mover processes must only run on the active CXFS metadata server (the DMF server node) and any DMF parallel data-mover nodes. Do not run data-mover processes on potential metadata server nodes.

DMF requires independent paths to tape drives so that they are not fenced by CXFS. The ports for the tape drive paths on the switch should be masked from fencing in a CXFS configuration.

The SAN must be zoned so that XVM does not failover CXFS filesystem I/O to the paths visible through the tape HBA ports when port fencing occurs. Therefore, either independent switches or independent switch zones should be used for CXFS/XVM volume paths and DMF tape drive paths.

To use DMF with CXFS, do the following:

- For server-capable administration nodes, install the `sgi-dmapi` and `sgi-xfsprogs` packages from the SGI InfiniteStorage Software Platform (ISSP) release. These are part of the software for the DMF server and the DMF parallel data-mover node. The DMF software will automatically enable DMAPI, which is required to use the `dmi` mount option.

  For CXFS client-only nodes, no additional software is required.

- When using the Parallel Data-Mover Option, install the **DMF Parallel Data Mover** software package, which includes the required underlying CXFS client-only software. (From the CXFS cluster point of view, the DMF parallel data-mover node is a CXFS client-only node but one that is dedicated to DMF data-mover activities.)

- Use the `dmi` option when mounting a filesystem to be managed.

- Start DMF on the CXFS active metadata server for each filesystem to be managed.

For more information about DMF, see the *DMF 6 Administrator Guide.*

Discovering the Active Metadata Server

This section discusses how to discover the active metadata server using various tools:

- "CXFS GUI and the Active Metadata Server" on page 321
- "`cxfs_admin` and the Active Metadata Server" on page 323
- "`clconf_info` and the Active Metadata Server" on page 324

CXFS GUI and the Active Metadata Server

To use the GUI to discover the active metadata server for a filesystem, 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.
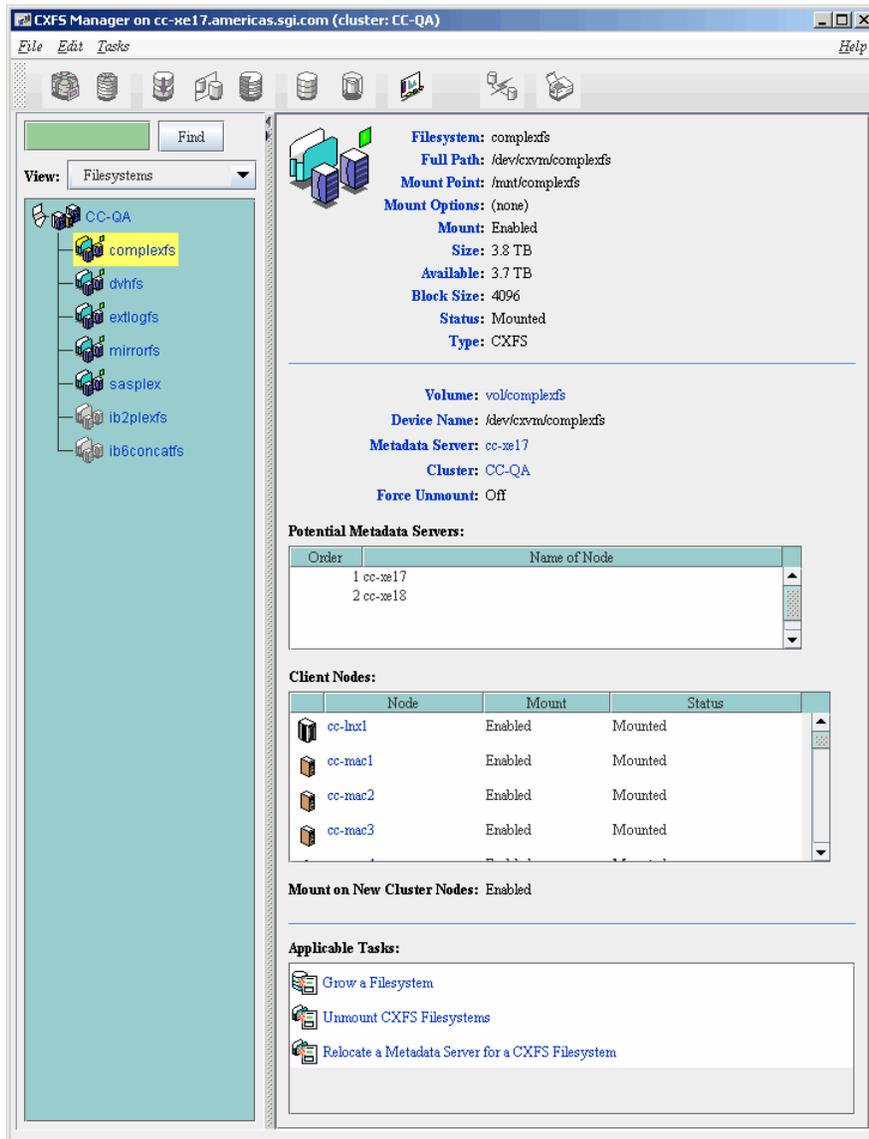
Figure 12-5 GUI Window Showing the Metadata Server

cxfs_admin and the Active Metadata Server

To use cxfs_admin to discover the active metadata server for a filesystem, do the following:

- To show information for all filesystems, including their active metadata servers:

  show server

  For example:

  ```
  cxfs_admin:clusterOne> show server
  Event at [ Oct 22 12:33:43 ]
  filesystem:zj01s0:status:server=bert
  filesystem:zj01s1:status:server=cxfsxe5
  ```

- 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 filesystem zj01s1 is a unique name in the cluster database:

  ```
  cxfs_admin:clusterOne> show zj01s1:*:server
  Event at [ Oct 22 12:34:43 ]
  filesystem:zj01s1:status:server=cxfsxe5
  ```

clconf_info **and the Active Metadata Server**

You can use the clconf_info command to discover the active metadata server for a given filesystem. For example, the following shows that bert is the metadata server for zj01s0:

```
# /usr/cluster/bin/clconf_info

Event at [2009-10-22 12:37:33]

Membership since Wed Oct 21 13:34:43 2009

_____ _____ _____ _____ _____
Node            NodeID Status   Age    CellID
_____ _____ _____ _____ _____
cxfsxe5              1 up          44       1
bert                3 up          33       0
pg-27               4 up           1       5
penguin17           5 up          54       4
cxfsxe10            6 Disabled     -       3
_____ _____ _____ _____ _____
2 CXFS FileSystems
/dev/cxvm/zj01s1 on /mnt/zj01s1  enabled  server=(cxfsxe5)  3 client(s)=(bert,penguin17,pg-27)  status=UP
/dev/cxvm/zj01s0 on /mnt/zj01s0  enabled  server=(bert)  3 client(s)=(cxfsxe5,penguin17,pg-27)  status=UP
```

Shutdown of the Database and CXFS

This section tells you how to perform the following:

- "Cluster Database Shutdown" on page 325

- "Normal CXFS Shutdown: Stop CXFS Services or Disable the Node" on page 326

- "Forced CXFS Shutdown: Revoke Membership of Local Node" on page 328

For more information about states, Chapter 14, "Monitoring Status" on page 381. If there are problems, see Chapter 15, "Troubleshooting" on page 393.

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 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 on a server-capable administration node:

```
server-admin# killall -TERM clconfd
server-admin# service cxfs_cluster stop
```

If you also want to disable the daemons from restarting at boot time, enter the following:

```
server-admin# /sbin/chkconfig grio2 off (If running GRIOv2)
server-admin# /sbin/chkconfig cxfs off
server-admin# /sbin/chkconfig cxfs_cluster off
```

For more information, see "chkconfig Arguments" on page 315.

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 is 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:

```
server-admin# service cxfs_cluster start
server-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.

- 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 206

- "Disable a Node with `cxfs_admin`" on page 263

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 328.

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 65.

> ⚠ **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 210

- "Disable a Node with `cxfs_admin`" on page 263

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 210

- "Disable a Node with `cxfs_admin`" on page 263

- "Enable a Node with `cxfs_admin`" on page 263

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 57.

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:

- Server-capable 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 on a server-capable administration node:

```
server-admin# /sbin/chkconfig grio2 off (If running GRIOv2)
server-admin# /sbin/chkconfig cxfs off
server-admin# /sbin/chkconfig cxfs_cluster off
server-admin# reboot
```

For more information, see "chkconfig Arguments" on page 315.

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
```

To customize log rotation, edit these scripts.

For information about log levels, see "Configure Log Groups with the GUI" on page 209.

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.

This section discusses the following:

- "Mounting Filesystems" on page 332
- "Unmounting Filesystems" on page 332
- "Growing Filesystems " on page 333

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 Guide*.

3. Mount the filesystem as an XFS filesystem using the `mount` command. For more information, see the `mount`(8) man page.

4. Use the `xfs_growfs` command or the CXFS GUI task; see "Grow a Filesystem with the GUI" on page 218.

5. Unmount the XFS filesystem. For more information, see the `umount`(8) man page.

6. Change the domain of the XVM volume back to a cluster volume using the `give` command.

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 server-capable administration 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 SAN cable rearrangement occurs

Note: Shut down the affected nodes before rearranging cables.

To check the status, use the following command on a server-capable administration node:

```
server-admin# /usr/cluster/bin/hafence -v
```

If any of the affected ports are disabled, you must manually enable them before starting CXFS on the affected nodes. For example, for a Brocade switch:

1. Connect to the switch using ssh or telnet.
2. Use the portenable command to enable the port.
3. Close the ssh or 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. 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 node:*
```

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 node:*
```

For more information, see Chapter 11, "cxfs_admin Command" on page 233.

Cluster Member Removal and Restoration

This section discusses removing and restoring cluster members for maintenance:

- "Manually Starting/Stopping CXFS" on page 337

- "Removing a Metadata Server from the Cluster Membership" on page 337

- "Restoring a Server-Capable Administration Node to the Cluster Membership" on page 339

- "Adding a New Server-Capable Administration Node to an Existing Cluster" on page 339

- "Adjusting the Cell ID Numbers" on page 344

- "Removing a Single Client-Only Node from the Cluster" on page 350

- "Restoring a Single Client-Only Node to the Cluster" on page 351

- "Stopping CXFS for the Entire Cluster" on page 353

- "Restarting the Entire Cluster" on page 353

These procedures are the safest way to perform these tasks but in some cases are not the most efficient. You should follow them 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 is 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 7 Client-Only 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:

```
server-admin# service grio2 start    (if running GRIOv2)
server-admin# service cxfs_cluster start
server-admin# service cxfs start
```

To stop CXFS processes manually on a server-capable administration node, enter the following commands:

```
server-admin# service grio2 stop    (stops GRIOv2 daemons)
server-admin# service cxfs stop    (stops the CXFS server-capable administration node control daemon)
server-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 Membership

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 membership, do the following:

1. Enable relocation by using the `cxfs_relocation_ok` system tunable parameter. See "Relocation" on page 26.

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 (such as admin2) by using the CXFS GUI or cxfs_admin. For example:

   cxfs_admin:mycluster> **relocate fs1 server=admin2**

3. Disable relocation. See "Relocation" on page 26.

   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 315.

Restoring a Server-Capable Administration Node to the Cluster Membership

To restore a server-capable administration node 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
   ```

Adding a New Server-Capable Administration Node to an Existing Cluster

To avoid problems, SGI recommends that the server-capable administration nodes all have lower cell ID numbers than any client-only nodes; see "Create an Initial Cluster of All Server-Capable Administration Nodes" on page 54. The cell ID number is established by CXFS when a node is added to the cluster definition, which happens automatically with cxfs_admin when you define a node.

If you want to add a new server-capable administration node to an existing cluster, you must determine if there is an available cell ID number for it that will be lower than the cell ID of any client-only node, or reconfigure the cluster definition so that such a number becomes available.

Following is an overview of the steps required:

1. Use the cxfs_admin status command to determine the current cell IDs. For more information, see "cxfs_admin and Status" on page 384.

Note: The GUI does not display cell ID numbers.

2. If a low cell ID number is available, define the new server-capable administration node with `cxfs_admin` (or use the corresponding GUI tasks to define the node and add it to the cluster).

3. If no low cell ID number is available, free the lowest cell ID that applies to a client-only node by using the `cxfs_admin disable` and `detach` commands to remove the associated client-only node from the cluster definition (or use the corresponding GUI task to remove the node from the cluster definition). For more information, see:

   - "Disable a Node with `cxfs_admin`" on page 263

   - "Move a Node Between the Cluster and the Pool with `cxfs_admin`" on page 269

   Note: Disabling a node will unmount CXFS filesystems and stop CXFS services on that node.

4. Define the new server-capable administration node with `cxfs_admin`, which automatically adds it to the cluster definition and assigns a cell ID number (or use the GUI tasks to define the node and add it to the cluster).

5. Restore the client-only node to the cluster definition by using the `cxfs_admin attach` and `enable` commands (or use the GUI task to add it to the cluster definition). For more information, see:

   - "Move a Node Between the Cluster and the Pool with `cxfs_admin`" on page 269

   - "Enable a Node with `cxfs_admin`" on page 263

6. Use the `cxfs_admin status` command again to verify that the new server-capable administration node has a lower cell ID number than any client-only node.

The following sections provide detailed examples using `cxfs_admin`:

- "Add Server: Low Cell ID Number is Available" on page 341

- "Add Server: No Low Cell ID Number is Available" on page 342

Note: To use the GUI, see the following sections:

- "Add or Remove Nodes in the Cluster with the GUI" on page 196
- "Define a Node with the GUI" on page 188

Add Server: Low Cell ID Number is Available

The following example (output truncated) shows the following:

- `mds1` and `mds2` are server-capable administration nodes (indicated by the a * character)

- `clientA` has the lowest cell ID (3) of any client-only node

- There is an available low cell ID number (2) between the current set of server-capable administration nodes (0 and 1) and the client-only nodes (3-5)

```
cxfs_admin:clusterOne > status
Event at [ Oct 22 13:08:07 ]
Cluster          : clusterOne
Tiebreaker       : clientA
Client Licenses  : allocated 3 of 5
------------------  --------  --------  -------------
Node                Cell ID   Age       Status
------------------  --------  --------  -------------
mds1 *              0         4         Stable
mds2 *              1         3         Stable
clientA             3         1         Stable
clientB             4         1         Stable
clientC             5         1         Stable
...
```

To add a new server-capable administration node to the `clusterOne` cluster, do the following:

1. Define the server-capable administration node (in this case, `mds3`), which automatically adds it to the cluster definition and assigns a cell ID number:

```
cxfs_admin:clusterOne> create node
Specify the attributes for create node:
 name? mds3
 type? server_admin
 private_net? 192.168.0.178
Event at [ Oct 22 13:08:08 ]
Node "mds3" has been created, waiting for it to join the cluster...
Please restart all cxfs and cluster services on the server "mds3" to make it
join the cluster.
```

2. Use the `status` command to verify that the new server-capable administration node has a lower cell ID number than any client-only node. For example:

```
cxfs_admin:clusterOne > status
Event at [ Oct 22 13:08:09 ]
Cluster         : clusterOne
Tiebreaker      : clientA
Client Licenses : allocated 3 of 5
------------------  --------  --------  -------------
Node                Cell ID   Age       Status
------------------  --------  --------  -------------
mds1 *              0         10        Stable
mds2 *              1         9         Stable
mds3 *              2         1         Stable
clientA             3         7         Stable
clientB             4         7         Stable
clientC             5         7         Stable
...
```

Add Server: No Low Cell ID Number is Available

The following example shows the following:

- `mds1` and `mds2` are server-capable administration nodes (indicated by the a * character)

- `clientA` has the lowest cell ID (2) of any client-only node

- There **is not** an available low cell ID number between the current set of
  server-capable administration nodes (0 and 1) and the client-only nodes (2–4)

```
cxfs_admin:clusterOne > status
Event at [ Oct 22 13:08:10 ]
Cluster         : clusterOne
Tiebreaker      : clientA
Client Licenses : allocated 3 of 5
------------------  --------  --------  ---------------
Node                Cell ID   Age       Status
------------------  --------  --------  ---------------
mds1 *              0         4         Stable
mds2 *              1         3         Stable
clientA             2         1         Stable
clientB             3         1         Stable
clientC             4         1         Stable
...
```

Because there is no available number, you must adjust the cell IDs. Do the following:

1. Free the lowest cell ID that applies to a client-only node (in this case, 2 for
   clientA) by using the following cxfs_admin commands to remove the
   associated client-only node from the cluster definition:

   ```
   cxfs_admin:clusterOne > disable clientA
   Event at [ Oct 22 13:08:11 ]
   Node "clientA" has been disabled, waiting for it to leave the cluster...
   Waiting for node clientA, current status: Disabled and unmounting
   Operation completed successfully
   cxfs_admin:clusterOne> detach clientA
   ```

2. Define the new server-capable administration node mds3, which automatically
   adds it to the cluster definition and assigns a cell ID number (the now-available 2):

```
cxfs_admin:clusterOne> create node
Specify the attributes for create node:
 name? mds3
 type? server_admin
 private_net? 192.168.0.178
Event at [ Oct 22 13:08:12 ]
Node "mds3" has been created, waiting for it to join the cluster...
Please restart all cxfs and cluster services on the server "mds3" to make it
```

```
join the cluster.
```

3. Restore the client-only node clientA to the cluster definition by attaching it and enabling it:

```
cxfs_admin:clusterOne > attach clientA
Event at [ Jan 11 15:27:58 ]
Node "clientA" has been enabled, waiting for it to join the cluster...
Waiting for node clientA, current status: Establishing membership
Waiting for node clientA, current status: Probing XVM volumes
Operation completed successfully
cxfs_admin:clusterOne> enable clientA
```

This automatically assigns a new cell ID number to clientA (the next available number, which is 5).

4. Use the status command to verify that the new server-capable administration node mds3 has a lower cell ID number than any client-only node:

```
cxfs_admin:clusterOne > status
Event at [ Oct 22 13:08:13 ]
Cluster        : clusterOne
Tiebreaker     : clientA
Client Licenses : allocated 3 of 5
-----------------  --------  --------  -----------------
Node               Cell ID   Age       Status
-----------------  --------  --------  -----------------
mds1 *             0         28        Stable
mds2 *             1         27        Stable
mds3 *             2         15        Stable
clientA            5         1         Stable
clientB            3         25        Stable
clientC            4         25        Stable
...
```

Adjusting the Cell ID Numbers

To avoid problems, SGI recommends that the server-capable administration nodes all have lower cell ID numbers than any client-only nodes; see "Create an Initial Cluster of All Server-Capable Administration Nodes" on page 54. The cell ID number is established by CXFS when a node is added to the cluster definition, which happens automatically with cxfs_admin when you define a node. When you remove a node from the cluster definition, its cell ID number is released.

If your cluster is experiencing problems, you should examine the current cell IDs and readjust as necessary. Following is an overview of the steps required:

1. Use the `cxfs_admin status` command to determine the current cell IDs. For more information, see "`cxfs_admin` and Status" on page 384.

Note: The GUI does not display cell ID numbers.

---

2. If a low cell ID number is available, assign it to the server-capable administration node that has an inappropriate cell ID by doing the following:

   a. Remove the inappropriate cell ID from the server-capable administration node by using the `cxfs_admin disable` and `detach` commands, which removes the node from the cluster definition and places it in the `poolnode` class (or use the corresponding GUI task to remove the node from the cluster definition, leaving it in the pool). For more information, see:

      • "Disable a Node with `cxfs_admin`" on page 263

      • "Move a Node Between the Cluster and the Pool with `cxfs_admin`" on page 269

   b. Assign a new cell ID (which will be the free low cell ID number) to the node by using the `cxfs_admin attach` and `enable` commands, which add the node back in to the cluster definition (or use the corresponding GUI task to add the node from the cluster definition).

3. If a low cell ID number is not available, adjust the cell IDs by doing the following:

   a. Remove the inappropriate cell ID from the server-capable administration node by using the `cxfs_admin disable` and `detach` commands, which removes the node from the cluster definition and places it in the `poolnode` class (or use the corresponding GUI task to remove the node from the cluster definition, leaving it in the pool). For more information, see:

      • "Disable a Node with `cxfs_admin`" on page 263

      • "Move a Node Between the Cluster and the Pool with `cxfs_admin`" on page 269

   b. Free the lowest cell ID that applies to a client-only node by using the `cxfs_admin disable` and `detach` commands to remove the associated

client-only node from the cluster definition (or use the corresponding GUI task to remove the node from the cluster definition).

c. Assign a new cell ID (which will be the now-free low cell ID number) to the server-capable administration node by using the `cxfs_admin attach` and `enable` commands, which add the node back in to the cluster definition (or use the corresponding GUI task to add the node from the cluster definition). For more information, see:

- "Move a Node Between the Cluster and the Pool with `cxfs_admin`" on page 269

- "Enable a Node with `cxfs_admin`" on page 263

d. Restore the client-only node to the cluster definition by using the `cxfs_admin attach` and `enable` commands (or use the GUI task to add it to the cluster definition). It will automatically be assigned a new, higher, cell ID number.

4. Use the `cxfs_admin status` command again to verify that the server-capable administration node has a lower cell ID number than any client-only node.

The following sections provide detailed examples using `cxfs_admin`:

- "Adjust Cell ID: Low Number is Available" on page 346

- "Adjust Cell ID: No Low Number is Available" on page 348

Note: To use the GUI, see the following sections:

- "Add or Remove Nodes in the Cluster with the GUI" on page 196
- "Define a Node with the GUI" on page 188

Adjust Cell ID: Low Number is Available

The following example (output truncated) shows the following:

- `mds1` and `mds2` are server-capable administration nodes (indicated by the a * character)

- `mds2` has a higher cell ID number (3) than a client-only node (`clientA` with number 2)

- `clientA` has the lowest cell ID (2) of any client-only node

- There is an available low number (1)

```
cxfs_admin:clusterOne > status
Event at [ Oct 22 13:08:14 ]
Cluster         : clusterOne
Tiebreaker      : clientA
Client Licenses : allocated 3 of 5
------------------  --------  --------  --------------
Node                Cell ID   Age       Status
------------------  --------  --------  --------------
mds1 *              0         4         Stable
mds2 *              3         1         Stable
clientA             2         2         Stable
clientB             4         1         Stable
clientC             5         1         Stable
...
```

To adjust the inappropriate cell ID for mds2, do the following:

1. Remove the inappropriate cell ID from the mds2 node by using the following commands to remove mds2 from the cluster definition:

```
cxfs_admin:clusterOne > disable mds2
Event at [ Jan 11 15:29:38 ]
Node "mds2" has been disabled, waiting for it to leave the cluster...
Waiting for node mds2, current status: Disabled and unmounting
Operation completed successfully
cxfs_admin:clusterOne> detach mds2
```

2. Assign a new cell ID (which will be the free number 1) to mds2 by using the following commands to add the node back into the cluster definition:

```
cxfs_admin:clusterOne > attach mds2
Event at [ Jan 11 15:27:58 ]
Node "mds2" has been enabled, waiting for it to join the cluster...
Waiting for node mds2, current status: Establishing membership
Waiting for node mds2, current status: Probing XVM volumes
Operation completed successfully
cxfs_admin:clusterOne> enable mds2
```

3. Use the `status` command to verify that `mds2` has a lower cell ID number than any client-only node:

```
cxfs_admin:clusterOne > status
Event at [ Oct 22 13:08:15 ]
Cluster          : clusterOne
Tiebreaker       : clientA
Client Licenses : 3 of 5
------------------  --------  --------  -------------
Node                Cell ID   Age       Status
------------------  --------  --------  -------------
mds1 *              0         22        Stable
mds2 *              1         1         Stable
clientA             2         20        Stable
clientB             4         19        Stable
clientC             5         19        Stable
...
```

Adjust Cell ID: No Low Number is Available

The following example (output truncated) shows the following:

- `mds1` and `mds2` are server-capable administration nodes (indicated by the a * character)

- `mds2` has a higher cell ID number (3) than a client-only node (`clientA` with number 1 and `clientB` with number 2)

- `clientA` has the lowest cell ID (1) of any client-only node

- There is no available lower number

```
cxfs_admin:clusterOne > status
Event at [ Oct 22 13:08:16 ]
Cluster          : clusterOne
Tiebreaker       : clientA
Client Licenses : allocated 3 of 5
------------------  --------  --------  -------------
Node                Cell ID   Age       Status
------------------  --------  --------  -------------
mds1 *              0         4         Stable
mds2 *              3         3         Stable
clientA             1         1         Stable
```

```
clientB              2            1            Stable
clientC              4            1            Stable
...
```

Because there is no available number, you must readjust the cell IDs. Do the following:

1. Remove the inappropriate cell ID from server-capable administration node `mds2` by removing it from the cluster definition:

```
cxfs_admin:clusterOne > disable mds2
Event at [ Oct 22 13:08:17 ]
Node "clientA" has been disabled, waiting for it to leave the cluster...
Waiting for node clientA, current status: Disabled and unmounting
Operation completed successfully
cxfs_admin:clusterOne> detach mds2
```

2. Free the lowest cell ID (`1`) by removing the associated client-only node (`clientA`) from the cluster definition:

```
cxfs_admin:clusterOne > disable clientA
Event at [ Oct 22 13:08:18 ]
Node "clientA" has been disabled, waiting for it to leave the cluster...
Waiting for node clientA, current status: Disabled and unmounting
Operation completed successfully
cxfs_admin:clusterOne> detach clientA
```

3. Restore `mds2` to the cluster definition (which automatically assigns it the now-free cell ID of `1`):

```
cxfs_admin:clusterOne > attach mds2
Event at [ Oct 22 13:08:19 ]
Node "mds2" has been enabled, waiting for it to join the cluster...
Waiting for node mds2, current status: Establishing membership
Waiting for node mds2, current status: Probing XVM volumes
Operation completed successfully
cxfs_admin:clusterOne> enable mds2
```

4. Restore `clientA` to the cluster definition:

```
cxfs_admin:clusterOne > attach clientA
Event at [ Oct 22 13:08:20 ]
Node "clientA" has been enabled, waiting for it to join the cluster...
Waiting for node clientA, current status: Establishing membership
```

```
Waiting for node clientA, current status: Probing XVM volumes
Operation completed successfully
cxfs_admin:clusterOne> enable clientA
```

5. Use the status command to verify that the new server-capable administration node has a lower cell ID number than any client-only node. For example:

```
cxfs_admin:clusterOne > status
Event at [ Oct 22 13:08:21 ]
Cluster        : clusterOne
Tiebreaker     : clientA
Client Licenses : allocated 3 of 5
------------------  --------  --------  -------------
Node              Cell ID   Age       Status
------------------  --------  --------  -------------
mds1 *            0         37        Stable
mds2 *            1         11        Stable
clientA           3         1         Stable
clientB           2         34        Stable
clientC           4         34        Stable
...
```

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:

```
mds# /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 7 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. *(If running GRIOv2)*Stop GRIOv2 services on a server-capable administration node:

```
server-admin# service grio2 stop
```

Repeat this step on each server-capable administration node.

3. Stop CXFS services on a server-capable administration node:

```
server-admin# service cxfs stop
```

Repeat this step on each server-capable administration node.

4. Stop the cluster daemons on a server-capable administration node:

```
server-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:

```
server-admin# service cxfs_cluster start
```

Repeat this step on each server-capable administration node.

2. Start CXFS services on a server-capable administration node:

```
server-admin# service cxfs start
```

Repeat this step on each server-capable administration node.

3. *(If running GRIOv2)*Start GRIOv2 services on a each server-capable administration node:

```
server-admin# service grio2 start
```

Repeat this on each server-capable administration node.

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 on a server-capable administration node to map XVM volumes to storage targets:

```
server-admin# /var/cluster/clconfd-scripts/cxfs-enumerate-wwns | grep -v "#"| sort -u
```

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) on Linux:

```
# 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)

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.

For example, the following Linux 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 356
- "Multiple Streams of Real-Time Applications" on page 357
- "The `filestreams` Mount Option" on page 359

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 12-6 shows an ideal frame layout.
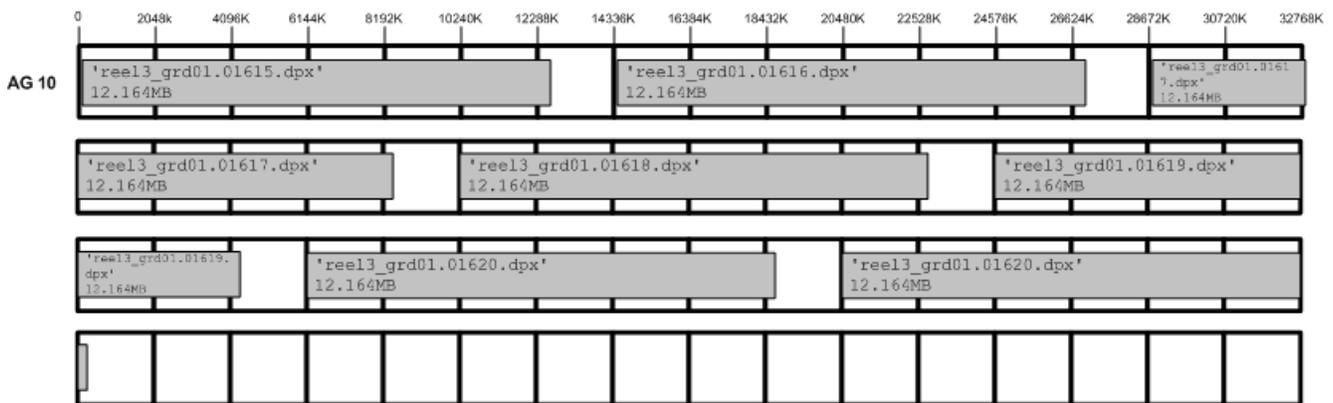
Figure 12-6 Ideal Frame Layout

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 12-7 shows an example of a 4-thread frametest read (2-MB stripe unit ∕ 1-GB cache size∕ prefetch = x1 ∕ 16 slices).
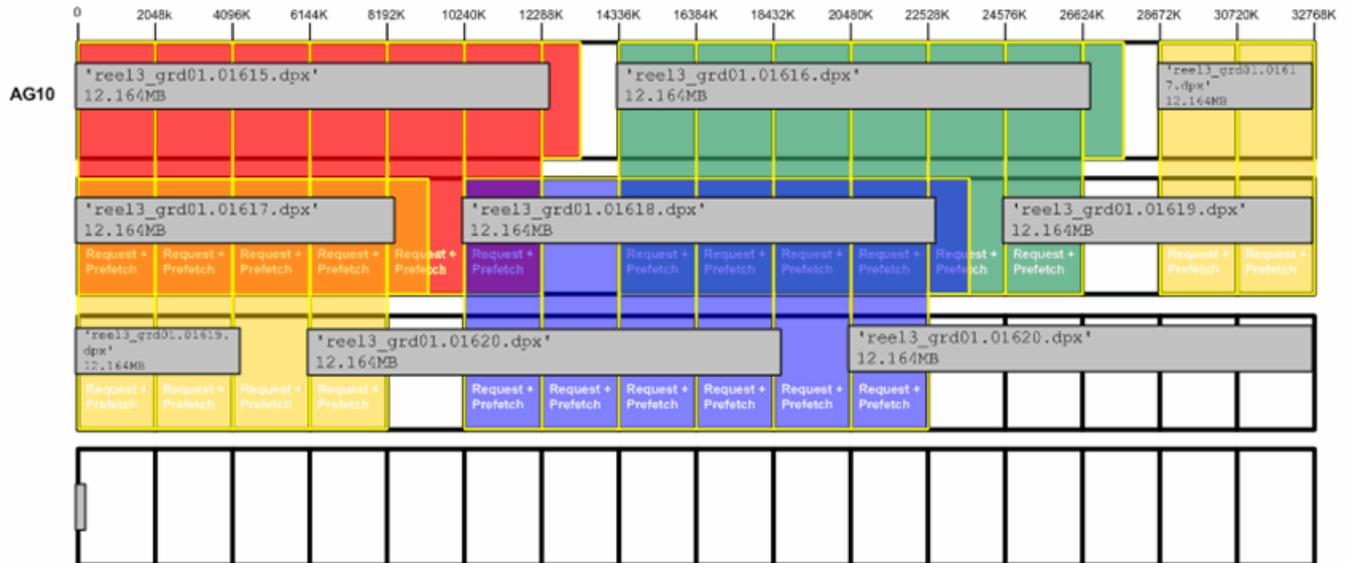
Figure 12-7 Ideal Frame Layout with RAID Prefetch

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 12-8 shows an example of multiple streams.Figure 12-9 shows an example of poor cache utilization.
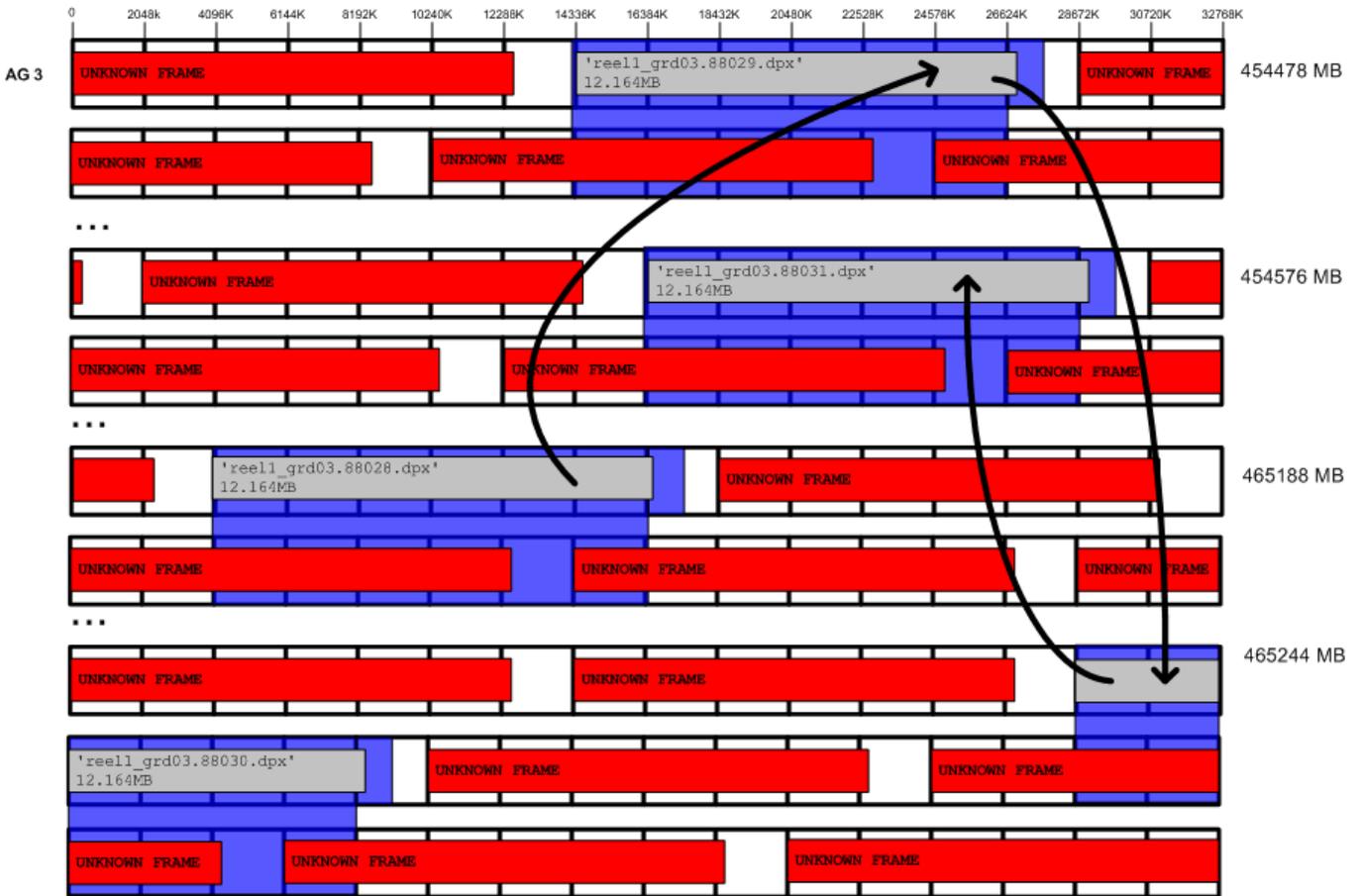
Figure 12-8 Multiple Streams of Real-Time Applications

```
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 12-9 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 a filesystem, therefore there is no need to rebuild a 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 12-10 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 12-10 Excellent Cache Utilization

For more information, contact SGI Support.

Creating a Case-Insensitive CXFS Filesystem

CXFS has limited support for case-insensitive filesystems:

- In ASCII filenames, lowercase and uppercase are treated as equal. This means the filesystem treats names that differ only in case as equivalent.

Note: It is not possible to rename a file to a name that only differs in case. For example, the following will not work:

```
# mv /cxfs/tp91/tmp/TST /cxfs/tp91/tmp/tst
mv: '/cxfs/tp91/tmp/TST' and '/cxfs/tp91/tmp/tst' are the same file
```

- The filesystem is case-preserving. This means the filesystem remembers the exact name that was used to create the file. This means that if a file was created with the name "File", it can be referenced using the name "FILE" or "file", but the reported name will always be "File".

- Case-insensitive CXFS filesystems are not supported on SLES 10 and RHEL client-only nodes. These nodes will fail to mount the filesystem with messages such as the following:

```
Preparing to mount CXFS file system "/dev/cxvm/tp91"
XFS: bad version
XFS: SB validate failed
```

Note: Nodes that use enhanced XFS support case-insensitive filesystems.

Note: Be aware that some applications rely on the case of filenames and will be confused when used with a case-insensitive filesystem.

Whether a CXFS filesystem is case-insensitive is determined when the filesystem is created. To create a case-insensitive filesystem, provide the following option to mkfs.xfs:

```
-n version=ci
```

For example:

```
# mkfs.xfs -n version=ci /dev/cxvm/tp91
meta-data=/dev/cxvm/tp91        isize=256   agcount=16, agsize=2746480 blks
        =                       sectsz=512  attr=2
data    =                       bsize=4096  blocks=43943680, imaxpct=25
        =                       sunit=16    swidth=32 blks
naming  =version 2              bsize=4096  ascii-ci=1
log     =internal log           bsize=4096  blocks=21472, version=2
        =                       sectsz=512  sunit=16 blks, lazy-count=0
realtime =none                  extsz=4096  blocks=0, rtextents=0
```

Cluster Database Management

This chapter contains the following:

- "Performing Cluster Database Backup and Restoration" on page 365
- "Validating the Cluster Configuration with `cxfs-config`" on page 370

Performing Cluster Database Backup and Restoration

This section discusses the following:

- "When to Perform a Database Backup" on page 365
- "Methods to Restore the Database" on page 365
- "Restoring a Database from Another Node" on page 366
- "Using `cdbBackup` and `cdbRestore` for the Cluster Database and Logging Information" on page 367

When to Perform a Database Backup

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.

Methods to Restore the Database

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 Database from Another Node" on page 366.

- 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 295.

- 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 367.

Restoring a 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 295.

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:

   server-admin# **service grio2 stop**

3. Stop the CXFS control daemon (clconfd) by running the following command on the node where step 1 was executed:

> ⚠️ **Caution:** Running this command will completely shut down all CXFS filesystem access on the local node.

```
server-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 1 was executed:

```
server-admin# service cxfs_cluster stop
```

5. Verify that all of the filesystems have been unmounted and the cluster membership has been terminated. To do this, run clconf_info on the active metadata server.

6. Run cdbreinit on the node where step 1 was executed:

```
server-admin# /usr/cluster/bin/cdbreinit
```

7. Wait for the following message to be logged to the syslog :

```
fs2d[PID]: Finished receiving CDB sync series from machine nodename
```

8. Start the CXFS control daemon by running the following command on the node where step 1 was executed:

```
server-admin# service cxfs start
```

9. *(If running GRIOv2)* Start the GRIOv2 daemon (ggd2) by running the following command on the node where step 1 was executed:

```
server-admin# service grio2 start
```

You can choose to have the cdbreinit command restart cluster daemons . 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. This
section discusses the following:

- "Backing Up the Cluster Database with cdbBackup" on page 368

- "Restoring the Cluster Database with cdbRestore" on page 368

Backing Up the Cluster Database with cdbBackup

To perform a backup of the cluster, use the cdbBackup command on each
server-capable administration node:

```
server-admin# /usr/cluster/bin/cdbBackup
```

⚠ **Caution:** Do not make configuration changes while you are using the cdbBackup
command.

For more information, see the cdbBackup(8) man page.

Restoring the Cluster Database with cdbRestore

Note: This procedure assumes that you have a good backup copy of the database,
obtained previously by using the cdbBackup command as described in "Backing Up
the Cluster Database with cdbBackup" on page 368

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. Disable all nodes in the cluster:

   ```
   server-admin# /usr/cluster/bin/cxfs_admin -A -c "disable node:*" [-i clustername]
   ```

   Note: If you have multiple clusters connected to the same public network, use the
   -i option to identify the cluster name.

2. Stop CXFS services on all nodes in the cluster.

3. Stop the cluster administration daemons on each server-capable administration
   node.

4. Delete the existing database files by running the cdbdelete command on each server-capable administration command:

```
server-admin# /usr/cluster/bin/cdbdelete /var/cluster/cdb/cdb.db
```

5. Use the cdbRestore command on each server-capable administration node:

```
server-admin# /usr/cluster/bin/cdbRestore  -d backup_directory -f backup_filename
```

Note: The name of the backup file (*backup_filename*) created by cdbBackup is displayed in the cdbBackup output.

For example, the following shows that cdb_cc-xe.Oct.27.2008.08:58:17.tar.Z is the value for *backup_filename* (line breaks shown here for readability):

```
server-admin# /usr/cluster/bin/cdbBackup
Saving cdb header file /var/cluster/cdb/cdb.db and
  cdb data directory /var/cluster/cdb/cdb.db# as cdb
  backup file cdb_cc-xe.Oct.27.2008.08:58:17.tar.Z
  under directory /var/cluster/cdb-backup
...done.
```

6. Start the cluster administration daemons on each server-capable administration node.

For example, to clear the database and then restore the database to all server-capable administration nodes, do the following (command output not shown):

Disable all nodes in the cluster:
```
server-admin# /usr/cluster/bin/cxfs_admin -A -c "disable node:*" [-i clustername]
```

(If running GRIOv2) On each server-capable administration node:
```
server-admin# service grio2 stop
```

On each server-capable administration node:
```
server-admin# service cxfs stop
```

On each server-capable administration node:
```
server-admin# service cxfs_cluster stop
```

Back up the database on one server-capable administration node:
```
server-admin# /usr/cluster/bin/cdbBackup
```

On each server-capable administration node:
server-admin# **/usr/cluster/bin/cdbdelete /var/cluster/cdb/cdb.db**

On each server-capable administration node:
server-admin# **/usr/cluster/bin/cdbRestore -d** *backup_directory* **-f** *backup_filename*

On each server-capable administration node:
server-admin# **service cxfs_cluster start**

On each server-capable administration node:
server-admin# **service cxfs start**

(If running GRIOv2) On each server-capable administration node :
server-admin# **service grio2 start**

For more information, see the cdbdelete(8) and cdbRestore(8) man pages.

Validating the Cluster Configuration with `cxfs-config`

The cxfs-config command validates configuration information in the cluster
database:

server-admin# **/usr/cluster/bin/cxfs-config**

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` or `ssh` 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, the following output shows that there are some issues with the way that server-capable administration node `bert` is configured:

```
cxfsxe5:~ # /usr/cluster/bin/cxfs-config
Global:
    cluster: clusterOne (id 9)
    cluster state: enabled
    tiebreaker:     dynamic heartbeat: disabled

Networks:
    net 0: type tcpip  10.0.0.0         255.255.255.0

Machines:
    node bert: node 3     cell 0  enabled  Linux64 server_admin
        hostname: bert.americas.sgi.com
        autoconf: false
        fail policy: Fence, Shutdown
        nic 0: address: 10.0.199.3 priority: 1 network: none

    node cxfsxe10: node 6     cell 3  disabled Linux64 client_only
        hostname: cxfsxe10.americas.sgi.com
        autoconf: false
```

```
        fail policy: Fence, Shutdown
        nic 0: address: 10.0.199.4 priority: 1 network: none

    node cxfsxe5: node 1     cell 1  enabled  Linux64 server_admin
        hostname: cxfsxe5.americas.sgi.com
        autoconf: false
        fail policy: Fence, Shutdown
        nic 0: address: 10.0.199.1 priority: 1 network: none

    node penguin17: node 5     cell 4  enabled  Linux32 client_only
        hostname: penguin17.americas.sgi.com
        autoconf: false
        fail policy: Fence, Shutdown
        nic 0: address: 10.0.199.17 priority: 1 network: none

    node pg-27: node 4     cell 5  enabled  Linux32 client_only
        hostname: pg-27.americas.sgi.com
        autoconf: false
        fail policy: Fence, Shutdown
        nic 0: address: 10.0.199.2 priority: 1 network: none

Autoconf:
    autoconf rule: pg-27
        policy: allowed
        hostname: pg-27
        enable_node: true

Filesystems:
    fs zj01s0: /mnt/zj01s0          enabled
        device = /dev/cxvm/zj01s0
        force = false
        options = []
        servers = bert (0), cxfsxe5 (1)
        clients = bert, cxfsxe10, cxfsxe5, penguin17, pg-27

    fs zj01s1: /mnt/zj01s1          enabled
        device = /dev/cxvm/zj01s1
        force = false
        options = []
        servers = bert (0), cxfsxe5 (1)
        clients = bert, cxfsxe10, cxfsxe5, penguin17, pg-27
```

```
Switches:
    switch 0: 192 port brocade admin@brocade26cp1          port 41: 210000e08b1a07d8 penguin17
        port 62: 100000062b126ff9 bert
        port 167: 210000e08b123d95 cxfsxe5
        port 170: 210000e08b1284c6 pg-27

cxfs-config warnings/errors:
    enabled machine bert has no NICs matching any net
    enabled machine cxfsxe5 has no NICs matching any net
    enabled machine penguin17 has no NICs matching any net
    enabled machine pg-27 has no NICs matching any net
    no NICs found on net 0
    server bert fail policy must not contain "Shutdown" for cluster
      with even number of enabled servers and no tiebreaker
    server cxfsxe5 fail policy must not contain "Shutdown" for cluster
      with even number of enabled servers and no tiebreaker
```

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 (output truncated):

```
cxfsxe5:~ # /usr/cluster/bin/cxfs-config -ping
Global:
...
Machines:
    node bert: node 3     cell 0  enabled  Linux64 server_admin
        hostname: bert.americas.sgi.com
        autoconf: false
        fail policy: Fence, Shutdown
        nic 0: address: 10.0.199.3 priority: 1 network: none
            ping: --- 10.0.199.3 ping statistics ---
            ping: 5 packets transmitted, 5 received, 0% packet loss, time 4004ms
            ping: rtt min/avg/max/mdev = 0.070/0.103/0.145/0.030 ms
...
```

- –xfs lists XFS information for each CXFS filesystem, such as size. For example (output truncated):

```
cxfsxe5:~ # /usr/cluster/bin/cxfs-config -xfs
Global:
...
Filesystems:
    fs zj01s0: /mnt/zj01s0          enabled
        device = /dev/cxvm/zj01s0
        force = false
        options = []
        servers = bert (0), cxfsxe5 (1)
        clients = bert, cxfsxe10, cxfsxe5, penguin17, pg-27
        xfs:
            magic: 0x58465342
            blocksize: 4096
            uuid: a53cf4d1-b89f-4c75-9b36-dcbdbaf80139
            data size 8.38 GB
...
```

- -xvm lists XVM information for each CXFS filesystem, such as volume size and topology. For example:

```
cxfsxe5:~ # /usr/cluster/bin/cxfs-config -xvm
Global:
...
Filesystems:
    fs zj01s0: /mnt/zj01s0          enabled
        device = /dev/cxvm/zj01s0
        force = false
        options = []
        servers = bert (0), cxfsxe5 (1)
        clients = bert, cxfsxe10, cxfsxe5, penguin17, pg-27
        xvm:
            vol/zj01s0                      0 online,open,accessible
                subvol/zj01s0/data       17576704 online,open,accessible
                    slice/zj01s0            17576704 online,open,accessible

            data size: 8.38 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(8) man page.

The following example shows errors reported by cxfs-config (line breaks shown for readability:

```
cxfsxe5:~ # /usr/cluster/bin/cxfs-config -check -all
Global:
    cluster: clusterOne (id 9)
    cluster state: enabled
    tiebreaker:     dynamic heartbeat: disabled

Networks:
    net 0: type tcpip  10.0.0.0         255.255.255.0

Machines:
    node bert: node 3     cell 0  enabled  Linux64 server_admin
        hostname: bert.americas.sgi.com
        autoconf: false
```

```
        fail policy: Fence, Shutdown
        nic 0: address: 10.0.199.3 priority: 1 network: none
            ping: --- 10.0.199.3 ping statistics ---
            ping: 5 packets transmitted, 5 received, 0% packet loss, time 4002ms
            ping: rtt min/avg/max/mdev = 0.048/0.099/0.134/0.033 ms

    node cxfsxe10: node 6     cell 3  disabled Linux64 client_only
        hostname: cxfsxe10.americas.sgi.com
        autoconf: false
        fail policy: Fence, Shutdown
        nic 0: address: 10.0.199.4 priority: 1 network: none

    node cxfsxe5: node 1     cell 1  enabled  Linux64 server_admin
        hostname: cxfsxe5.americas.sgi.com
        autoconf: false
        fail policy: Fence, Shutdown
        nic 0: address: 10.0.199.1 priority: 1 network: none
            ping: --- 10.0.199.1 ping statistics ---
            ping: 5 packets transmitted, 5 received, 0% packet loss, time 4000ms
            ping: rtt min/avg/max/mdev = 0.007/0.008/0.013/0.004 ms

    node penguin17: node 5     cell 4  enabled  Linux32 client_only
        hostname: penguin17.americas.sgi.com
        autoconf: false
        fail policy: Fence, Shutdown
        nic 0: address: 10.0.199.17 priority: 1 network: none
            ping: --- 10.0.199.17 ping statistics ---
            ping: 5 packets transmitted, 5 received, 0% packet loss, time 4001ms
            ping: rtt min/avg/max/mdev = 0.074/1.365/6.425/2.530 ms

    node pg-27: node 4     cell 5  enabled  Linux32 client_only
        hostname: pg-27.americas.sgi.com
        autoconf: false
        fail policy: Fence, Shutdown
        nic 0: address: 10.0.199.2 priority: 1 network: none
            ping: --- 10.0.199.2 ping statistics ---
            ping: 5 packets transmitted, 5 received, 0% packet loss, time 4002ms
            ping: rtt min/avg/max/mdev = 0.061/0.597/2.488/0.946 ms

Autoconf:
    autoconf rule: pg-27
```

```
        policy: allowed
        hostname: pg-27
        enable_node: true

Filesystems:
    fs zj01s0: /mnt/zj01s0          enabled
        device = /dev/cxvm/zj01s0
        force = false
        options = []
        servers = bert (0), cxfsxe5 (1)
        clients = bert, cxfsxe10, cxfsxe5, penguin17, pg-27
        xvm:
            vol/zj01s0                      0 online,open,accessible
                subvol/zj01s0/data        17576704 online,open,accessible
                    slice/zj01s0            17576704 online,open,accessible

            data size: 8.38 GB
        xfs:
            magic: 0x58465342
            blocksize: 4096
            uuid: a53cf4d1-b89f-4c75-9b36-dcbdbaf80139
            data size 8.38 GB

    fs zj01s1: /mnt/zj01s1          enabled
        device = /dev/cxvm/zj01s1
        force = false
        options = []
        servers = bert (0), cxfsxe5 (1)
        clients = bert, cxfsxe10, cxfsxe5, penguin17, pg-27
        xvm:
            vol/zj01s1                      0 online,open,accessible
                subvol/zj01s1/data        17576704 online,open,accessible
                    slice/zj01s1            17576704 online,open,accessible

            data size: 8.38 GB
        xfs:
            magic: 0x58465342
            blocksize: 4096
            uuid: ff8ea02c-d27b-4810-85b7-0f7f73988e79
            data size 8.38 GB
```

```
Switches:
    switch 0: 192 port brocade admin@brocade26cp1          port 41: 210000e08b1a07d8 penguin17
        port 62: 100000062b126ff9 bert
        port 167: 210000e08b123d95 cxfsxe5
        port 170: 210000e08b1284c6 pg-27

cxfs-config warnings/errors:
    enabled machine bert has no NICs matching any net
    enabled machine cxfsxe5 has no NICs matching any net
    enabled machine penguin17 has no NICs matching any net
    enabled machine pg-27 has no NICs matching any net
    no NICs found on net 0
    server bert fail policy must not contain "Shutdown" for cluster with even number
      of enabled servers and no tiebreaker
    server cxfsxe5 fail policy must not contain "Shutdown" for cluster with even number
      of enabled servers and no tiebreaker
```

Monitoring Status

This chapter discusses the following::

- "Methods to View System Status" on page 381
- "Status in Log Files" on page 382
- "Cluster, Node, and CXFS Filesystem Status" on page 383
- "I/O Fencing Status" on page 389
- "XVM Statistics" on page 391

Methods to View System Status

You can view the system status in the following ways:

- Query the status of an individual node or the cluster by using the GUI or `cxfs_admin`.

- Keep continuous watch on the state of a cluster using the GUI view area or the following `cxfs_admin` command:

```
# /usr/cluster/bin/cxfs_admin [-i clustername] -r -c "status interval=seconds"
```

> **Note:** If you have multiple clusters connected to the same public network, use the `-i` option to identify the cluster name.

- 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.

- View the system log file on client-only nodes.

- Manually test the filesystems with the `ls` command.

- 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: 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 when you are logged in as root on a host that has permission to access the CXFS cluster database (some tasks require that cxfs_admin be running on a server-capable administration node)

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: /var/log/cxfs/cad_log

* Kernel status (from clconfd): /var/log/cxfs/clconfd_*hostname*

* Command line interface log: /var/log/cxfs/cli_*hostname*

* Monitoring of other daemons: /var/log/cxfs/cmond_log

* Reset daemon log: /var/log/cxfs/crsd_*hostname*

* Output of the diagnostic tools such as the network connectivity tests: /var/log/cxfs/diags_*hostname*

* Cluster database membership status: /var/log/cxfs/fs2d_log

* System administration log: /var/lib/sysadm/salog (contains a list of the commands run by the GUI)

For information about client-only nodes, see *CXFS 7 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 383

- "cxfs_admin and Status" on page 384

- "cxfs_info and Status" on page 386

- "clconf_info and Status" on page 387

Also see "Key to Icons and States" on page 182.

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, "CXFS GUI" on page 167.

The easiest way to keep a continuous watch on the state of a cluster is to use the GUI 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.

- **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 permission for the CXFS cluster database. For complete details about `cxfs_admin`, see Chapter 11, "`cxfs_admin` Command" on page 233.

For example:

- To query node and cluster status, use the following `cxfs_admin` command on any host that has `monitor` access to the CXFS cluster database:

  ```
  status
  ```

  For more information about `monitor` access, see "Setting `cxfs_admin` Access Permissions" on page 241.

- 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:clusterOne (read only) > status
Event at [ Oct 22 13:08:07 ]
Cluster        : clusterOne
Tiebreaker     :
Client Licenses : allocated 2 of 5
----------------- -------- -------- ------------------------------------------------------------------------
Node             Cell ID  Age      Status
----------------- -------- -------- ------------------------------------------------------------------------
bert *           0        4        Mounted 1 of 2 filesystems
cxfsxe5 *        1        3        Stable
cxfsxe10         3        -        Disabled
penguin17        4        0        Establishing membership
pg-27            5        0        Establishing membership


----------------- ------------------ ------------------------------------------------------------------------
Filesystem       Server             Status
----------------- ------------------ ------------------------------------------------------------------------
zj01s0           cxfsxe5            1 of 5 nodes mounted, bert trying to mount
zj01s1           bert               Mounted [2 of 5 nodes]


----------------- ---------- --------------------------------------------------------------------------------
Switch           Port Count Known Fenced Ports
----------------- ---------- --------------------------------------------------------------------------------
brocade26cp1     192        4, 20, 21, 132, 223
```

Note: A filesystem name that is longer than 18 characters will be truncated in the status output. To display the entire filesystem name, use the show command. See "Show a CXFS Filesystem" on page 285.

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 stable client-only node. The path to cxfs_info varies by platform.

The cxfs_info command provides information about the cluster status, node status, and filesystem status. cxfs_info is run from a client-only node.

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 Linux client-only node named pg-27:

```
pg-27% /usr/cluster/bin/cxfs_info
cxfs_client status [timestamp Oct 22 12:45:55 / generation 5645836]

CXFS client:
    state: reconfigure (5), cms: quiesce, xvm: down, fs: down
Cluster:
    clusterOne (9) - enabled
Local:
    pg-27 (5) - enabled
Servers:
    bert       enabled  DOWN  0
    cxfsxe5    enabled  DOWN  1
Nodes:
    cxfsxe10   disabled DOWN  3
    penguin17  enabled  DOWN  4
    pg-27      enabled  DOWN  5
Filesystems:
    zj01s0
    zj01s1
```

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:

```
cxfsxe5:~ # /usr/cluster/bin/clconf_info

Event at [2009-10-22 13:12:41]

Membership since Thu Oct 22 13:12:41 2009
_____ _____ _____ _____ _____
Node            NodeID Status   Age    CellID
_____ _____ _____ _____ _____
cxfsxe5              1 up            3      1
bert                 3 Disabled     -       0
pg-27                4 Disabled     -       5
penguin17            5 Disabled     -       4
cxfsxe10             6 Inactive     -       3
_____ _____ _____ _____ _____
2 CXFS FileSystems
/dev/cxvm/zj01s1 on /mnt/zj01s1  enabled  server=(cxfsxe5)  0 client(s)=()  status=UP
/dev/cxvm/zj01s0 on /mnt/zj01s0  enabled  server=(cxfsxe5)  0 client(s)=()  status=UP
```

This command displays the following fields:

- Node is the node name.

- NodeID is the node identification number.

- Status is the status of the node, which may be up, Inactive, or Disabled.

- 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 identification number, 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 383.

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 `show fencing` command within `cxfs_admin` to show a summary of the fencing status per node

- Use the `query fence` command within `cxfs_admin` (when `cxfs_admin` is executed directly on a server-capable administration node) to show fencing details

- Use the `hafence` command as follows:

      /usr/cluster/bin/hafence -q

For example, the following output shows that all nodes are enabled:

```
cxfs_admin:clusterOne > show fencing
Event at [ Jan 28 14:18:05 ]
node:bert:status:fencing=Stable
node:cxfsxe5:status:fencing=Stable
node:cxfsxe10:status:fencing=Stable
node:penguin17:status:fencing=Stable
node:pg-27:status:fencing=Stable


cxfs_admin:clusterOne > query fence
Event at [ Jan 28 14:18:22 ]
Waiting for shell command to end
  Switch[0] "brocade26cp0" has 256 ports
    Port 41 type=FABRIC status=enabled hba=210000e08b1a07d8 on host penguin17
```

```
    Port 62 type=FABRIC status=enabled hba=100000062b126ff9 on host bert
    Port 167 type=FABRIC status=enabled hba=210000e08b123d95 on host cxfsxe5
    Port 170 type=FABRIC status=enabled hba=210000e08b1284c6 on host pg-27
    Port 201 type=FABRIC status=enabled hba=100000062b115f35 on host cxfsxe10
Operation completed successfully
```

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.

For example, the following `hafence` verbose (`-v`) output shows that port 4 is fenced (output truncated):

```
cxfsxe5:~ # /usr/cluster/bin/hafence -v
  Switch[0] "brocade26cp1" has 192 ports
    Port 0 type=FABRIC status=enabled hba=100000062b117c8c on host UNKNOWN
    Port 1 type=FABRIC status=enabled hba=100000062b117954 on host UNKNOWN
    Port 2 type=FABRIC status=enabled hba=100000062b117c7c on host UNKNOWN
    Port 3 type=FABRIC status=enabled hba=100000062b11796c on host UNKNOWN
    Port 4 type=FABRIC status=disabled hba=100000062b117ff0 on host UNKNOWN
    Port 5 type=FABRIC status=enabled hba=100000062b117958 on host UNKNOWN
    Port 6 type=FABRIC status=enabled hba=100000062b117f60 on host UNKNOWN....
```

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:

```
cxfsxe5:~ # /usr/cluster/bin/cms_failconf -q

CMS failure configuration:
        cell[0] bert      Reset
        cell[1] cxfsxe5  Reset
        cell[3] cxfsxe10          Fence Shutdown
        cell[4] penguin17         Fence Shutdown
        cell[5] pg-27     Fence Shutdown
```

XVM Statistics

You can use Performance Co-Pilot to monitor XVM statistics. To do this, you must enable the collection of statistics by using the pmstore command:

* 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 pmdumptext command 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 to monitor read and write activity at each XVM node.

* By using the pmchart command to view time-series data in the form of a moving graph.Figure 14-1 shows an example.
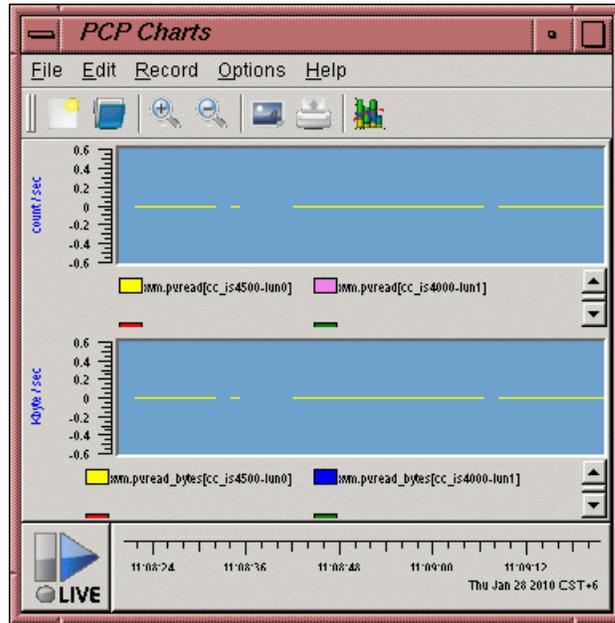
Figure 14-1 `pmchart`

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 Strategies" on page 393
- "Troubleshooting Tools" on page 398
- "Potential Problems" on page 408
- "Understanding Error Messages" on page 429
- "Corrective Actions" on page 463
- "Reporting Problems to SGI" on page 472

See the *CXFS 7 Client-Only Guide for SGI InfiniteStorage* for additional troubleshooting information.

Troubleshooting Strategies

To troubleshoot CXFS problems, do the following:

- "Know the Troubleshooting Tools" on page 394
- "Understand the Log Files" on page 394
- "Identify the Cluster Status" on page 394
- "Eliminate a Residual Cluster" on page 395
- "Determine If a Node Is Fenced" on page 396
- "Determine the Quorum Master" on page 396
- "Locate the Problem" on page 396
- "Redirect Switch Logs" on page 397

To avoid problems in the first place, follow the recommendations in Chapter 2, "CXFS Best Practices" on page 47.

Know the Troubleshooting Tools

You should be able to use the tools discussed in "Troubleshooting Tools" on page 398.

Understand the Log Files

You should have a good understanding of the log files discussed in "Status in Log Files" on page 382.

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 151.

- 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/log/cxfs/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/log/cxfs/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 hwinfo(8) command to display the hardware inventory. See "Physical Storage Tools" on page 398.

  – 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 320.

- Is the metadata server in the process of recovery? Look at the following file:

  /var/log/messages

  Messages indicate the recovery status. For example:

  – Recovery is in process:

Mar 13 11:31:02 1A:p2 unix: ALERT: CXFS Recovery: Cell 1: Client Cell 0 Died, Recovering </scratch/p9/local>

  – Recovery is complete:

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 kernel membership from a previous cluster. Enter the following to check for a cmsd kernel thread:

server-admin# **ps -ef | grep cmsd**

If the output shows a cmsd kernel thread, perform a forced CXFS shutdown by entering the following:

server-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 kernel 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)
```

Determine the Quorum Master

The quorum master is the server-capable administration node with the lowest node ID number (not cell ID). To confirm which node is the quorum master, look in the /var/log/cxfs/fs2d_log file for the most recent quorum message, such as the following example, which shows that node3 is the quorum master:

```
Tue Jul  7 12:52:41.490 cxfsopus4 fs2d  - Checking quorum with 4 members for any unknown members.
Tue Jul  7 12:52:41.490 cxfsopus4 fs2d  -  All quorum member machines known, master is node3
```

Locate the Problem

To locate the problem, do the following:

- Examine the log files (see "Understand the Log Files" on page 394):

  - Search for errors in all log files. See "Status in Log Files" on page 382. 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 202.

- Determine how the nodes of the cluster see the current CXFS kernel membership by entering the following command on each server-capable administration node:

  ```
  server-admin# /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:

  ```
  server-admin # /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.

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 398
- "Cluster Configuration Tools" on page 400
- "Cluster Control Tools" on page 401
- "Networking Tools" on page 402
- "Cluster/Node Status Tools" on page 403
- "Performance Monitoring Tools" on page 404
- "System Dump Analysis Tool" on page 404
- "Cluster Information Gathering Tool (`cxfsdump`)" on page 407

Physical Storage Tools

Understand the following physical storage tools:

- "Display Hardware Inventory with `lsscsi`"
- "Probe the LUNs with `echo`"
- "Discover Devices with `cxfs-reprobe`"
- "Show Physical Volumes"

Display Hardware Inventory with `lsscsi`

To display the hardware inventory:

```
# lsscsi
```

For more information, see the `lsscsi`(8) man page.

Probe the LUNs with `echo`

For all nodes with Fibre Channel or serial-attached storage (SAS) connections and some nodes with InfiniBand connections, use the following command to probe the LUNs on the specified SCSI host *hostX*:

```
# echo "- - -" > /sys/class/scsi_host/hostX/scan
```

Each "-" character is a wildcard for *bus*, *target*, and *LUN*, respectively.

Note: InfiniBand drivers and infrastructure on Linux are moving targets and the behaviors encountered can be distribution-specific; their documentation beyond the scope of this guide. In some Linux distributions, there is an issue with the InfiniBand `srp` driver such that issuing the above command can cause redundant targets to be created per InfiniBand host. For those distributions, the following is the preferred command to probe each selected target for new LUNs

```
# echo "bus target -" > /sys/class/scsi_host/hostX/scan
```

To determine the appropriate *bus* and *target* values, use the `lsscsi` output showing *host:bus:target:lun* for already-discovered targets. You may need to use the `srp_daemon` command to dynamically add new targets.

Discover Devices with `cxfs-reprobe`

Use the `cxfs-reprobe` script look for devices. `cxfs-reprobe` will also issue an XVM probe to tell XVM that there may be new devices available:

- On server-capable administration nodes:

  ```
  server-admin# /var/cluster/clconfd-scripts/cxfs-reprobe
  ```

- On client-only nodes:

  ```
  client# /var/cluster/cxfs_client-scripts/cxfs-reprobe
  ```

Show Physical Volumes

To show the physical volumes, use the `xvm` command. For example, from a Linux node:

```
linux# /sbin/xvm show -v phys/
```

The path to the xvm command varies by platform. For more information, see the appendix in *CXFS 7 Client-Only Guide for SGI InfiniteStorage.* For details about XVM, see the *XVM Volume Manager Administrator Guide.*

Cluster Configuration Tools

This section discusses the following:

- "XVM Configuration with the xvm Command" on page 400

- "CXFS Configuration with the CXFS GUI and cxfs_admin" on page 400

- "Database Reinitialization with cdbreinit" on page 401

- "Cluster Validation with cxfs-config" on page 401

XVM Configuration with the xvm Command

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 7 Client-Only Guide for SGI InfiniteStorage.* For details about XVM, see the *XVM Volume Manager Administrator Guide.*

CXFS Configuration with the CXFS GUI and cxfs_admin

To configure CXFS, use the CXFS GUI or cxfs_admin:

- The GUI:

  # **/usr/bin/cxfsmgr**

  See "GUI Features" on page 173 and Chapter 10, "CXFS GUI" on page 167.

- The cxfs_admin command:

  # **/usr/cluster/bin/cxfs_admin [-i *clustername*]**

  See

  > **Note:** If you have multiple clusters connected to the same public network, use the -i option to identify the cluster name.

  "Initial Setup with the cxfs_admin Command" on page 157 and Chapter 11, "cxfs_admin Command" on page 233.

Database Reinitialization with cdbreinit

To reinitialize the database, use the cdbreinit command:

server-admin# **/usr/cluster/bin/cdbreinit**

See "Recreating the Cluster Database" on page 468.

Cluster Validation with cxfs-config

To check the cluster configuration, use the following command from a server-capable administration node in the cluster:

server-admin# **/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 370.

Cluster Control Tools

Understand the cluster control tools:

- "Cluster Administration Daemons" on page 43

- 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 will cause CXFS to completely shut down on the local node.

See the following:

- "Ensure Cluster Database Membership Quorum Stability" on page 53

- "Restarting CXFS Services" on page 464

- "Clearing the Cluster Database" on page 464

- "Stopping and Restarting Cluster Administration Daemons" on page 468

• "CXFS Services" on page 29. Stopping CXFS services or shutting down CXFS services will cause its filesystems to be recovered by another potential metadata server.

Note: 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 210

  - "Allow Membership of the Local Node with the GUI" on page 210

  - "Disable a Node with cxfs_admin" on page 263

  - "Enable a Node with cxfs_admin" on page 263

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 151.

- To see cluster and filesystem status, use one of the following:

  - GUI:

    # **/usr/bin/cxfsmgr**

    See "Display a Cluster with the GUI" on page 205.

  - cxfs_admin command:

    # **/usr/cluster/bin/cxfs_admin -c status [-i *clustername*]**

    See "Display a Cluster with cxfs_admin" on page 274.

  - cxfs_info command on an client-only node (see *CXFS 7 Client-Only Guide for SGI InfiniteStorage*)

- To see the mounted filesystems:

  # **/bin/mount**
  # **/bin/df**

  For more information, see the mount(8) and df(1) man pages.

  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 Guide.*

Performance Monitoring Tools

Understand the following performance monitoring tools:

- To monitor system activity, use the sar(1) command:

  ```
  # /usr/bin/sar
  ```

- To monitor system input/output device loading on a Linux node, use the iostat(1) command. For example, to monitor at 2–second intervals for 10000000 times:

  ```
  linux# iostat 2 1000000
  ```

- To monitor process status, memory consumption, paging activity, block I/O operations, interrupts, context switches, and processor usage on a Linux node, use the vmstat(8) command. For example, to monitor at 1–second intervals for 1000 times:

  ```
  linux# 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 Guide.*

- To monitor system performance, use Performance Co-Pilot. See the following:

  -

  - *Performance Co-Pilot for Linux User's and Administrator's Guide*

  - pmie(1) man page

  - pmieconf(1)man page

System Dump Analysis Tool

For system dump analysis, use the crash(8) tool provided with SLES or RHEL.

To enable the collection of crash dumps, do the following:

1. Install the following RPMs, where *kernelrev* matches your installed kernel:

   - RHEL:

- – `kernel-debuginfo-`*kernelrev*

- – `kernel-debuginfo-common-`*kernelrev*

- – `system-config-kdump-`*kernelrev*

- SLES:

  - – `kernel-default-debuginfo-`*kernelrev*

  - – `kdump-`*version*

  - – `kexec-tools-`*version*

  For example, for the SLES `2.6.27.19-5-default` kernel, you would require `kernel-default-debuginfo-2.6.27.19-5.1` RPM, which would install the following file:

  `/usr/lib/debug/boot/vmlinux-2.6.27.19-5-default.debug`

  When you install the RHEL `kdump-`*version* RPM, it will automatically add the following information onto the `kernel` lines in the `/boot/grub/menu.lst` file:

  `crashkernel=256m-:128M@16M`

  Note: When you install the `kdump` RPM, `kdump` is automatically enabled.

2. Reboot, which activates the kernel and reserves the required memory. You will see the following message on the console:

   ```
   Loading kdump                                    done
   ```

3. Verify that the machine is set up correctly by requesting an NMI from the console:

   `console# `**`echo "c">/proc/sysrq-trigger`**

   Note: If there are several old dump files, the oldest one might be deleted by this process.

   For example:

```
console# echo "c">/proc/sysrq-trigger
SysRq : Trigger a crashdump
Initializing cgroup subsys cpuset
```

```
Initializing cgroup subsys cpu
...
```
(pages of output)

The key piece of information to look for are lines such as the following at the end of the output:

```
Saving dump using makedumpfile
--------------------------------------------------------------------------
Copying data                   : [ 100 %]

The dumpfile is saved to /root/var/crash/2009-10-28-13:05/vmcore.

makedumpfile Completed.
--------------------------------------------------------------------------
Generating README          Finished.
Copying System.map         Finished.
Copying kernel             Finished.
Copying kernel.debug       Finished.
```

Then the machine will reboot normally.

4. Go to the `/var/crash` directory and look for the dump directories that named according to the date and time. Each date directory will contain the files required for analysis. For example:

```
# cd /var/crash
console# ls
2009-10-13-21:02/  2009-10-26-15:55/
# ls -1 2009-10-26-15:55
README.txt
System.map-2.6.27.19-5-default
vmlinux-2.6.27.19-5-default.debug
vmlinux-2.6.27.19-5-default.gz
vmcore
```

For more information, see the crash(8) man page.

Note: On systems with CXFS installed, a set of crash extensions is available. These extensions are loaded with the `extend sgidbg` command in crash. The `sgi-cxfs-kmp-`*kernelrev* and `cxfs_utils` RPMs are required.

Cluster Information Gathering Tool (`cxfsdump`)

Before reporting a problem to SGI, do the following:

1. Ensure that there is passwordless `root ssh` access from the server-capable administration node to all other nodes in the cluster, other than Windows nodes. (Reverse access from the client-only nodes to the server—capable administration node is not needed.)

2. Run the `cxfsdump`(8) command on a server-capable administration node, which will collect the information such as the following across the cluster:

   • System information

   • CXFS client logs

   • CXFS version information

   • Network settings

   Execute the following:

   ```
   mds# /usr/cluster/bin/cxfsdump -secure -fast
   ```

> **Note:** The `-fast` option skips the collection of certain data that can be time consuming. The use of this flag is acceptable for most CXFS related issues.

3. If your cluster includes Windows nodes, run `cxfsdump` manually on those nodes. See the Windows chapter of *CXFS 7 Client-Only Guide for SGI InfiniteStorage.*

Potential Problems

The following are common problems and solutions:

- "GUI Will Not Run" on page 409
- "GUI Displays Invalid Filesystems" on page 410
- "`cxfs_admin` Output is Not Current" on page 410
- "Cannot Log In" on page 411
- "Client Membership Loss" on page 411
- "Node is Permanently Fenced" on page 413
- "Unable to Define a Node" on page 413
- "Node is Detected but Never Joins Membership" on page 413
- "Inappropriate Node Membership Loss" on page 413
- "System is Hung" on page 415
- "Cannot Access Filesystem" on page 415
- "Log Files Consume Too Much Disk Space" on page 415
- "Cell ID Count and `Membership delivered` Messages" on page 416
- "I/O Error in Filesystem" on page 416
- "Cannot Mount Filesystems" on page 417
- "Multiple `client_timeout` Values" on page 417
- "No HBA WWPNs are Detected" on page 418

- "XFS Internal Errors in System Log File" on page 420
- "Clients Unable to Remount Filesystems" on page 420
- "Forced Filesystem Shutdown Messages and XFS File Corruption" on page 421
- "IPMI Issues" on page 422
- "clconfd Is Not Running" on page 424
- "Slow Access to Files" on page 425
- "CXFS Cannot Access the Switch" on page 425
- "Metadata Server Panics After Reboot" on page 426
- "Issues with Dynamic TCP Port Assignment" on page 426
- "Issues after Restarting rpcbind" on page 426
- "Brocade telnet Session is Hung" on page 426
- "Clients Cannot Join the Cluster After Relocation" on page 427
- "df Display Problems for NFSv4-Exported Filesystems " on page 427
- "mkinitrd Fails" on page 427
- "Stability Issues on Clients Due to Token Optimizations" on page 427
- "Membership Issues After Installing or Upgrading Licenses" on page 428
- "XVM Volume Problems on Only One Node" on page 428

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 150

  – "License Key Error" on page 437

- Are the cluster daemons running? See "Verify that the Cluster Daemons are Running" on page 151.

- 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.

- Is the following line is commented out in the file /etc/ld.so.conf?

  ```
  /usr/lib64/sysadm/lib
  ```

  For example:

  ```
  Error connecting to server: /usr/lib64/sysadm/bin/sysadmd: error
        while loading shared libraries: libsysadmUtil.so.0: cannot open
        shared object file: No such file or directory.
  ```

  To resolve the problem, do the following:

  1. Uncomment the /usr/lib64/sysadm/lib or /usr/lib/sysadm/lib line in the file /etc/ld.so.conf.

  2. Make your change take effect:

     ```
     # ldconfig
     ```

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.

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:

```
server-admin# /usr/cluster/bin/hafence -U
```

Cannot Log In

If you cannot log in to a server-capable administration node, you can use one of the following commands, assuming that the node you are on is listed in the other nodes' `.rhosts` files:

```
# rsh hostname ksh -i
# rsh hostname csh -i
```

Client Membership Loss

The following message indicates a problem with client membership:

```
Error -N reading mesg header channel X cell Y
```

where *N* can be one of:

Number	Description
61	No data available
104	Connection reset by peer
110	Connection timed out

The following series of messages indicate that a client has lost membership (line breaks added here for readability):

```
Mar 15 10:55:35 5A:mvcxfs2 kernel: Error -61 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 -61 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 -61 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 clues as to why the channel was disconnected. Table 15-1 on page 412 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 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 334.

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 125.

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 Chapter 8, "Postinstallation Steps" on page 137.

Inappropriate Node Membership Loss

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 ia64 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 (these are 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 Systems" on page 67.

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.

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 331. You may also want to consider choosing a less-verbose log level; see the following:

- "`cad.options` on Server-Capable Administration Nodes" on page 138

- "`fs2d.options` on Server-Capable Administration Nodes" on page 139

- "Configure Log Groups with the GUI" on page 209

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. In the following example, `704` is a 64–bit hexadecimal bitmask of cells included in the membership and cell 2 has been in the last 4 CXFS memberships:

```
[10798.974447] Cell 10 (cxfsxe14) joined the membership
[10798.989690] Membership delivered for cells
<0:0:0:0:0:0:0:704>
[10799.011084] Cell(age): 2(4) 8(7) 9(17) 10(1)
```

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 Chapter 8, "Postinstallation Steps" on page 137.

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 421, 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 334.

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 Guide*.

Multiple `client_timeout` Values

The `clconfd` and `cxfs_client` daemons set the `client_timeout` value. The value depends on the order in which filesystems are mounted on the various nodes and adapts to help ensure that all filesystems get mounted in a timely manner. The value may differ among nodes and has no effect on the filesystem operation after it is mounted.

⚠️ **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.

The `retry` value is forced to be 0 and you cannot change it.

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 Linux 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, SAS, or InfiniBand 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 ssh (for Brocade or InfiniBand) or 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`, or `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 216) 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`.

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 79.

If you think you have a filesystem with real corruption, do the following from an Linux node with XFS commands RPM (sgi-xfsprogs) installed:

1. Disable the filesystem in CXFS:

   linux# **/usr/cluster/bin/cxfs_admin -A -c 'unmount *filesystem*' [-i *clustername*]**

2. Mount the device in order to replay the log:

   linux# **mount *device* *any_mount_point***

3. Unmount the filesystem:

   linux# **unmount *device***

4. Check the filesystem:

   linux# **xfs_check *device***

5. View the repairs that could be made, using xfs_repair in no-modify mode:

    linux# **xfs_repair -n** *device*

6. Capture filesystem file name and inode pairs:

    linux# **xfs_ncheck device > xfs_ncheck.out**

7. If you are certain that the repairs are appropriate, complete them:

    linux# **xfs_repair** *device*

8. Enable the filesystem in CXFS:

    linux# **/usr/cluster/bin/cxfs_admin -A -c 'mount** *filesystem***' [-i** *clustername***]**

IPMI Issues

This section discusses the following IPMI issues:

- "BMC Does Not Respond to a ping Command" on page 422
- "ipmitool Command Fails" on page 422
- "Node is Not Reset" on page 424

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 Appendix C, "BMC System Controller" on page 487.

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 Appendix C, "BMC System Controller" on page 487.
- 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 Appendix C, "BMC System Controller" on page 487.

- Does the BMC have a valid IP address assigned? See step 4 in Appendix C, "BMC System Controller" on page 487.

- 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:

```
# 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 422.

- Is the correct version of `ipmitool` installed? See step 1 in Appendix C, "BMC System Controller" on page 487.

- Have the `admin` user name and password been set on the BMC with the required `ADMINISTRATOR` privileges? See step 3 in Appendix C, "BMC System Controller" on page 487.

- Does the `ipmitool` command contain all of the required arguments, including the `lan` 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 lan -o intelplus -H bmc-nodename -U admin -P admin_password command
```

  For example:

```
# ipmitool -I lan -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 Appendix C, "BMC System Controller" on page 487.

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 198

  - "Create or Modify a Node with `cxfs_admin`" on page 253

- 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 422.

- 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 422

  - Appendix C, "BMC System Controller" on page 487

`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.

The `clconfd` process will not run if there is a licensing problem. See "`clconfd` Daemon Death" on page 436.

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
  ```

CXFS Cannot Access the Switch

There is only a single active admin login permitted to Fibre Channel, SAS, or InfiniBand switches that are properly configured for CXFS. If there is a defunct admin login remaining, CXFS may be unable to access the switch.

To solve this problem, do the following:

1. Log in to the switch as root.

2. Determine the process IDs of the inactive admin logins. For example:

   ```
   brocade:root> ps -ef | grep defunct
   root       9179  9173  0 Apr12 ?        00:00:00 [login] <defunct>
   root      18311 18308  0 May13 ?        00:00:00 [login] <defunct>
   ```

3. Kill the defunct login process IDs. For example:

   ```
   brocade:root> kill -9 9179 18311
   ```

4. Remove the contents of the login records file:

   ```
   brocade:root> cat /dev/null > /var/run/utmp
   ```

Metadata Server Panics After Reboot

Rebooting the metadata server without first shutting down CXFS services can cause the metadata server to panic. You must use the proper procedures for shutting down nodes. See "Shut Down Nodes Unobtrusively" on page 80.

Issues with Dynamic TCP Port Assignment

The dynamic TCP port assignment used by the fs2d daemon may result in a collision with another service that is already using a fixed port number.

One example that has been observed is Samba hanging during startup when trying to query the CUPS printing system for printer information. Other applications that expect to connect on a well-known port may be affected.

To see which ports fs2d is using, use the following command:

```
# lsof -i TCP | grep fs2d
```

If you encounter this problem, one option is to restart fs2d with a SIGHUP signal so that it will move to another port:

```
# /usr/bin/killall -HUP fs2d
```

You may have to repeat this command until fs2d is assigned to a port that is not in contention.

Issues after Restarting rpcbind

If you restart rpcbind without the -w option to cause a *warm start*, the fs2d daemon will not be able to communicate with other CXFS daemons, and the cluster will not function normally. In this case, you should force a restart of the fs2d daemon so that it can continue to communicate with the various CXFS daemons:

```
# killall -9 fs2d
```

Brocade telnet Session is Hung

The Brocade telnet login session can become hung for Brocade switches running firmware version V5* and later. To clear the situation, do the following:

1. Log in to the switch as root.

2. Kill the defunct `telnet` login process IDs.

3. Execute the following command:

   `# `**`cat /dev/null > /var/run/utmp`**

Clients Cannot Join the Cluster After Relocation

If a CXFS client fails or exits the cluster during the metadata server relocation process, the relocation process and the client recovery are likely to hang. This prevents any clients, including the failed client, from joining the cluster.

Once in this state, it may be possible to resolve the deadlock by resetting or power-cycling the `fs2d` quorum master. See "Determine the Quorum Master" on page 396.

`df` Display Problems for NFSv4-Exported Filesystems

NFSv4-exported filesystems do not display properly in `df` output on Linux nodes. This is an NFS issue that exists independent of CXFS.

`mkinitrd` Fails

If `mkinitrd` fails, it may be because a new kernel module was installed but the `sgi-xfsprogs` RPM was not installed. To resolve this problem, install the `sgi-xfsprogs` RPM and rerun `mkinitrd`.

Stability Issues on Clients Due to Token Optimizations

CXFS token prefetch and range tokens are designed as optimizations for applications using CXFS filesystems on a CXFS client. However, under some workloads, may cause stability issues. If directed to do so by SGI Support, you can use the `cell_tkm_feature_disable` system tunable parameter to disable these features on Linux and Mac OS X clients.

Note: You should modify `cell_tkm_feature_disable` only if directed to do so by SGI Support.

For more information, see "`cell_tkm_feature_disable`" on page 510 and the Linux and Mac OS X chapters of *CXFS 7 Client-Only Guide for SGI InfiniteStorage*.

Membership Issues After Installing or Upgrading Licenses

If you encounter membership issues after installing or upgrading licenses, you may need to restart the fs2d daemon. See "Restarting fs2d After Installing or Upgrading Licenses" on page 120.

XVM Volume Problems on Only One Node

If an XVM volume is not functioning on only one node, it may be that there is a mismatch of required RPMs on that node. For XVM to function properly, a node must have matching versions of the following RPMs:

- XVM kernel: sgi-xvm-kmp-default (SLES) or kmod-xvm (RHEL)

- XVM user space: sgi-xvm-commands

- Path manager: sgi-pm-commands

If there is an inconsistency in the RPM levels, the XVM volume may function normally on the CXFS metadata server and other client-only nodes. However, on the node with a mismatch of RPM versions, XVM will report the volume as problematic (as offline, as inaccessible, and/or with no physical connection) despite the fact that the SCSI device (/dev/sd*) can be read without problems. For example:

```
xvm:cluster> show *
phys/zj                          0 cluster,inaccessible
slice/zjs0               366878464 offline,open,inaccessible
slice/zjs1               366878464 offline,inaccessible
slice/zjs2               366878464 offline,inaccessible
slice/zjs3               366878464 offline,inaccessible
subvol/zjs0/data         366878464 offline,pieceoffline,open,inaccessible
subvol/zjs1/data         366878464 offline,pieceoffline,inaccessible
subvol/zjs2/data         366878464 offline,pieceoffline,inaccessible
subvol/zjs3/data         366878464 offline,pieceoffline,inaccessible
vol/zjs0                         0 offline,open,no physical connection,inaccessible
vol/zjs1                         0 offline,no physical connection,inaccessible
vol/zjs2                         0 offline,no physical connection,inaccessible
vol/zjs3                         0 offline,no physical connection,inaccessible
```

If a mismatch is the cause, a message similar to one of the following should appear in the system log:

```
XVM api version 24 doesn't match kernel api version 23: kernel/user api

    pm api version mismatch: user:7, kernel:8
```

To resolve the problem, upgrade the out-of-date client-only node with the appropriate RPMs.

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.

Note: The `/var/log/messages` file may contain spurious error messages. This problem occurs in clusters with multiple private networks when one of the active network interfaces is downed by using the `ifconfig` command. This problem may also happen when the network is interrupted for other reasons.

Some of the transport failure messages generated may have cell IDs that are different from the cell ID of the node with the downed interface. The spurious error messages do not appear to affect the continued operation of the cluster.

This section discusses the following:

- "Normal Messages" on page 430
- "`cxfs-config` Messages" on page 433
- "Relocation Error" on page 434
- "Shutdown Failure" on page 434
- "Controller Disable Messages" on page 435
- "CMS Error Messages" on page 435
- "`clconfd` Daemon Death" on page 436
- "Out of Logical Swap Space" on page 436

- "Lost CXFS Membership" on page 436

- "License Key Error" on page 437

- "IP Address Error" on page 438

- "System Log File Errors" on page 439

- "Daemon Log File Errors" on page 451

- "`cdbreinit` Errors" on page 456

- "`cxfs_admin` Errors" on page 457

- "Mount Errors" on page 458

- "Authorization Errors" on page 458

- "Connection Error" on page 459

- "`remote version is too old` Error" on page 459

- "`node is downrev` Error" on page 459

- "`EXTENT` Errors" on page 460

- "GRIOv2 Errors" on page 461

- "`foswitch` Errors" on page 461

- "Leaving Transient State Errors" on page 461

- "`CXFS Cluster Admin Shutdown:failed` Message" on page 462

- "`alive` Message Errors" on page 462

- "Remote Clients Error" on page 463

- "`unable to lock bootconfig` Error" on page 463

Normal Messages

Many messages are normal and do not indicate a problem. Following is a subset of normal messages:

```
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 <0:0:0:0:0:0:0:704>
```

> Membership has been delivered for the specified node. 704 is a
> binary bitmask of cell numbers for which membership has been
> delivered; 704 equates to cells 2, 8, 9, and 10.

```
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 <0:0:0:0:0:0:0:704>
```

> The number here is a bitmask of node numbers on which a reset is
> being requested. In this case, 704 is a binary bitmask of cells being
> reset (704 equates to cells 2, 8, 9, and 10. 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.

`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.

```
cell 9:  Membership daemon started
```

> The CXFS membership daemon has started on the node with cell ID 9.

```
Starting tcp server for interface 192.168.14.47 channel 0
```

> The machine has started accepting connections from other nodes in the cluster on the indicated interface.

```
Starting udp discovery for interface 192.168.13.163
```

> The machine has started listening for heartbeat packets.

```
Starting udp multicast discovery for interface 192.168.13.163
```

> The machine has started listening for heartbeat packets.

```
Discovered cell 8 (cxfsxe7) [priority 1 at 192.168.13.101 via
192.168.13.47] 1 of 2 interfaces up
```

> A connection is being established between this node and node `cxfsxe7`; the local and remote network addresses used for this connection are shown as well.

`cxfs-config` Messages

The following are common error and warning messages displayed by `cxfs-client`. For more information, see "Validating the Cluster Configuration with `cxfs-config`" on page 370.

enabled machine *hostname* has no NICs matching any net

> There are failover networks defined, but the network interfaces for *hostname* are not members of the defined networks. See:
>
> - "Network Failover Modification Tasks with `cxfs_admin`" on page 286
> - "I/O Fencing" on page 58

one or more machines have fencing enabled, but no switches are defined

> CXFS requires either system reset or I/O fencing for all nodes. Fencing requires a switch. See "Create or Modify a Node with `cxfs_admin`" on page 253.

server *hostname* fail policy must not contain "Shutdown" for cluster
    with even number of enabled servers and no tiebreaker

If there are an even number of server-capable administration nodes 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 25

- "Use a Client-Only Tiebreaker" on page 55

```
tiebreaker must not be set in cluster with exactly two enabled nodes  both server capable
```

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. See "Use a Client-Only Tiebreaker" on page 55.

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 26.

If you see an error message such as the following when trying to shut down a node, it means that the node which is the active metadata server for one or more filesystems and was unable to unmount the filesystems, and therefore was unable to shutdown gracefully:

```
Aug 31 10:06:18 cc-xe kernel: Filesystem "xvm-2": relocation from this host is disabled
```

Also see "Shutdown Failure" on page 434.

Shutdown Failure

If a node is unable to shutdown gracefully, it may be because it was unable to unmount the filesystems for which it is the active metadata server. See "Relocation Error" on page 434.

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, SAS, or InfiniBand switch is misconfigured:

```
controller disable is not supported on loop
```

CXFS fencing recovery operations do not support loop mode. Verify that all 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 <0:0:0:0:c000000000000000:0:0:704> 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. `704` is a binary bitmask of excluded cells.

```
CMS calculation limited to last membership:configuration change incomplete on cells <0:0:0:0:0:0:0:704>
```

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). `704` 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.

```
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.

```
Cell XXX is tardy
```

Generated when cell *XXX* has not reported in to the cluster membership leader during a membership change, resulting in a timeout. This situation will cause the node to be removed from the cluster membership and can result in a shutdown of the entire cluster if quorum is lost.

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/log/cxfs/clconfd_*nodename*) for error messages. For information about licensing error messages, see "License Key Error" on page 437.

You must properly install the license keys before you can use CXFS. See Chapter 5, "CXFS Licensing" on page 113.

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 84.

Daemons could also be leaking memory in this case. You may need to restart them:

* On server-capable administration nodes:

  server-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 210 or "Enable a Node with cxfs_admin" on page 263.

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/log/cxfs/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:

```
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.

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
```

Message similar to the following will appear in the client-log file:

- Successful:

    – Server license key granted, regardless of local client license key:

```
date CXFS_Client: cis_license_apply successfully reapplied for server-based license
date CXFS_Client: cis_license_apply allocated 1 "license_type" license(s).
```

- Unsuccessful (CXFS will not start):

    – Server denies a license key, regardless of local license key presence:

    ```
    date CXFS_Client: cis_license_apply ERROR: No license available
    ```

On a server-capable administration node, the error will appear in the `clconfd` log.

You must properly install the license key before you can use CXFS. See Chapter 5, "CXFS Licensing" on page 113.

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. 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 this problem, make the cluster ID numbers unique. First try the steps in "Eliminate a Residual Cluster" on page 395. If that does not work, reboot the nodes that have stale information. Stale nodes will complain about all of the nodes, but the up-to-date nodes will complain only about the stale nodes. The `/var/log/cxfs/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 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/comm/clconfd-ipc_cxfs0
```

Table 15-2 breaks down 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/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:

- "General System Log File Messages" on page 441
- "cli System Log File Error Messages" on page 442
- "clconfd System Log File Error Messages" on page 443
- "crsd System Log File Error Messages" on page 447

General System Log File 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 468.

```
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.

```
CI_CRFERR_NOLCK, failed to raise fence on cell 0
```

> This message might indicate that CXFS cannot log in to the Fibre Channel, SAS, or InfiniBand switch. This could occur if the switch operation has slowed due to a status change. You should examine the following:
>
> - The SAN fabric
>
> - The HBA SAN connections
>
> - The Fibre Channel, SAS, or InfiniBand switch
>
> You may need to reboot the switch to ensure the stability of CXFS.
>
> This message could also indicate that an existing telnet session has hung on the switch. The fix to this situation, see "Brocade telnet Session is Hung" on page 426 or reboot the switch.

```
sw core-fc, port 95, host mds02 :  still transient, 292s until
timeout
```

> This message indicates that the port status is temporary (neither enabled nor disabled). This could be due to an unused port or a

> problem with the switch operation, either running too slowly or with too much traffic.

`cli` System Log File 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 468.

`clconfd` System Log File 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 exists.
>
> If you get an error or the dump is empty, re-create the database; for more information, see "Clearing the Cluster Database" on page 464. If the database exists, restart the cluster daemons; see "Stopping and Restarting Cluster Administration Daemons" on page 468.

`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_ERR_NOMEM, get_fsinfo:  out of space:  got 2096184, need 2845224`

> The maximum number of filesystems defined in CXFS has been exceeded. CXFS has a static limit that depends upon a number of factors, including the number of nodes in the cluster.

`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 468.

`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 188, or "Create or Modify a Node with `cxfs_admin`" on page 253.

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 468.
>
>   If the problem persists, clear the database, reboot, and re-create the
>   database. See "Clearing the Cluster Database" on page 464.

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/log/cxfs/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 469.

CI_CRFERR_NOLCK, failed to lock switch for fencing update

> The CXFS I/O fencing facility is unable to lock access to one or more Fibre Channel, SAS, or InfiniteBand switches, perhaps due to defunct `admin` login processes. See "CXFS Cannot Access the Switch" on page 425.

CI_FAILURE, CXFS filesystem (/dev/cxvm/tp95) on (/cxfs/tp95) already mounted with different options

> A server-capable administration node was added to or deleted from the cluster while the filesystems for which it is a potential metadata server remained mounted, causing confusion. See "Unmount Filesystems Before Adding or Deleting Server-Capable Administration Nodes" on page 55.

`crsd` System Log File 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 468.

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 138.

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 138.

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 422.

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 188 or "Create or Modify a Node with cxfs_admin" on page 253.

cmond System Log File 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:

   server-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 468.

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 System Log File 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` **System Log File 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 468. If the problem persists, clear the database, reboot, and re-create the database. See "Clearing the Cluster Database" on page 464.

```
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 468. If the problem persists, clear the database, reboot, and re-create the database. See "Clearing the Cluster Database" on page 464.

```
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 468. If the problem persists, clear the database, reboot, and re-create the database. See "Clearing the Cluster Database" on page 464.

```
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 468. If the problem persists, clear the database, reboot, and re-create the database. See "Clearing the Cluster Database" on page 464.

```
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.

Daemon Log File Errors

CXFS maintains logs for each of the CXFS daemons. For information about customizing these logs, see "Set Log Configuration with the GUI" on page 207.

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 452, 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.

This section discusses the following:

- "`cad` Log File Error Messages" on page 452
- "`cli` Log File Error Messages" on page 454
- "`crsd` Log File Error Messages" on page 455
- "`fs2d` Log File Error Messages" on page 456

`cad` Log File Error Messages

The following are examples of messages from `/var/log/cxfs/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 468.

`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 468.

`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 468.

```
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` Log File Error Messages

The following are examples of messages from `/var/log/cxfs/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/comm/clconfd-ipc_cxfs0`

> The local node (`cxfs0`) is not defined in the cluster database.

```
CI_IPCERR_NOSERVER, Connection file
/var/cluster/comm/clconfd-ipc_cxfs0 not present.
```

> The local node (cxfs0) is not defined in the cluster database.

crsd Log File Error Messages

The following are examples of messages from /var/log/cxfs/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 the 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 node cxfs0 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` Log File Error Messages

The following are examples of messages from `/var/log/cxfs/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) configuration. For more information, see the `rpcinfo`(8) and `rpcbind`(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` Errors

```
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 connected to the same public network, use the -i option to identify the cluster name. See "Accessing the Correct Cluster at a Multiple-Cluster Site" on page 243.

```
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.

```
transient bootstrap state for cluster clustername detected
```

This message usually indicates that there are two clusters with the same name running on the network. You must rename one of the clusters.

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
```

Authorization Errors

If a node is not allowed to be configured automatically by the `cxfs_admin` command, errors such as the following will appear in the `cxfs_client` log:

```
Aug 06 10:36:51 cxfs_client: cis_cdb_go ERROR: Error returned from server: authorization
error (8)
Aug 06 10:36:51 cxfs_client: cis_autoconf client is not in an autoconf "allowed" policy
```

For more information about automatic configuration, see "Automatically Configure a Client-Only Node" on page 250.

The `cxfs_admin` command will also display errors if the node running `cxfs_admin` does not have `admin` or `monitor` access to the cluster. See "Setting `cxfs_admin` Access Permissions" on page 241.

Connection Error

The following message indicates that the system is unable to open UDP socket connection to another system (in this case, cell ID 4) because the connection attempt is refused:

```
mesg_connect:cell 4:channel 0:transport 2:connect error -111
```

Frequently repeated occurrences of this message may indicate an incorrect network configuration or may require reinstalling the CXFS software on either the CXFS client-only node or the server-capable administration node. This message may also occur occasionally when a node is started or shut down.

`remote version is too old` Error

The following error is generated when the cluster contains nodes running incompatible versions of CXFS software, such as cluster with nodes running CXFS 7.0 (ISSP 3.0) and CXFS 6.6 (ISSP 2.6):

```
[  552.680802] Cell 3 (pg-5) transport [from 192.168.14.163:5450 to
192.168.14.54:5450] connection refused: remote version is too old.
```

`node is downrev` Error

All server-capable administration nodes in the cluster must run the same version of CXFS except during a rolling upgrade. The following types of errors indicate that the server-capable administration nodes are running different versions of CXFS software:

```
CXFS cell C (name) mount refused: node is downrev (tag X, has N, need M)
```

```
CXFS cell C (name) relocation refused: node is downrev (tag X, has N, need M)
```

```
Cell C (nodename) is downrev and will not be able to connect
```

An operation may be forbidden because of a version mismatch between the metadata server and the other node involved. The following operations may fail:

- An attempt by a client-only node running a newer version (such as CXFS 5.5) to mount a filesystem served by a metadata server running an older version (such as CXFS 5.4)

- An attempt to relocate a filesystem to between nodes with different versions (for example, from a CXFS 5.4 metadata server to a CXFS 5.5 server-capable administration node)

For example, the following message indicates that a client-only node tried to mount a filesystem served by a metadata server that is running an older version of CXFS:

```
CXFS cell 0 (xo-xe1) mount refused: node is downrev (tag 30, has -1, need 0)
```

In another example, the following message that appears on a client-only node (with a cell ID of 5) indicates that server-capable administration node cxfsxe6 (with a cell ID of 9) is running an older CXFS release:

```
Cell 9 (cxfsxe6) is downrev and will not be able to connect
```

In this case, the following message will also appear at 1–second intervals on cxfsxe6:

```
mesg_connect:cell 5:channel 0:transport 2:connect error -111
```

This combination of messages indicates that the client-only node is running a version of CXFS from ISSP 2.*x* and cxfsxe6 is a server-capable administration node running a later version of CXFS from ISSP 1.*x*, which is an unsupported configuration.

During a rolling upgrade, relocation of a filesystem between server-capable administration nodes running different versions of CXFS may be disabled. Recovery rather than relocation is used to move filesystems to an upgraded server-capable node. See "CXFS Release Versions and Rolling Upgrades" on page 301.

EXTENT Errors

If you see messages like the following on the console or in the system log file, turn off CXFS extents deltas:

```
EXTENT: a8000002c9e60600 bno 10 not at boundary (got = 9+5)
```

This is best done on the metadata server by setting the fs.cxfs.cxfs_extents_delta system tunable parameter to 0. A reboot is not required. For more information, see "cxfs_extents_delta" on page 504.

GRIOv2 Errors

If you see errors like the following, it means that the server-capable administration node was not rebooted after installing GRIOv2:

```
# /usr/sbin/grioadmin -sv
print_streams: error: Cannot get GRIO server info: Resource temporarily unavailable
grioadmin: print_streams: error: grio_get_streams: Resource temporarily unavailable
```

foswitch Errors

The following are examples of errors that can be produced by the xvm foswitch command, which changes the path used to access a physical disk:

```
mds1-hd kernel: foswitch error i/o error on cell 19
```

The test I/O on cell 19 failed when trying the new path.

```
mds1-hd kernel: foswitch error failover: no device of specified affinity on cell 9
```

There is no a path with the desired affinity on cell 9. For more information, see the failover2.conf file on the node with the specified cell ID.

```
mds1-hd kernel: foswitch error failover: potentially recoverable error, try again on cell 0
```

An foswitch command was attempted while a failover was happening.

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

Leaving Transient State Errors

Multiple messages of the following type indicate that the port on the switch is not fenced but the Fiber Channel connection with the node is gone:

```
<I0 clconfd clconf 25808:10 clconfd_fence.c:245> sw
msdsan2, port 5, host dpox01 : leaving transient state (timeout => no
connection), clearing cdb entries and lowering fences on this port
```

You should check the connection between the nodes and the switches, and verify that the switches are running properly.

CXFS Cluster Admin Shutdown:failed Message

The following error message when trying to stop the CXFS filesystem service (cxfs) may indicate that filesystems cannot be unmounted:

```
# /etc/init.d/cxfs stop
Stopping CXFS cluster services...
Stopping clconfd:
                                                                 done
CXFS Cluster Admin Shutdown:
CXFS Cluster Admin Shutdown failed
                                                                 failed
/etc/init.d/cxfs: line 333:  2200 Killed          $CAP_CXFS "${BIN}/cxfs_shutdown -f -r 0" > /dev/null 2>&1

MDS:
...
```

This can be the result if the node has already been removed from the cluster (despite the shutdown failure message).

alive Message Errors

The following message indicates that a CXFS node on the CXFS private network is running an older, incompatible version of CXFS software:

```
mtcp ignoring improperly sized alive message from node_IP_address:nnnnn (got 16 want 72)
```

You should disable or upgrade the CXFS software software on the indicated node. If the node belongs to a different cluster, the private networks for the two clusters should be segregated. Running multiple clusters on a shared private network is not a supported configuration.

This message may also occur if there is more than one cluster using the public network as a backup private network. This is a valid configuration. In this case the message frequency can be modified or the messages disabled using the mtcp_hb_warn_period system tunable parameter. See "mtcp_hb_warn_period" on page 498.

Remote Clients Error

The following error indicates that the XVM server is waiting for a reply from the client with the identified cell number, but the client has not responded:

mds `kernel: remote_clients_verify: cell` *NN* `timed out`

XVM operations in the cluster will be blocked until the client responds. If you seen the message, review the client's system log for I/O errors and try to execute the following command on the client:

`client#` **`xvm show -v phys`**

Most likely, this `xvm` command will hang. If so, trigger an NMI or otherwise panic the system in order to generate a kernel dump.

`unable to lock bootconfig` Error

It is normal for the following kernel log message to appear occasionally on CXFS server-capable administration nodes, where *X* and *Y* are replaced by the cell IDs of the actual nodes involved:

`XVM client` *X* `unable to lock bootconfig, held by cell` *Y*

However, a problem is indicated when the message appears continually. To resolve the problem, reset cell *Y*.

Corrective Actions

This section covers the following corrective actions:

- "Restarting CXFS Services" on page 464

- "Clearing the Cluster Database" on page 464

- "Rebooting" on page 465

- "Rebooting without Rejoining the Cluster" on page 466

- "Recovering a Cluster with Two Server-Capable Administration Nodes" on page 466

- "Stopping and Restarting Cluster Administration Daemons" on page 468

- "Recreating the Cluster Database" on page 468
- "Verifying Connectivity in a Multicast Environment" on page 469
- "Power-Cycling a Node" on page 471
- "Resetting a Node" on page 471
- "Starting CXFS without Mounting Filesystems after an Abrupt Power Outage" on page 471
- "Using SGI Knowledgebase" on page 472

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 database 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 206
- "Enable a Node with cxfs_admin" on page 263

Clearing the Cluster Database

To clear the cluster database on all of the server-capable administration nodes, 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:

   ```
   server-admin# service cxfs stop
   ```

2. Enter the following on all server-capable administration nodes:

```
server-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:

```
server-admin# /usr/cluster/bin/cdbreinit
```

See also "Reboot Before Changing Node ID or Cluster ID" on page 84.

4. Enter the following on all server-capable administration nodes:

```
server-admin# service cxfs_cluster start
```

5. Enter the following on all server-capable administration nodes:

```
server-admin# service cxfs start
```

See "Eliminate a Residual Cluster" on page 395 to get rid of possible stale cluster configuration in the kernel. If needed, reboot the nodes.

Rebooting

The following are situations that may require 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

Enter the following individually on every node to reboot the cluster (other than Windows, which uses a different reboot mechanism):

```
# reboot
```

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 206

- "Enable a Node with `cxfs_admin`" on page 263

For information about client-only nodes, see the *CXFS 7 Client-Only Guide for SGI InfiniteStorage*.

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:

```
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 315.

Recovering a Cluster with Two Server-Capable Administration Nodes

There are specific issues with recovering a cluster with two server-capable administration nodes. Suppose the following sequence of events:

1. You have cluster named `clusterA` that has two server-capable administration nodes, any number of client-only nodes, and 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.

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 207

    - "Create a Cluster with cxfs_admin" on page 271

2. Revoke the CXFS kernel membership of node2. See:

    - "Revoke Membership of the Local Node with the GUI" on page 210

    - "Disable a Node with cxfs_admin" on page 263

3. Allow CXFS kernel membership of node2. See:

    - "Allow Membership of the Local Node with the GUI" on page 210

    - "Enable a Node with cxfs_admin" on page 263

4. Unset the CXFS tiebreaker node capability. See:

    - "Set Tiebreaker Node with the GUI" on page 207

    - "Create or Modify a Node with cxfs_admin" on page 253

⚠ **Caution:** All two-server-capable administration node clusters without a tiebreaker set must have fencing or reset configured. SGI recommends reset.

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.

Stopping and Restarting Cluster Administration Daemons

⚠ **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.

The commands to stop and restart cluster administration daemons depend upon the platform.

To stop and restart cluster administration daemons, enter the following:

• On server-capable administration nodes:

```
server-admin# service cxfs_cluster restart
```

• On client-only nodes:

```
client# killall cxfs_client
client# cxfs_client start
```

These commands affect the cluster administration daemons only.

See also "Restarting CXFS Services" on page 464. For general information about the daemons, see "Kernel Threads" on page 475.

Recreating the Cluster Database

See Chapter 13, "Cluster Database Management" on page 365. If a node will not join cluster membership, you must reboot it.

Verifying Connectivity in a Multicast Environment

Verification of multicast connectivity requires multicast `ping` on each node in the cluster (other than Windows nodes).

On Linux nodes, you must enable multicast `ping` by using one of the following methods (the permanent method will not take affect until after a reboot):

- Immediate but temporary method:

  ```
  server-admin# echo "0" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
  ```

  For more information, see http://kerneltrap.org/node/16225

- Immediate but temporary method:

  ```
  server-admin# sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=0"
  ```

- Permanent method upon reboot (survives across reboots):

  1. Remove the following line (if it exists) from the /etc/sysctl.conf file:

     net.ipv4.icmp_echo_ignore_broadcasts = 1

  2. Add the following line to the /etc/sysctl.conf file:

     net.ipv4.icmp_echo_ignore_broadcasts = 0

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 126.

Power-Cycling a Node

To power-cycle a node, see "Control and Contact a Node with cxfs_admin" on page 267.

Resetting a Node

To reset a node, see "Control and Contact a Node with cxfs_admin" on page 267.

Starting CXFS without Mounting Filesystems after an Abrupt Power Outage

In the case of an abrupt power outage, you can bring up CXFS without mounting the previously mounted filesystems by doing the following:

1. Boot each server-capable administration node to single-user mode.

2. Execute the following on each server-capable administration node, to ensure that CXFS will not be started upon reboot:

   # **chkconfig cxfs off**

3. Boot each server-capable administration node to multi-user mode.

4. On **one** of the server-capable administration nodes, use the cxfs_admin command with the unmount operation to unmount all CXFS filesystems.

   For example, to unmount filesystem myfs from nodes node1, node2, and node3:

   cxfs_admin:mycluster> **unmount myfs nodes=node1,node2,node3**

   See "Unmount a CXFS Filesystem with cxfs_admin" on page 283.

5. Execute the following on each server-capable administration node, to ensure that CXFS will be started upon reboot:

   # **chkconfig cxfs on**

6. Execute the following on each server-capable administration node, to start CXFS:

   # **service cxfs start**

Using SGI Knowledgebase

If you encounter problems and have an SGI support contract, you can log on to Supportfolio and access the Knowledgebase tool to help find answers.

To log in to Supportfolio Online, see:

https://support.sgi.com/login

Then click on **Search the SGI Knowledgebase** and select the type of search you want to perform.

If you need further assistance, contact SGI Support.

Reporting Problems to SGI

Before reporting a problem to SGI, you should retain the information by using the cxfsdump(8) utility. See "Cluster Information Gathering Tool (cxfsdump)" on page 407.

Also do the following:

- Retain any messages that appeared in the system logs immediately before the system exhibited the problem.

- After a system kernel panic, retain the debugger information from the KDB built-in kernel debugger. See "System Dump Analysis Tool" on page 404.

- 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" on page 475
- "Communication Paths" on page 477
- "Flow of Metadata for Reads and Writes" on page 481

Also see the following:

- "Cluster Administration Daemons" on page 43
- "CXFS Control Daemons" on page 44
- *CXFS 7 Client-Only Guide for SGI InfiniteStorage*

Kernel Threads

Table A-1 on page 476, discusses kernel threads. CXFS shares with XFS the Linux `xfsbufd` and `xfsdatad` kernel threads to push buffered writes to disk.

Note: In the `ps` command output, the thread names begin with a `*` character, such as [`*mtcp_notify`]).

Table A-1 Kernel Threads

Kernel Thread	Description
cmsd	Manages CXFS kernel membership and CXFS kernel heartbeating.
Recovery	Manages recovery protocol for node.
corpse leader	Coordinates recovery between nodes.
dcshake	Purges idle CXFS vnodes on the CXFS client.
cxfsd	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	Reads messages (one per open message channel).
mtcp_notify	Accepts new connections.
mtcp_discovery	Monitors and discovers other nodes.
mtcp_xmit	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.
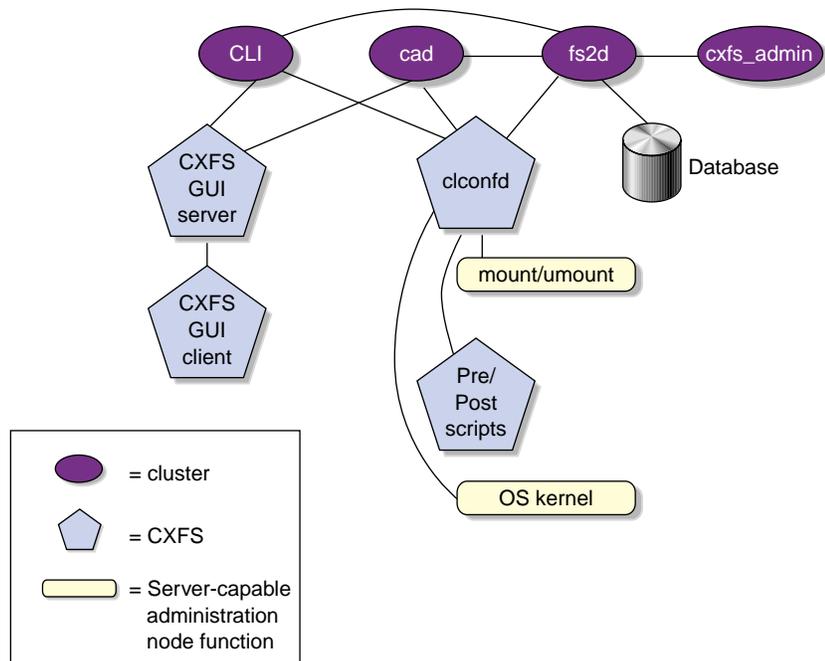
Figure A-1 Communication Within One Server-Capable Administration Node
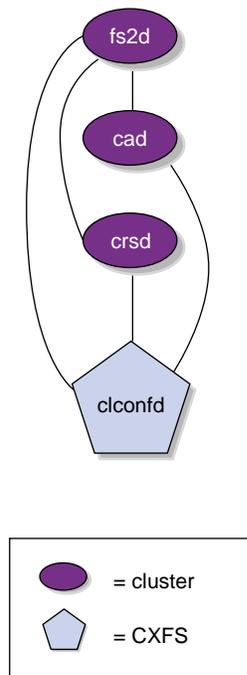
Figure A-2 Daemon Communication Within One Server-Capable Administration Node
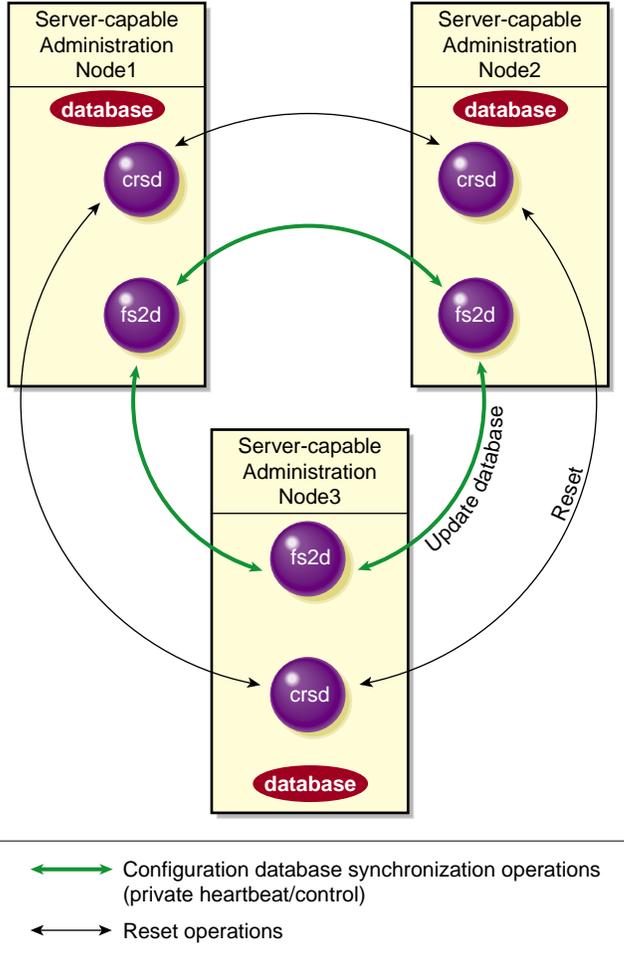
Figure A-3 Communication Among Nodes in the Pool
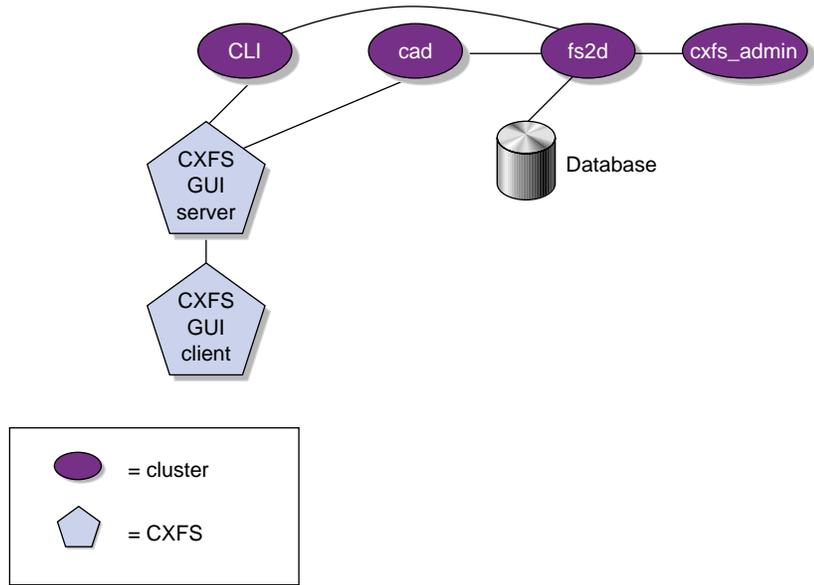
Figure A-4 Communication for a Server-Capable Administration Node Not in a Cluster

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. Figure A-5describes the communication among nodes, showing just a single client-only node as an example.
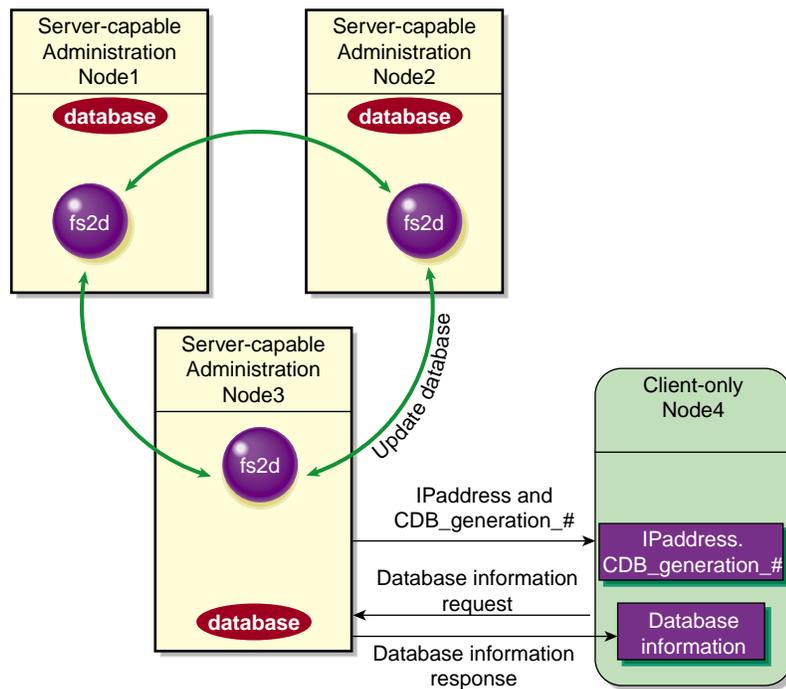
Figure A-5 Communication Among Nodes

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.
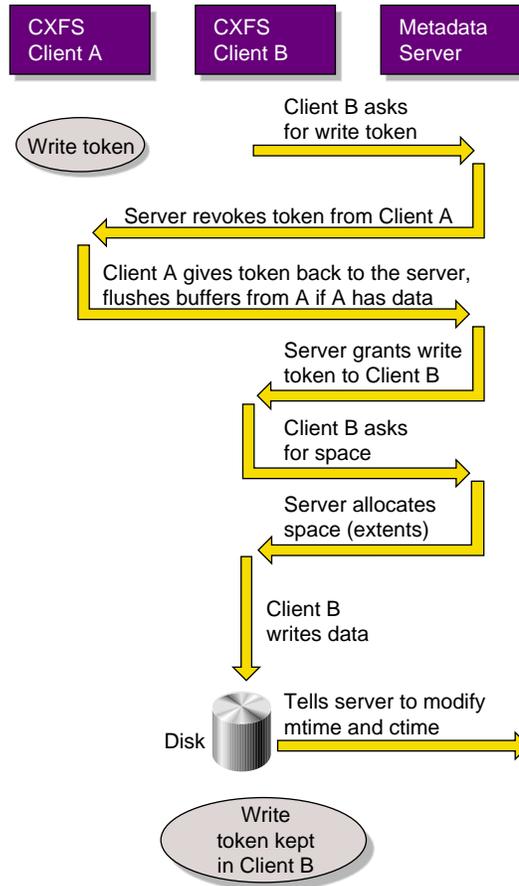
Figure A-6 Metadata Flow on a Write
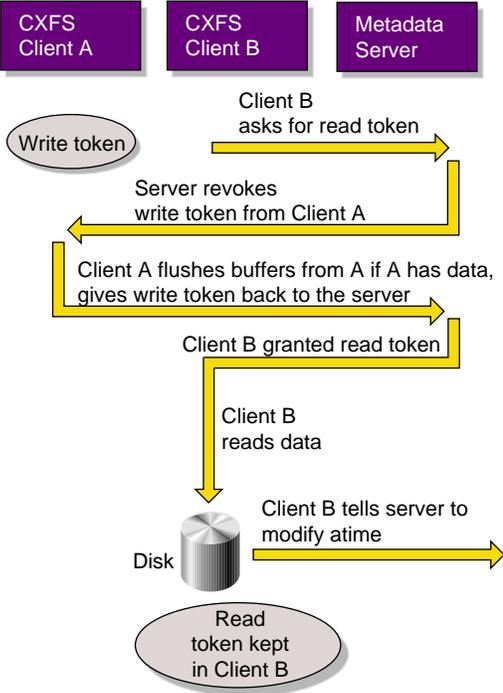
Figure A-7 Metadata Flow on a Read on Client B Following a Write on Client A
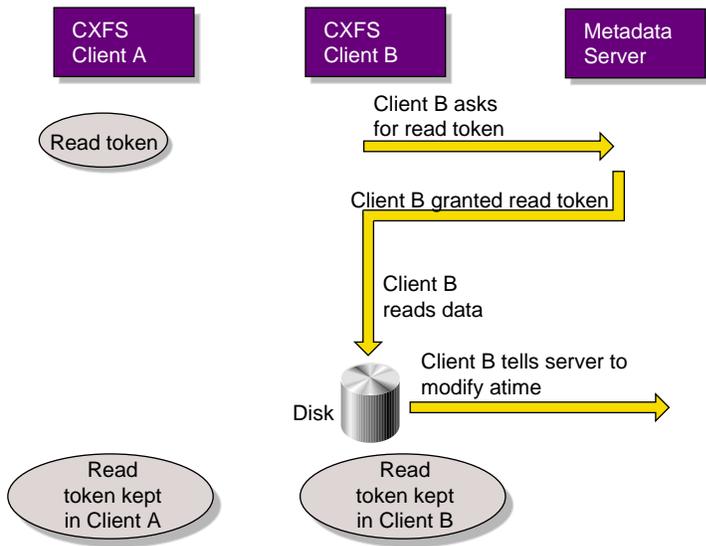
Figure A-8 Metadata Flow on a Read on Client B Following a Read on Client A

Path Summary

This appendix lists the locations for commonly used commands on server-capable administration nodes.

Table B-1 Paths for Server-Capable Administration Nodes

Command/File	Path
cdbreinit	/usr/cluster/bin/cdbreinit
chkconfig	/bin/chkconfig
cxfs_admin	/usr/cluster/bin/cxfs_admin
df	/bin/df
grioadmin	/usr/sbin/grioadmin
griomon	/usr/sbin/griomon
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 configuration information	/usr/cluster/bin/clconf_info
CXFS/cluster daemon initialization	/etc/init.d/cxfs_cluster
CXFS license verification command:	/usr/cluster/bin/cxfslicense

BMC System Controller

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.

SGI x86_64 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 SGI x86_64 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
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
```

For more information, see:

- ipmitool(1) man page

- The user guide or quick start guide for your system

CXFS System Tunable Kernel Parameters

This appendix discusses the following:

- "Overview of the CXFS System Tunable Kernel Parameters" on page 493
- "Site-Configurable Parameters" on page 497
- "Debugging Parameters" on page 507

The parameters described in this appendix apply to CXFS server-capable administration nodes and client-only nodes running Linux. For more information about system tunable parameters on client-only nodes, see *CXFS 7 Client-Only Guide for SGI InfiniteStorage.*

Overview of the CXFS System Tunable Kernel Parameters

This section discusses the following:

- "Using Appropriate Parameter Settings" on page 493
- "Interpretations of Bit Values for Standard and Debug Kernels" on page 494
- "Making Permanent Parameter Changes" on page 495
- "Making Temporary Parameter Changes" on page 496
- "Querying a Current Parameter Setting" on page 496

Using Appropriate Parameter Settings

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. You should change debugging parameters only at the recommendation of SGI Support.

The values of these parameters vary in different releases of the product. When upgrading the product, consult SGI Support to determine whether any changes made to the parameters in this chapter should be carried forward. Setting these parameters incorrectly may render the system unstable or otherwise unusable.

See also "Set System-Tunable Kernel Parameters Appropriately" on page 87.

Interpretations of Bit Values for Standard and Debug Kernels

A number of parameters have a value that is interpreted in the same manner: the least significant 4 bits are used in standard (nondebug) kernels, the next 4 bits are used in debug kernels. 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 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

A common default (0xf5) is to always set the enable bit, to print only some messages in the nondebug kernel case, and print all messages and panic in the debug kernel case.

Figure D-1 shows an example.
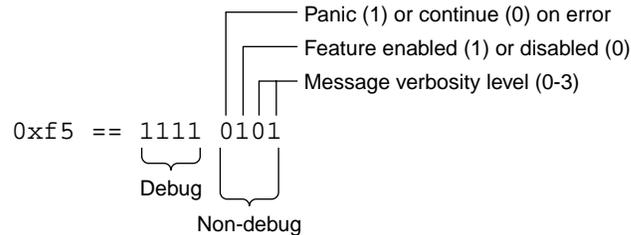
Figure D-1 Explanation of Bit Values

Making Permanent Parameter Changes

You can set a parameter permanently across reboots on a SLES system by adding it to the `/etc/modprobe.d/sgi-cxfs-xvm.conf` file. Use the following format:

options *module* *systune*=*value*

where:

- *module* is one of the following:

  ```
  sgi-cxfs
  sgi-cell
  ```

  The section that describes a parameter lists the module name.

- *systune* is the parameter name, such as `mtcp_hb_watchdog`

- *value* is the value you want to set for the parameter, such as 2

Note: Do not use spaces around the = character.

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.

For example, to permanently set the `mtcp_hb_watchdog` parameter (which is in the `sgi-cell` module) to 2, add the following line to `/etc/modprobe.d/sgi-cxfs-xvm.conf`:

options sgi-cell mtcp_hb_watchdog=2

The change will take effect upon reboot.

Making Temporary Parameter Changes

For a temporary change to a dynamic parameter, use the Linux sysctl(8) command as follows:

sysctl *prefix.systune=value*

where:

- *prefix* is one of the following:

  ```
  fs.cxfs
  kernel.cell
  ```

- *systune* is the parameter name, such as mtcp_hb_watchdog

- *value* is the value you want to set for the parameter, such as 2

Note: Do not use spaces around the = character.

For example, to temporarily set the mtcp_hb_watchdog parameter (which has the kernel.cell prefix) to 2, enter the following:

```
# sysctl kernel.cell.mtcp_hb_watchdog=2
kernel.cell.mtcp_hb_watchdog = 2
```

Querying a Current Parameter Setting

To query the current setting of a parameter, use the Linux sysctl(8) command:

sysctl *prefix.systune*

where:

- *prefix* is one of the following:

  ```
  fs.cxfs
  kernel.cell
  ```

- *systune* is the parameter name, such as mtcp_hb_watchdog

For example, to query the current setting of the `mtcp_hb_watchdog` parameter (which has the `kernel-cell` prefix):

```
# sysctl kernel.cell.mtcp_hb_watchdog
kernel.cell.mtcp_hb_watchdog = 2
```

Site-Configurable Parameters

This section discusses site-configurable parameters:

- "Static Parameters that are Site-Configurable" on page 497
- "Dynamic Parameters that are Site-Configurable" on page 502

Static Parameters that are Site-Configurable

Static parameters require a reboot to take affect.

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.

Range of 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.

Prefix: `kernel.cell`

Module: `sgi-cell`

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 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 Systems" on page 67.

Prefix: `kernel.cell`

Module: `sgi-cell`

mtcp_hb_warn_period

Specifies the minimum interval between warnings about heartbeat size, in seconds. A setting of `0` suppresses the warnings. You should only modify this parameter setting if there is more than one cluster using the public network as a backup private network.

Range of values:

- Default: `60`

- Minimum: `0`

- Maximum: `604800`

Prefix: `kernel.cell`

Module: `sgi-cell`

Also see "`alive` Message Errors" on page 462.

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` disables the watchdog (default)

- `1` specifies that watchdog expiration causes CXFS shutdown

- `2` specifies that watchdog expiration causes panic

Prefix: `kernel.cell`

Module: `sgi-cell`

mtcp_nodelay

> Specifies whether to enable or disable `TCP_NODELAY` on CXFS message channels.
>
> Range of values:
>
> - `0` disables
>
> - `1` enables (default)
>
> Prefix: `kernel.cell`
>
> Module `sgi-cell`

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
>
> Prefix: `kernel.cell`
>
> Module: `sgi-cell`

rhelpd_aux

> Specifies the maximum number of auxiliary `rhelpd` threads to run. (The `rhelpd` threads help out recovery and relocation tasks. They can be used for activities such as asynchronous inode reconstruction and parallel recoveries. The `rhelpd` thread pool is global in nature and is created during module load time.)
>
> The system automatically uses a calculated value that is four times the number of CPUs, so long as that the value is below `128`. That is:
>
> 4 * *number_of_CPUs* = *rhelpd_max_value*
> *If:*
> 0 <= *rhelpd_max_value* <= 128

Range of values:

- Default: 0, which specifies an automatically calculated value (to disable automatic calculation, set rhelpd_aux to a non-zero value)

- Minimum: 0

- Maximum: 128

Prefix: fs.cxfs

Module: sgi-cxfs

rhelpd_max

Specifies the maximum number of rhelpd threads to run. The system automatically uses a calculated value. For more information about rhelpd threads and the calculated value, see "rhelpd_aux" on page 500.

Range of values:

- Default: 0, which specifies an automatically calculated value (to disable automatic calculation, set rhelpd_max to a non-zero value)

- Minimum: 0

- Maximum: 128

Prefix: fs.cxfs

Module: sgi-cxfs

rhelpd_min

Specifies the minimum number of rhelpd threads to run. The system automatically uses a calculated value. For more information about rhelpd threads and the calculated value, see "rhelpd_aux" on page 500.

Range of values:

- Default: 0, which specifies an automatically calculated value (to disable automatic calculation, set rhelpd_min to a non-zero value)

- Minimum: 0

- Maximum: 8 (when the value is set explicitly)

Prefix: `fs.cxfs`

Module: `sgi-cxfs`

Dynamic Parameters that are Site-Configurable

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

Prefix: `kernel.cell`

Module: `sgi-cell`

`cxfs_client_push_period`

Specifies (in hundredths of a second) the length of time 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 505.

Range of values:

- Default: `25`

- Minimum: `0`

- Maximum: `1000`

Prefix: `fs.cxfs`

Module: `sgi-cxfs`

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

Prefix: fs.cxfs

Module: sgi-cxfs

cxfs_token_fault_tolerant

Specifies that CXFS should tolerate certain recoverable errors in the token subsystem. See "Interpretations of Bit Values for Standard and Debug Kernels" on page 494.

Range of values:

- Default: 0xf5

- Minimum: 0

- Maximum: 0xff

Prefix: fs.cxfs

Module: sgi-cxfs

cxfs_verify_existence_token

Verifies that a client has the existence token before trying to obtain additional tokens. See "Interpretations of Bit Values for Standard and Debug Kernels" on page 494.

Range of values:

- Default: 0xf5

- Minimum: 0

- Maximum: 0xff

Prefix: `fs.cxfs`

Module: `sgi-cxfs`

cxfs_validate_objid

On a server-capable administration node, checks that an `objid` received from a client corresponds to an object of the expected type. On a client-only node, verifies the level of reporting on receipt of an `EBADOBJID` error from the server. See "Interpretations of Bit Values for Standard and Debug Kernels" on page 494.

Range of values:

- Default: `0xf5`
- Minimum: `0`
- Maximum: `0xff`

Prefix: `fs.cxfs`

Module: `sgi-cxfs`

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)

Prefix: `fs.cxfs`

Module: `sgi-cxfs`

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

Prefix: `fs.cxfs`

Module: `sgi-cxfs`

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. See:

- "Node Types, Node Functions, and the Cluster Database" on page 12
- "Relocation" on page 26

Prefix: `fs.cxfs`

Module: `sgi-cxfs`

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 502.

Range of values:

- Default: `25`
- Minimum: `0`
- Maximum: `1000`

Prefix: `fs.cxfs`

Module: `sgi-cxfs`

cxfsd_aux

Specifies the maximum number of auxiliary `cxfsd` threads to run per CXFS filesystem.

Range of values:

- Default: `0` (calculates value 2 * *number_of_cpus*, but with a minimum of 4 and a maximum of 64)
- Minimum: `0`
- Maximum: `2048`

Prefix: `fs.cxfs`

Module: `sgi-cxfs`

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`.

Prefix: `fs.cxfs`

Module: `sgi-cxfs`

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`

Prefix: `fs.cxfs`

Module: `sgi-cxfs`

Debugging Parameters

This section discusses parameters that should be changed only for debugging purposes and only at the recommendation of SGI:

• "Dynamic Parameters for Debugging Purposes Only" on page 510

• "Static Parameters for Debugging Purposes Only" on page 508

Static Parameters for Debugging Purposes Only

⚠ **Caution:** Debugging parameters are potentially dangerous. You should change them only at the recommendation of SGI Support.

cxfs_disable_splice

Specifies whether or not to disable the Linux `splice` implementation, which is a performance optimization for I/O-intensive applications that allows them to perform zero-copy reads and writes from the Linux page cache. `splice()` The NFS server is an example of an application that is capable of using this interface. A value of `1` disables `splice()`.

Note: The `splice` implementation is required for applications using the `splice`(2), `vmsplicetee`(2), and `sendfile`(2)system calls. Applications using these system calls will fail, in the case of a CXFS filesystem, if `cxfs_disable_splice` system is set to `1`.

Range of values:

- `0` enables `splice`
- `1` disables `splice` (default)

Prefix: `fs.cxfs`

Module: `sgi-cxfs`

cxfs_extents_delta_depth

Specifies the number of changes to the extent list kept by the CXFS metadata server for generating extents deltas.

Range of values:

- Default: `5`
- Minimum: `0`
- Maximum: `32`

Prefix: `fs.cxfs`

Module: `sgi-cxfs`

cxfs_shutdown_time

Specifies the minimum amount of time (in hundredths of a second) that a node is allowed to complete a withdrawal from the cluster on loss of membership. If the metadata server cannot confirm that the node has shut down, it will initiate the node's fail-policy processing to forcefully remove the node from the cluster.

Range of values:

- Default: `50`
- Minimum: `0`
- Maximum: `6000`

Prefix: `kernel.cell`

Module: `sgi-cell`

mesg_delay_time

Specifies (in nsec) the message delay time. By default, there is no delay.

Range of values:

- Default: `0` (no delay)
- Minimum: `0`
- Maximum: `1000000`

Prefix: `kernel.cell`

Module: `sgi-cell`

mtcp_reserve_size

Specifies the TCP window size space reservation for CXFS sockets.

Range of values:

- Default: `61440`

- Minimum: `2048`

- Maximum: `1073741824`

Prefix: `kernel.cell`

Module: `sgi-cell`

Dynamic Parameters for Debugging Purposes Only

> ⚠ **Caution:** Debugging parameters are potentially dangerous. You should change them only at the recommendation of SGI Support.

`cell_tkm_feature_disable`

Disables selected features of the token module by setting a hexadecimal flag bit:

- `0x1` disables speculative token acquisition

- `0x2` (unused)

- `0x4` disables token prefetching

- `0x8` uses multiple RPCs to obtain a token set if the rank and class conflict

- `0x10` disables token lending

- `0x20` disables the blocking of cached tokens

- `0x40` disables range tokens

Note: CXFS token prefetch and range tokens are designed as optimizations for applications using CXFS filesystems on a CXFS client. However, under some workloads, they may cause stability issues and should be disabled, at the direction of SGI Support.

Range of values:

- Default: `0`

- Maximum: `0`

- Minimum: `0x7fff`

Prefix: `kernel.cell`

Module: `sgi-cell`

`cms_fence_timeout`

Specifies the number of seconds that the `cms` daemon will wait for a fence operation to complete. If a fence operation completion message is not received within that time, `cms` takes the action specified by `cms_fence_timeout_action`. A value of `0` is an infinite wait.

Range of values:

- Default: `0`

- Minimum: `0`

- Maximum: `10000`

Prefix: `kernel.cell`

Module: `sgi-cell`

`cms_fence_timeout_action`

Specifies the action that the `cms`daemon should take if the fence operation times out.

⚠️ **Caution:** This parameter is potentially dangerous. You should change it only at the recommendation of SGI Support.

Range of values:

- `0` proceeds as if the fence operation returned an error

- `1` proceeds as if the fence operation succeeded

- `2` panics

Prefix: `kernel.cell`

Module: `sgi-cell`

cms_reset_error_override

Specifies whether or not to ignore reset errors.

⚠ **Caution:** This parameter is potentially dangerous. You should change it only at the recommendation of SGI Support.

Range of values:

- 0 does not ignore reset errors (default)
- 1 ignores reset errors

Prefix: kernel.cell

Module: sgi-cell

cms_trace_enable

Enables or disables tracing for the cms subsystem.

Legal values:

- 0 enables
- 1 disables

Prefix: kernel.cell

Module: sgi-cell

cms_reset_timeout_action

Specifies the action that the cmsdaemon should take if reset times out.

⚠ **Caution:** This parameter is potentially dangerous. You should change it only at the recommendation of SGI Support.

Range of values:

- 0 proceeds as if the reset returned an error
- 1 proceeds as if the reset succeeded

- 2 panic

Prefix: `kernel.cell`

Module: `sgi-cell`

cred_age_max

Specifies the maximum number of entries to store in the `cred-to-credid` cache.

Range of values:

- Default: `1000`

- Minimum: `100`

- Maximum: `10000`

Prefix: `kernel.cell`

Module: `sgi-cell` module)

cred_age_pri

Specifies the priority of the `cred` aging thread.

> ⚠️ **Caution:** This parameter is potentially dangerous. You should change it only at the recommendation of SGI Support.

Range of values:

- Default: `255`

- Minimum: `90`

- Maximum: `255`

Prefix: `kernel.cell`

Module: `sgi-cell`

cred_age_timeout

Specifies how long (in seconds) that an entry stays in the `cred-to-credid` cache.

Range of values:

- Default: `15`
- Minimum: `5`
- Maximum: `3600`

Prefix: `kernel.cell`

Module: `sgi-cell`

cxfs_client_range_age_max

Specifies the maximum age of a granted range, measured in generations, before a client will voluntarily return it.

Range of values:

- Default: `10`
- Minimum: `0`
- Maximum: `1000`

Prefix: `kernel.cell`

Module: `sgi-cell`

See also "`cxfs_server_range_age_max`" on page 517.

cxfs_conversion_delay

During CXFS filesystem relocation/recovery, the data structures for a server-capable administration node must be converted from those for the active metadata server to those for a CXFS client. This parameter specifies the delay (in seconds) before the conversion begins.

⚠️ **Caution:** This parameter is potentially dangerous. You should change it only at the recommendation of SGI Support.

Range of values:

- Default: `0`

- Minimum: `0`

- Maximum: `86400` (24 hours)

Prefix: `kernel.cell`

Module: `sgi-cell`

cxfs_recovery_slowdown

Slows down recovery by inserting delays (measured in milliseconds).

Range of values:

- Default: `0`

- Minimum: `0`

- Maximum: `60000`

Prefix: `kernel.cell`

Module: `sgi-cell`

cxfs_recovery_timeout_panic

Specifies the action taken when a node with stalled recovery is discovered.

Range of values:

- `0` shuts down a node with stalled recovery (default)

- `1` panics a node with stalled recovery

Prefix: `kernel.cell`

Module: `sgi-cell`

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`

Prefix: `kernel.cell`

Module: `sgi-cell`

`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; see "Prevent Stalled-Recovery Timeout in a Non-HA DMF Environment" on page 74)

- Maximum: `3600`

Prefix: `kernel.cell`

Module: `sgi-cell`

`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`

Prefix: `kernel.cell`

Module: `sgi-cell`

cxfs_server_range_age_max

Specifies the maximum age of a granted range, measured in generations, before the server will recall it.

Range of values:

- Default: 10

- Minimum: 0

- Maximum: 1000

Prefix: kernel.cell

Module: sgi-cell

See also "cxfs_client_range_age_max" on page 514.

cxfs_token_track

Verifies compliance with the token and locking hierarchy. The value is a bitmask as described in "Interpretations of Bit Values for Standard and Debug Kernels" on page 494.

Range of values:

- Default: 0x0

- Minimum: 0

- Maximum: 0xff

Prefix: kernel.cell

Module: sgi-cell

cxfsd_sync_force

Specifies the bitmask that indicates whether cxfsd tasks must be run synchronously or asynchronously by threads from a cxfsd thread pool. The bits correspond to the opcodes in cxfsd.h.

Range of values:

- Minimum: 0 (all cxfsd operations asynchronous)

- Maximum `0x7fffffff` (all cxfsd operations synchronous)

Prefix: `fs.cxfs`

Module: `sgi-cxfs`

mesg_ce_min

Specifies the minimum number of threads to use for processing cluster events.

Range of values:

- Default: `0` (specifies that the system will pick a suitable value)
- Minimum: `0`
- Maximum: `100000`

Prefix: `kernel.cell`

Module: `sgi-cell`

mesg_ce_max

Specifies the maximum number of threads to use for processing cluster events.

Range of values:

- Default: `0` (specifies that the system will pick a suitable value)
- Minimum: `0`
- Maximum: `100000`

Prefix: `kernel.cell`

Module: `sgi-cell`

mlb_notify_min

Specifies the minimum number of threads to use for processing general (low-priority) message traffic within the local node.

Range of values:

- Default: `0` (specifies that the system will pick a suitable value)

- Minimum: `0`

- Maximum: `100000`

Prefix: `kernel.cell`

Module: `sgi-cell`

mlb_notify_max

Specifies the maximum number of threads to use for processing priority message traffic within the local node. This ensures that even if `mlb_notify_min` threads are tied up handling general traffic, additional threads can be created to handle priority traffic.

Range of values:

- Default: `0` (specifies that the system will pick a suitable value)

- Minimum: `0`

- Maximum: `100000`

Prefix: `kernel.cell`

Module: `sgi-cell`

mlb_notify_aux

Specifies the maximum number of auxiliary `mlb_notify` threads used for processing messages for which processing can take a long time, for message traffic within the local node.

Range of values:

- Default: `0` (specifies that the system will pick a suitable value)

- Minimum: `0`

- Maximum: `100000`

Prefix: `kernel.cell`

Module: `sgi-cell`

mlb_notify_idle_timeout

Specifies how long a thread within the local node will sleep when it finds no work, to help reduce the rate at which CXFS creates and destroy threads.

Range of values:

- Default: `0` (specifies that the system will pick a suitable value)

- Minimum: `0`

- Maximum: `3600`

Prefix: `kernel.cell`

Module: `sgi-cell`

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

Prefix: `kernel.cell`

Module: `sgi-cell`

mtcp_notify_aux

Specifies the maximum number of auxiliary `mtcp_notify` threads used for processing message traffic between nodes for those messages that can take a long time.

Range of values:

- Default: `0` (specifies that the system will pick a suitable value)

- Minimum: `0`

- Maximum: `100000`

Prefix: `kernel.cell`

Module: `sgi-cell`

mtcp_notify_max

Specifies the maximum number of threads to use for processing priority message traffic between nodes. This ensures that even if `mtcp_notify_min` threads are tied up handling general traffic, additional threads can be created to handle priority traffic.

Range of values:

- Default: `0` (specifies that the system will pick a suitable value)

- Minimum: `0`

- Maximum: `100000`

Prefix: `kernel.cell`

Module: `sgi-cell`

mtcp_notify_min

Specifies the minimum number of threads to use for processing general (low-priority) message traffic between nodes.

Range of values:

- Default: `0` (specifies that the system will pick a suitable value)

- Minimum: `0`

- Maximum: `100000`

Prefix: `kernel.cell`

Module: `sgi-cell`

mtcp_notify_idle_timeout

Specifies how long a thread between nodes will sleep when it finds no work, to help reduce the rate at which CXFS creates and destroy threads.

Range of values:

- Default: `0` (specifies that the system will pick a suitable value)
- Minimum: `0`
- Maximum: `3600`

Prefix: `kernel.cell`

Module: `sgi-cell`

task_age_max

Specifies the maximum number of entries to store in the `task-to-cred` cache.

Range of values:

- Default: `1000`
- Minimum: `100`
- Maximum: `10000`

Prefix: `kernel.cell`

Module: `sgi-cell`

task_age_timeout

Specifies how long (in seconds) that an entry stays in the `task-to-cred` cache.

Range of values:

- Default: `15`

- Minimum: `5`
- Maximum: `3600`

Prefix: `kernel.cell`

Module: `sgi-cell`

Migration from `cmgr` to `cxfs_admin`

The `cmgr` command is no longer supported. 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 295.

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 295.

Note: The `bash` shell interprets `cmgr`-generated scripts, but `cxfs_admin` interprets `cxfs_admin`–generated scripts.

For more information about `cmgr`, see "Appendix F. Reference to `cmgr` Tasks" in *CXFS 5 Administration Guide for SGI InfiniteStorage* (007–4016–030) posted on the SGI Technical Publications Library:

http://techpubs.sgi.com/library/manuals/4000/007-4016-030/sgi_html/apf.html

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 Linux server-capable administration nodes. It discusses the following:

- "Differences Between IRIX and Linux System Administration" on page 527

- "Caveats for Migrating from IRIX" on page 528

- "Migration Procedure" on page 531

- "Migration Troubleshooting" on page 535

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 125. For assistance, contact SGI Installation Services.

Differences Between IRIX and Linux System Administration

If you are migrating from a cluster with IRIX metadata servers to a cluster with Linux 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:

- "Limitations and Considerations for Server-Capable Administration Nodes" on page 131
- Chapter 12, "Administration and Maintenance" on page 299
- Appendix B, "Path Summary" on page 485

Caveats for Migrating from IRIX

This section discusses the following:

- "Changing SGIRDAC Mode to SGIAVT Mode for SGI RAID" on page 528
- "Recreating Filesystems with the Appropriate Naming Version" on page 529
- "Recreating Filesystems with Large Block Sizes" on page 531
- "Using Project Quotas" on page 531

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

On SGI x86_64 server-capable administration nodes, the maximum block size is 4K. If you have filesystems with larger block sizes, 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.

Using Project Quotas

Under Linux, project quotas are applied to the files below a given directory. You can view the project quotas only from the active CXFS metadata server, not from CXFS client-only nodes. See "Quotas Differences" on page 33.

Migration Procedure

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 Linux.

Following is a procedure for using `cxfs_admin` to migrate from a cluster with IRIX server-capable administration nodes to a cluster with Linux server-capable administration nodes. For details about using `cxfs_admin`, see Chapter 11, "`cxfs_admin` Command" on page 233.

Note: The cluster cannot be used for production work during this process.

For information about using the GUI to perform similar tasks, see Chapter 10, "CXFS GUI" on page 167.

1. As a precaution, save the current cluster database contents by using the `build_cmgr_script` command.

2. Ensure you have CXFS 7.0 licenses installed on the Linux server-capable administration nodes. See:

- "Upgrading Licenses" on page 303

- Chapter 5, "CXFS Licensing" on page 113

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 (line breaks here for readability):

```
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
```

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# /etc/init.d/grio2 stop (if running GRIOv2)
   irixadmin# /etc/init.d/cxfs stop
   irixadmin# /etc/init.d/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 317.

      • Remove the IRIX server-capable administration nodes physically from the network.

For more information about using the GUI, see the following:

• "Unmount CXFS Filesystems with the GUI" on page 224

• "Stop CXFS Services with the GUI" on page 206

• "Define a Node with the GUI" on page 188

• "Add or Remove Nodes in the Cluster with the GUI" on page 196

• "Modify a CXFS Filesystem with the GUI" on page 222

• "Add or Remove Nodes in the Cluster with the GUI" on page 196

• "Delete a Node with the GUI" on page 201

• "Start CXFS Services with the GUI" on page 206

• "Mount CXFS Filesystems with the GUI" on page 223

Migration Troubleshooting

The following sections discuss possible problems you may encounter after migrating from an IRIX cluster to a Linux cluster:

- "Filesystems Will Not Mount" on page 535

- "DMF Filesystems Will Not Mount" on page 535

- "Do Not Use extlog or rtfs Filesystems" on page 536

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 3b in "Migration Procedure" on page 531).

> ⚠️ **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 Linux server-capable administration nodes, you cannot use `extlog` or `rtfs` filesystems.

Filesystem Specifications

The following table lists filesystem specifications for Linux running on a server-capable administration node.

Item	Description
Maximum filesystem size	2^{64} bytes
Maximum files size/offset	2^{63}-1 bytes
Filesystem block size (in bytes)[1]	512, 1024, 2048, 4096, 8192, or 16384
XVM device block size (in bytes)	512, or greater than 512 for Linux kernels 2.6.32 or later (XVM on Mac OS X and Windows clients only supports 512 byte block sizes)
Physical LUN limit for DVH-labeled disks	2 TB
Physical LUN limit for GPT-labeled disks[2]	2^{63} device blocks
Maximum concatenated slices	65536

---

[1] If the filesystem is to be accessible by other platforms in a multiOS cluster, its block size must be supported on all platforms in the cluster

[2] Note the following about physical LUN limits for GPT-labeled disks:

- Physical LUNs with GPT labels are not constrained by XVM or CXFS to be smaller than the largest possible filesystem.

- Cluster nodes may constrain the LUN size to be smaller due to driver or other operating system constraints. A LUN used in the cluster may not be larger than the maximum size allowed by any node.

- All nodes that mount a filesystem using LUNs larger than 2 TB must be upgraded to CXFS 4.2 or later.

`mkfs` Options and CXFS

Table H-1 lists those `mkfs` options and arguments that have certain requirements applicable to their use with CXFS or are not supported with CXFS. Arguments that are not listed (such as `-d name`) behave the same way with CXFS as they do with XFS.

Table H-1 `mkfs` Options and CXFS

Option Type	Arguments	Notes
Block size (`-b`)	`log`	Recommended value: 12 Minimum values: 9 for Linux, 12 for Mac OS X.
	`size`	Recommended value: 4096 Minimum values: 512 for Linux, 4096 for Mac OS X.
Data section (`-d`)	`file`	Not supported in CXFS
	`sectlog`	Mac OS X and Windows nodes only support the default of `9`
	`sectsize`	Mac OS X and Windows nodes only support the default of `512`
Inode (`-i`)	`attr`	RHEL 4 does not support a value of `2`
Log section (`-l`)	`sectlog`	Mac OS X and Windows nodes only support the default of `9`
	`sectsize`	Mac OS X and Windows nodes only support the default of `512`
	`lazy-count`	RHEL 4 and RHEL 5 do not support a value of `1`.
Naming (`-n`)	`version`	Only version `2` is supported
Real-time (`-r`)	`extsize`	Not supported
	`size`	Not supported
	`rtdev`	Not supported
	`file`	Not supported
	`name`	Not supported
	`noalign`	Not supported
Sector size	`log`	Mac OS X and Windows only support the default of `9`

Option Type	Arguments	Notes
	sectlog	Mac OS X and Windows only support the default of 9
	size	Mac OS X and Windows only support the default of 512
	sectsize	Mac OS X and Windows only support the default of 512

Deprecated Commands

Table I-1 lists commands whose use has been replaced by cxfs_admin, the CXFS GUI, and cxfs-config.

Table I-1 Deprecated Commands

Command	Description
clconf_status	Provides provides a curses interface to display status information gathered by the cad daemon
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 47 and Chapter 15, "Troubleshooting" on page 393, before attempting to complete these procedures.

Procedure J-1 Initial Configuration Checklist

1. 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.)

2. Connect the SAN hardware. See the RAID documents.

3. Is there a private network? This is a requirement.

4. 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? This is required to ensure data integrity for **all** nodes. See "Protect Data Integrity on All Nodes" on page 57.

5. 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 54 and "Use a Client-Only Tiebreaker" on page 55.

6. Read this book, the release notes, and the *CXFS 7 Client-Only Guide for SGI InfiniteStorage*.

7. Verify that the network is usable.

8. Install the CXFS software. See the *SGI InfiniteStorage Software Platform* release notes, Chapter 7, "Server-Capable Administration Node Installation" on page 131, and the *CXFS 7 Client-Only Guide for SGI InfiniteStorage*.

9. Completely configure and run a **small** cluster (such as 3 nodes). See Chapter 9, "Initial Setup of the Cluster" on page 149.

10. Look for errors in the daemon log files in the /var/log/cxfs directory.

11. If all is well, add the rest of the nodes. If there are problems, see Chapter 15, "Troubleshooting" on page 393.

12. Set up the filesystems. See Chapter 9, "Initial Setup of the Cluster" on page 149.

Summary of New Features from Previous Releases

This appendix contains a summary of the new features for the CXFS releases from previous versions of the CXFS administration guides (specific part numbers are noted).

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 Update

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(8):

- 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 5 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 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 5 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 007–4016–020 version contains the following:

- Support for client-only nodes running the Mac OS X operating system in the configuration tools.

- Information about console redirection.

- Information about the SGI TP9300, SGI TP9300S, and SGI TP9500S.

- Information about hardware changes and I/O fencing.

- Updated migration examples.

- Removal of the unimplemented GUI feature to define CXFS kernel messaging networks.

CXFS 3.2 Update

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 5 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 connected to the same public 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.

CXFS Version 5: Linux Servers

CXFS version 5 supports CXFS metadata servers running Linux.

CXFS 5.0

The 007–4016–027 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 5 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 the IRIX chapter in this book.

Note: CXFS does not support `SGIRDAC` mode.

- New initial installation and update procedures.

- 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`.

- The ability to generate streaming workload for SD/HD/2K/4K formats of video streams by using the `frametest`(1) command.

- Support for the `framesort` utility, which provides easy file-layout analysis and advanced file-sequence reorganization.

- The new site-changeable static system-tunable parameter `mtcp_hb_local_options`.

- Reorganization to improve clarity.

- Information about the `cmgr` command is located primarily in an appendix (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.

CXFS 5.2

The 007–4016–028 version contains the following:

- Support for *easy client configuration* using the cxfs_admin command and the new autoconf object, which was introduced in CXFS 5.1. This feature lets you specify new client-only nodes that are allowed to be automatically configured into the cluster database. (This action only applies to hosts that are not currently defined in the cluster database.)

- CXFS server-capable nodes must run SGI Foundation Software 1.

  SGI Foundation Software 1 is a new product from SGI consisting of technical support tools, utilities, and driver software that enable SGI's Linux systems to run reliably and consistently. SGI ProPack 6 is the next generation of SGI's suite of performance-optimization libraries and tools that accelerate applications on SGI's Linux systems. SGI ProPack 6 may be optionally installed on any CXFS node running SGI Foundation Software 1. For more information on the content of these products, upgrades, ordering, service contracts, and licensing, see Supportfolio.

- Information about limited support for case-insensitive CXFS filesystems, first introduced with CXFS 5.1.

- Support for *edge serving,* in which CXFS client nodes can act as servers for NFS, Samba, CIFS, or any third-party network filesystem exporting files from a CXFS filesystem. However, there are no performance guarantees when using edge serving; for best performance, SGI still recommends that you use the active metadata server. If you require a high-performance solution, contact SGI Professional Services.

CXFS 5.4

The 007–4016–029 version contains corrections the following new sections:

- "Unmount Filesystems Before Adding or Deleting Server-Capable Administration Nodes"

- "Change the Brocade Password when Prompted"

- "Use a Time Synchronization Application"

- "Turn Off Local XVM on Linux Nodes if Unused"

- "Configuring SuSEfirewall2"

- "Dynamic Parameters for Debugging Purposes Only"

CXFS 5.6

The 007–4016–030 version contains the following:

- Support for SLES 11

- Information about the `node is downrev` error

- Inclusion of the following debugging dynamic system tunable parameters (added in CXFS 5.5) that specify the maximum age of a granted range, measured in generations, before the client will voluntarily return it or the server will recall it:

  - `cxfs_client_range_age_max`

  - `cxfs_server_range_age_max`

- Guaranteed Rate I/O (GRIO) version 2 is disabled by default on server-capable administration nodes.

- RPM name change (implemented in ISSP 1.7): `cxfs-sysadm` changes to `sgi-sysadm`

- Some caveats and considerations that were formerly listed in the CXFS general release note have been incorporated into this guide.

CXFS Version 6: Linux SLES 11 Servers

CXFS version 6 supports CXFS metadata servers running SLES 11.

CXFS 6.0

The 007–5618–001 version contains the following:

Note: This new guide supersedes *CXFS 5 Administration Guide for SGI InfiniteStorage* (007–4016–030) for the CXFS 6 series. (The CXFS 5 guide remains available on the SGI Technical Publications Library.)

This new guide contains the following:

- Support for SLES 11 on server-capable administration nodes

- New client-only support:

  Mac OS X Snow Leopard 10.6
  RHEL 5.4
  SLES 10 SP3 (added in ISSP 1.9)

- Support for CXFS 6.0 and XVM 6.0 licenses. Upgrading customers must obtain new licenses; contact your SGI support person.

- Support for the following changes to cxfs_admin:

  – Support for entering read-only mode by default, requiring you to actively enter lock mode before making changes to the cluster database or performing administrative actions (assuming you are running cxfs_admin from a server-capable administration node or a node that has been given admin permission). You can override this default behavior by entering the -A (for *Admin write mode*) option on the command line, by changing options in the $HOME/.cxfs_admin file, or setting the CXFS_ADMIN_WRITE_MODE environment variable.

  – Support for the following new capabilities:

    - Control and contact a node with cxfs_admin

    - Query fence status

    - Raise a fence

    - Lower a fence

Note: The cmgr command is no longer supported. If you still require information about cmgr, see "Appendix F. Reference to cmgr Tasks" in *CXFS 5 Administration Guide for SGI InfiniteStorage* (007–4016–030) posted on the SGI Technical Publications Library:

http://techpubs.sgi.com/library/manuals/4000/007-4016-029/sgi_html/apf.html

- Support for external log filesystems. See "Supported XFS Features" on page 4.

- Removal of information about the AIX, IRIX, and Solaris client-only platforms and associated system controllers.

> **Note:** AIX, IRIX, and Solaris clients are not supported in ISSP 2.0 and 2.*X* releases going forward.
>
> The AIX, IRIX, and Solaris clients are still fully supported in the CXFS 5.*X* series in ISSP 1.*X*.

CXFS 6.2

The 007–5618–002 version contains the following:

- Support for SUSE Linux Enterprise Server 11 Service Pack 1 (SLES 11 SP1) on server-capable administration nodes

- Attributes in `cxfs_admin` that let you specify a list of values can now also take the + and − symbols to specify items to add or remove (first added in CXFS 6.1). The symbol must precede the list. For example, for the `nodes` attribute to the `modify` command:

  - `nodes=`*entirelist* to specify the entire list of nodes that can mount the filesystem, given as a comma-separated list; you must include the server-capable administration nodes in this list.

  - `nodes=+`*additionalnodes* to specify additional (+) nodes that can mount the filesystem, which will be added to the current list.

  - `nodes=−`*removednodes* to specify which nodes that were previously allowed to mount the filesystem but should now be prevented (−) from mounting the filesystem.

  > **Note:** The `+`/`−` symbols are not applicable in prompting mode.

- Clarification that if a path used for XVM failover V2 is not present, none of the attributes assigned to it in the `failover2.conf` file will take effect.

- Support for XVM block sizes of greater than 512 bytes for Linux nodes running the 2.6.32 or later kernel.

- `mkfs` options and CXFS appendix

- Changes to `clconf_info` status terminology:

Old Term	New Term
Inactive	Disabled
DOWN	Inactive

CXFS 6.4

The 007–5618–003 version contains the following:

- The ability to use `ssh`(1) to communicate with the Brocade switch for fencing, added in ISSP 2.3

- Clarifications about setting system-tunable kernel parameters

CXFS 6.6

The 007–5618–004 version includes the following changes:

- Support for SUSE Linux Enterprise Server (SLES) 11 SP2 server-capable administration nodes.

- Support for the `reset_user` attribute to the `create` and `modify` operations in `cxfs_admin` (added in ISSP 2.5). This lets you set the user name for the node's system controller port, which may not apply to all systems. If you wish to set or change the system controller user, consult the hardware manual for your node.

- The new section "Use Appropriate `affinity` Values in `/etc/failover2.conf` for XVM Devices".

- Clarifications to the following sections:

  - "Physical Storage Tools"

  - "Stability Issues on Clients Due to Token Optimizations"

- It is no longer necessary to create the files `/etc/apache2/conf.d/cxfs.conf` and `/etc/apache2/conf.d/xvm.conf` when running the web-based version of the CXFS GUI and XVM GUI.

CXFS Version 7: Linux SLES 11 and RHEL Servers

CXFS version 7 supports CXFS metadata servers running SLES 11 or RHEL, as documented in the release notes.

CXFS 7.0

The 007–5618–005 version contains the following:

- New CXFS 7.0 licenses are required to run CXFS 7.0. The new license structure uses a simplified node-count scheme; there is no restriction on the CPU count for server-capable administration nodes or client-only nodes.

  If you use an SGI UV 100, UV 1000, UV 2000, or Altix ia64 system as a CXFS client-only node, you must also purchase the CXFS 7.x Feature Enabler for SGI NUMAlink systems. (These systems are not supported as CXFS server-capable administration nodes.) A single Feature Enabler is required per system and it does not generate an LK key. For a partitioned NUMAlink system, each partition requires a client license but only one Feature Enabler is required for the entire system.

  Note: If you are upgrading, you must obtain new CXFS license keys from SGI Support prior to installing the ISSP 3.0 software.

  Prior versions of CXFS will not function with the CXFS new license keys. If you are upgrading, you should keep your prior licenses available if you must downgrade. If the `/etc/lk/keys/dat` file contains licenses from a prior release, they will be ignored after you upgrade to CXFS 7.0.

  GRIO V2 keys are unchanged; customers can use their existing `GRIO_CLUSTER` key with CXFS 7.0.

- Support for x86_64 server-capable administration nodes all running one of the following operating systems:

  - SUSE® Linux® Enterprise Server 11 Service Pack 2 (SLES 11 SP2)

  - Red Hat® Enterprise Linux 6 (RHEL 6)

  For more details, see the *General Release Note for CXFS.*

All of the server-capable administration nodes and optional edge-serving nodes must run the same operating system.

Note: As of ISSP 3.0, CXFS does not support ia64 server-capable administration nodes. However, CXFS does still support ia64 SLES client-only nodes as listed in the CXFS general release note. For information about L2 reset on client-only nodes, see the *CXFS 7 Client-Only Guide for SGI InfiniteStorage*

ISSP 3.0 provides CXFS 7.0, which is a major release. To upgrade to CXFS 7.0, you must first disable the cluster and then upgrade all of the nodes to ISSP 3.0. You cannot use the rolling upgrade procedure to upgrade from a release prior to ISSP 3.0. If you attempt to run a cluster with an incorrect mix of releases, you will get `aremove version is too old` error.

- Internet Protocol version 6 (IPv6) use is documented but is provided only as a **technology preview**.

⚠ **Caution:** Technology previews are not supported for production use. For more information, see the ISSP release note.

See:

- "Network and Connectivity" on page 36

- "IPv6: Use Any Legitimate Address Representation" on page 75

- CXFS now supports relocation/recovery of a CXFS filesystem's metadata server to a server-capable administration node that is already using the filesystem, such as for NFS or Samba. (CXFS no longer requires a *standby node*, which is a potential metadata server for a given filesystem that does not currently run any applications using the filesystem.)

Note: Relocation is disabled by default. You should enable relocation only during the time when you intend to perform relocations; otherwise, leave relocation disabled.

- New topics in the best-practices chapter:

- Using mount options appropriately

– Modifying `updatedb` to avoid unnecessary load

– Preventing stalled-recovery timeout in a non-HA DMF environment

– Preallocating space for directories when appropriate

– Avoiding the dropping of out-of-order frames

– Suppressing change notification for switch ports connected to nodes

• The format for the cell set shown in various messages has changed in ISSP 3.0 to the following format, where *bitmask* is a 64–bit hexadecimal bitmask that represents the cell IDs:

`<0:0:0:0:0:0:0:0:`*bitmask*`>`

CXFS 7.1

The 007–5618–006 version contains the following:

• Support for a CXFS cluster with up to 96 nodes (supported as of ISSP 3.1)

• Support for users with greater than 32 groups (supported as of ISSP 3.0)

• Clarifications about the following:

– Filesystem status for a node displayed by the `cxfs_admin` command

– Rolling upgrades, including a new section about the importance of upgrading all servers first

CXFS 7.2

The 007–5618–007 version contains the following:

• You can now list multiple network addresses for the `private_net` attribute when you define a node, and those addresses will be used in order by default so long as the same network subnet is defined in order for each node in the cluster. If you want to limit the networks chosen or reprioritize the list, you can use the `failover_net` attribute on a cluster-wide basis.

• Corrections and clarifications.

CXFS 7.3

The 007–5619–008 version includes corrections and clarifications, including a section about defining `failover_net` to order IPv6 addresses for automatic CXFS configuration.

Glossary

ACL

Access control list.

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*.

administration node

See *server-capable administration node*.

administrative stop

See *forced CXFS shutdown*.

advanced mode

The cxfs_admin complexity mode that provides a list of possible choices when using the <TAB> key, prompts for all possible fields, displays all attributes, and includes debugging information in output.

age

The number of 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).

ALUA

Asymmetric logical unit access, a feature of some RAID devices that permits automatic path failover

ARP

Address resolution protocol.

basic mode

The `cxfs_admin` complexity mode that only shows the common options and attributes in `show` output, provides a list of possible choices when using the `<TAB>` key, and uses prompting.

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 when it is defined with `cxfs_admin`. The first node in the cluster has cell ID 0, and each subsequent node added gets the next available incremental cell ID. If a node is removed from the cluster, its cell ID becomes available. It differs from *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*.

client recovery

The process by which the active metadata server resolves the state of client nodes after a client is removed from the CXFS kernel membership due to an interruption in CXFS services. See also *client recovery* and *metadata-server recovery*.

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 SAN 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 44.

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 39.

cluster administrator

The person responsible for managing and maintaining a cluster.

cluster database

The database that contains configuration information about all nodes and the cluster. The database is managed by the cluster administration daemons.

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 domain

The XVM concept in which a filesystem applies to the entire cluster, not just to the local node. See also *local domain*.

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.

cluster node

A node that is defined as part of the cluster. See also *node*.

cluster services

See *CXFS cluster services* .

complexity mode

The manner in which `cxfs_admin` operates. See *basic mode* and *advanced mode.*

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.

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 cluster services on a client-only node.

CXFS control daemon

The daemon (`clconfd`) that controls CXFS cluster services on a server-capable administration node. See "Cluster Administration Daemons" on page 43.

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 cluster 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 cluster services are `clconfd` on a server-capable administration node and `cxfs_client` on a client-only node. See "CXFS Services" on page 29.

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.*

easy client configuration

Using the `cxfs_admin` command and the `autoconf` object to specify new client-only nodes that are allowed to be automatically configured into the cluster database.

edge-serving

See *NFS edge-serving.*

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.

GRIO

Guaranteed rate I/O.

GRIOv2 daemon

The `ggd2` daemon.

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, "CXFS GUI" on page 167.

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 InfiniteStorage 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-nodeA log only on the node nodeA 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 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 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 the 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 active metadata server moves from one node to another due to an interruption in CXFS services on the first node. See also *client recovery* and *recovery*.

multiOS cluster

A cluster that is running multiple operating systems, such Linux and Windows.

multiport serial adapter cable

A device that provides four DB9 serial ports from a 36-pin connector.

NFS edge-serving

A configuration in which CXFS client nodes can export data with NFS.

node

A *node* is an operating system (OS) image, usually an individual computer. (This 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 only one cluster. See also *client-only node* and *server-capable administration 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 differs from *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 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.)

When you define a node using `cxfs_admin`, it is automatically added to both the pool and the specific cluster definition. If you define a node with the CXFS GUI, it is merely added to the pool and you must explicitly add it to the cluster.

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.)

port type

The type of system controller port used for node reset.

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 **defined in the cluster** plus the tiebreaker node (usually a client-only node) are required to **form** an initial membership

  - Half (**50%**) of the server-capable administration nodes **defined in the cluster** are required to **maintain** an existing membership

- For **cluster database** membership, **50%** of the server-capable administration nodes **in the pool** are required to **form and maintain** a cluster.

Note: When using the CXFS GUI, a newly defined node is added to the pool and must be explicitly added to the cluster definition; when using the `cxfs_admin` tool, a newly defined node is added automatically to both the pool and the cluster definition.

---

quorum master

The node that is chosen to propagate 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 *client recovery* and *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 the system controller.

reset communication

The communication interface used to send a system reset to a controller port on a remote node.

reset method

The action taken upon node failure.

RSCN

Registered state change notification.

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 of client-only nodes that may join the cluster membership.

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 an unresponsive server-capable administration node) and the lack of reset 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.

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

XSDM, 30
XVM, 34
  logical volume creation, 160
  requirement, 37
xvm, 394, 395, 399, 400, 403, 404
XVM cluster mirror status error, 449
xvm dump, 49
XVM failover, 74
  commands, 146
XVM failover V2, 145
XVM failover version 1, 9
XVM in local mode, 88
XVM label corruption, 49
XVM mirroring, 75
XVM path, 89
XVM repair, 50

XVM shortcuts, 175
XVM statistics, 391
XVM volume mapping to storage targets, 354
XVM volumes, 400
xvmgr, 40

Y

YAST configuration, 51

Z

zoning, 25