sgi®

High Availability Guide for SGI® InfiniteStorage

007–5617–007

New Features in this Guide

This revision contains the following:

- DMF servers, DMF parallel data mover nodes, CXFS servers, and CXFS edge-serving nodes must be x86_64 systems. Therefore, support for the `l2network` resource agent has been removed.
- An updated procedure in "Configuring Samba for HA" on page 135.

Record of Revision

Version	Description
001	July 2010 Original publication, supporting the SGI® InfiniteStorage Software Platform (ISSP) 2.1 release
002	September 2010 Revision to support ISSP 2.2
003	January 2011 Revision to support ISSP 2.3
004	April 2011 Revision to support ISSP 2.4
005	October 2011 Revision to support ISSP 2.5
006	April 2012 Revision to support ISSP 2.6
007	April 2013 Revision to support ISSP 3.0

Contents

Figures

Tables

About This Guide

This publication provides information about creating resources for the high-availability (HA) SGI® resource agents that SGI provides for use with the Corosync, OpenAIS, and Pacemaker products. These products are provided with the SUSE® Linux® Enterprise High Availability Extension (HAE).

Prerequisites

To use this guide, you must have access to the SUSE HAE *High Availability Guide* provided by the following website:

http://www.suse.com/documentation/sle_ha/

Related SGI Publications

The following SGI publications contain additional information:

- *CXFS 7 Administrator Guide for SGI InfiniteStorage*
- *CXFS 7 Client-Only Guide for SGI InfiniteStorage*
- *DMF 6 Administrator Guide for SGI InfiniteStorage*
- *DMF 5 Filesystem Audit Guide for SGI InfiniteStorage*
- *DMF 5 Filesystem Audit Guide for SGI InfiniteStorage*
- *OpenVault Administrator Guide for SGI InfiniteStorage*
- *SGI InfiniteStorage Software Platform* (ISSP) release note (`README.txt`)
- *TMF 5 Administrator's Guide for SGI InfiniteStorage*
- *XVM Volume Manager Administrator Guide*
- The hardware guide for your SGI server

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

Conventions

In this guide, *High Availability Extension* and *HAE* refer to the SUSE Linux Enterprise High Availability Extension product.

Note: This guide shows administrative commands that act on a group by using the variable *resourceGROUP* or the example group name, such as `dmfGroup`. Other commands that act on a resource primitive use the variable *resourcePRIMITIVE* or an example primitive name, such as `dmf`.

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.
...	Ellipses indicate that a preceding element can be repeated.
`manpage`(*x*)	Man page section identifiers appear in parentheses after man page names.
GUI	This font denotes the names of graphical user interface (GUI) elements such as windows, screens, dialog boxes, menus, toolbars, icons, buttons, boxes, fields, and lists.
`cxfsclient#`	This prompt indicates that the example command is executed on a CXFS client-only node
`cxfsserver#`	This prompt indicates that the example command is executed on a CXFS server-capable administration node
`dmfserver#`	This prompt indicates that the example command is executed on a DMF server
`downnode#`	This prompt indicates that the example command is executed on the HA node that requires maintenance
`ha#`	This prompt indicates that the example command is executed on any node that is or will be in the HA cluster
`nfsclient#`	This prompt indicates that the example command is executed on an NFS client outside of the HA cluster
`node1#`	This prompt indicates that the example command is executed on `node1`, a node that is or will be in the HA cluster
`node2#`	This prompt indicates that the example command is executed on `node2`, a node that is or will be in the HA cluster
`otherhost#`	This prompt indicates that the example command is executed on machine outside of the HA cluster

`pdmn#`	This prompt indicates that the example command is executed on a DMF parallel data mover node
`pdmn1#`	This prompt indicates that the example command is executed on `pdmn1` (analogous to `node1`), a node that is or will be in the HA cluster
`pdmn2#`	This prompt indicates that the example command is executed on `pdmn2` (analogous to `node2`), a node that is or will be in the HA cluster
`upnode#`	This prompt indicates that the example command is executed on the HA node that does not require maintenance

Reader Comments

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

You can contact SGI in any of the following ways:

- Send e-mail to the following address:

  techpubs@sgi.com

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

- Send mail to the following address:

  SGI
  Technical Publications
  46600 Landing Parkway
  Fremont, CA 94538

SGI values your comments and will respond to them promptly.

Introduction

This chapter discusses the following:

- "High Availability Overview" on page 1
- "SGI Resource Agents and RPMs" on page 2
- "Failover Example Scenarios" on page 4
- "Configuration Tools" on page 10

High Availability Overview

The Corosync, OpenAIS, and Pacemaker products provide the infrastructure to fail over individual *highly available (HA) resources* and entire *HA services* that survive a single point of failure.

A *resource* is managed by HA. A *resource group* is a set of resources that must be managed and failed over from one node to another as a set. An *entire HA service* can include resource groups and individual resources and is usually associated with an IP address. HA starts, monitors, and stops resources and entire HA services. A *resource agent* is the set of software that allows an application to be highly available without modifying the application itself.

HA uses the IP address of a resource to direct clients to the node currently running the resource. Each resource is actively owned by one node. If that node fails, an alternate node restarts the HA services of the failed node. To application clients, the services on the alternate node are indistinguishable from those on the original node.

This guide does not discuss HA in general, nor does it provide details about configuring an HA cluster; for those details, see the SUSE *High Availability Guide* provided by the following website:

http://www.suse.com/documentation/sle_ha/

SGI Resource Agents and RPMs

Table 1-1 and Table 1-2 list the Open Cluster Framework (OCF) resource agents and the STONITH (*shoot the other node in the head*) resource agents that SGI provides in the **SGI ISSP High Availability** YaST pattern.

For information about software installation, see the *SGI InfiniteStorage™ Software Platform* (ISSP) release note and Chapter 4, "Outline of the Configuration Procedure" on page 39.

Table 1-1 Resource Agents in the `sgi-ha-ocf-plugins` RPM

Resource Agent	Description
`copan_ov_client`	COPAN MAID OpenVault client for DMF parallel data mover nodes and DMF servers. See: • "Creating the COPAN MAID OpenVault Client Primitive" on page 121 • "Creating the COPAN OpenVault Client Primitive" on page 162
`cxfs`	CXFS™ clustered filesystems whose metadata server location must follow the location of another resource, such as DMF or NFS. See "Creating the CXFS Primitive" on page 79.
`cxfs-client`	CXFS clustered filesystems (mounted on a CXFS client-only node) that are required to support another resource, such as those to be NFS-served from a CXFS client-only node or those used for a DMF parallel data mover node. See: • "Creating the CXFS Client Primitive" on page 63 • "Creating the CXFS Client Primitive" on page 159
`cxfs-client-nfsserver`	NFS server on a CXFS client-only node. See "Creating the CXFS Client NFS Server Primitive" on page 66.
`cxfs-client-smnotify`	Network Status Monitor (NSM) lock reclaim notification on a CXFS client-only node. See "Creating the CXFS Client NSM Notification Primitive" on page 71.
`dmf`	DMF server. See "Creating the DMF Primitive" on page 127.
`dmfman`	DMF Manager tool. See "Creating the DMF Manager Primitive" on page 142.

Resource Agent	Description
`dmfsoap`	DMF client Simple Object Access Protocol (SOAP) service. See "Creating the COPAN MAID OpenVault Client Primitive" on page 121.
`lxvm`	Local XVM volume manager. See "Creating the Local XVM Primitive" on page 83.
`openvault`	OpenVault mounting service for DMF. See "Creating the OpenVault Primitive" on page 114.
`tmf`	Tape Management Facility (TMF) mounting service for DMF. See "Creating the TMF Primitive" on page 101.

Table 1-2 Resource Agents in the `sgi-ha-stonith-plugins` RPM

Resource Agent	Description
`sgi-ipmi`	STONITH node-level fencing for systems with a baseboard management controller (BMC) using intelligent platform management interface (IPMI), such as SGI x86_64. See "Creating the SGI IPMI STONITH Primitive " on page 166.

Although the SGI resource agents can be used independently, this guide provides example procedures to configure the set of resources required to provide highly available versions of the following:

- CXFS NFS edge-serving from CXFS client-only nodes in a two-node active/active HA cluster. See "CXFS™ NFS Edge-Serving Failover" on page 4.
- DMF in a two–node active/passive HA cluster (which can optionally include COPAN MAID shelves). See "DMF Failover" on page 7.
- COPAN MAID shelves in an active/active HA cluster that consists of two parallel data mover nodes. See "COPAN MAID OpenVault Client HA Service for Mover Nodes" on page 8.

Although other configurations may be possible, SGI has tested and recommends the above HA environments.

Note: The attributes and the various value recommendations listed in this guide are in support of the examples used in this guide. If you are using the resources in a different manner, you must evaluate whether these recommendations and use of meta attributes apply to your intended site-specific purpose.

Failover Example Scenarios

This section discusses the following examples:

- "CXFS™ NFS Edge-Serving Failover" on page 4
- "DMF Failover" on page 7
- "COPAN MAID OpenVault Client HA Service for Mover Nodes" on page 8

CXFS™ NFS Edge-Serving Failover

Figure 1-1 and Figure 1-2 describe an example process of failing over an CXFS NFS edge-serving HA service in a two-node HA cluster using active/active mode.

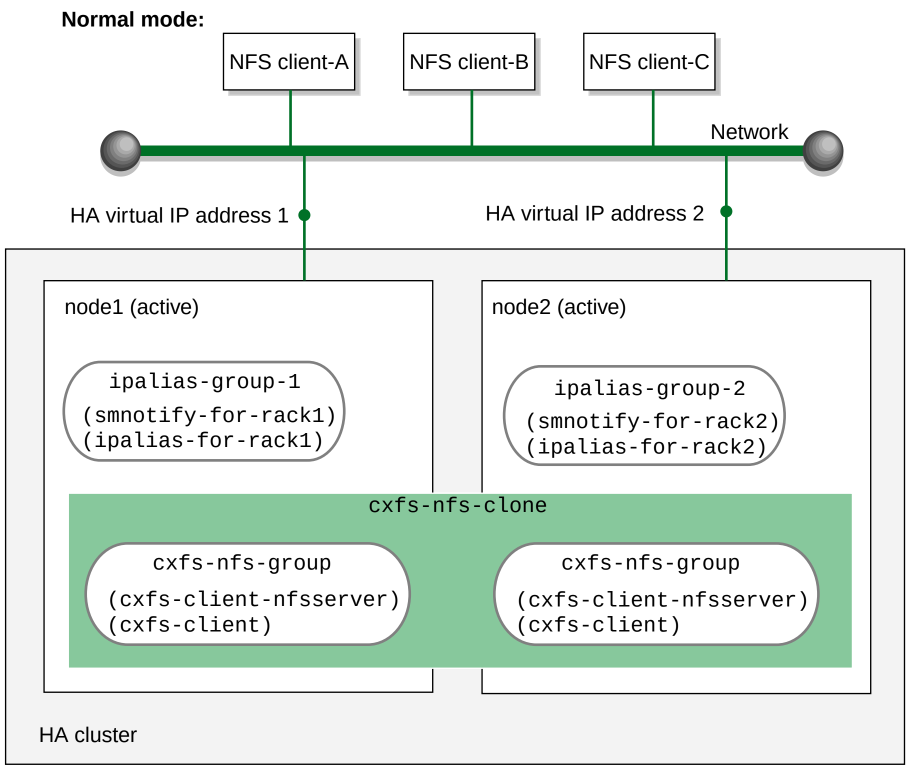

Figure 1-1 CXFS NFS Edge-Serving HA Service — Normal State

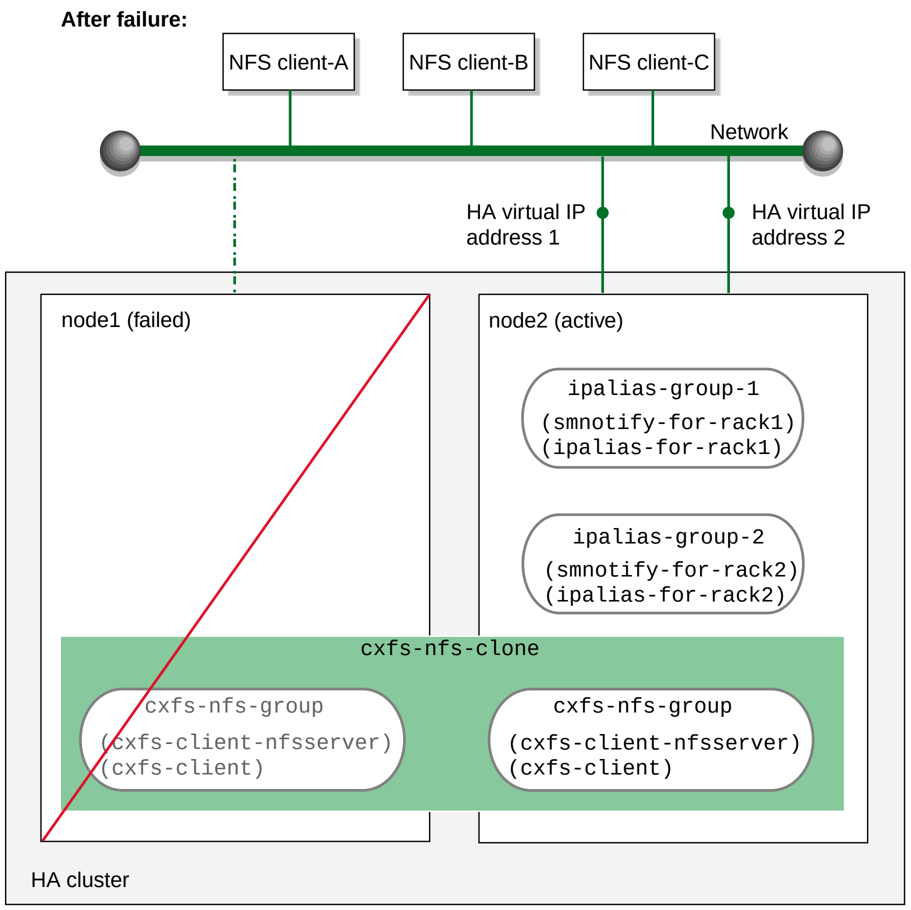

Figure 1-2 CXFS NFS Edge-Serving HA Service — After Failover

DMF Failover

Figure 1-3 and Figure 1-4 describe an example process of failing over a DMF HA service in a two-node HA cluster using active/passive mode.

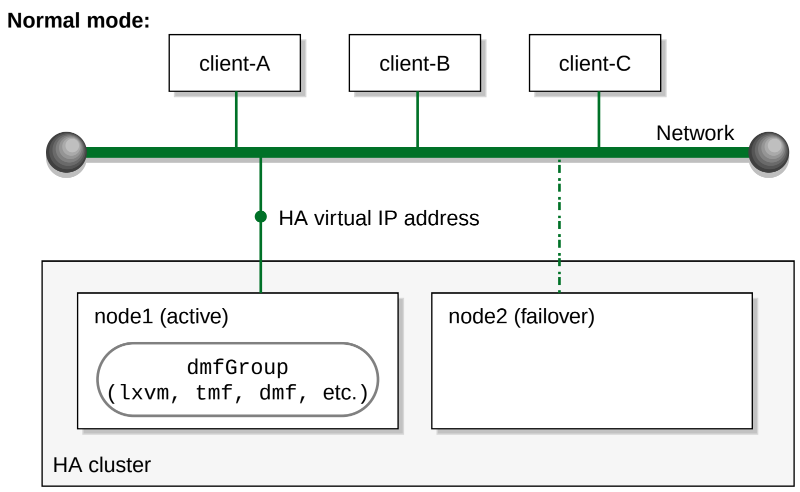

Figure 1-3 DMF HA Service — Normal State

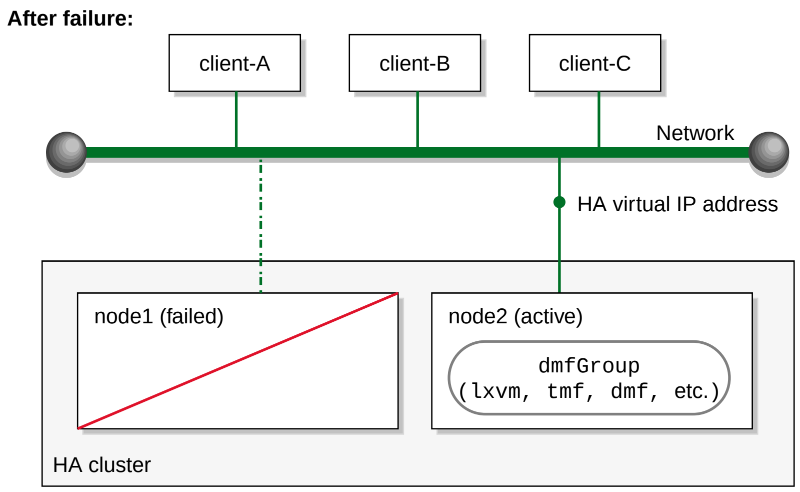

Figure 1-4 DMF HA Service — After Failover

COPAN MAID OpenVault Client HA Service for Mover Nodes

Figure 1-5 and Figure 1-6 describe an example process of failing over the OpenVault client service for a COPAN MAID shelf in a two-node HA cluster (consisting of two parallel data mover nodes) using active/active mode.

At initialization, each parallel data mover node is the default owner node of two COPAN OpenVault client resources, as represented in the dark lines in Figure 1-5 (where `pdmn1` is the owner node of shelves 0 and 1, and `pdmn2` is the owner node of shelves 3 and 4).

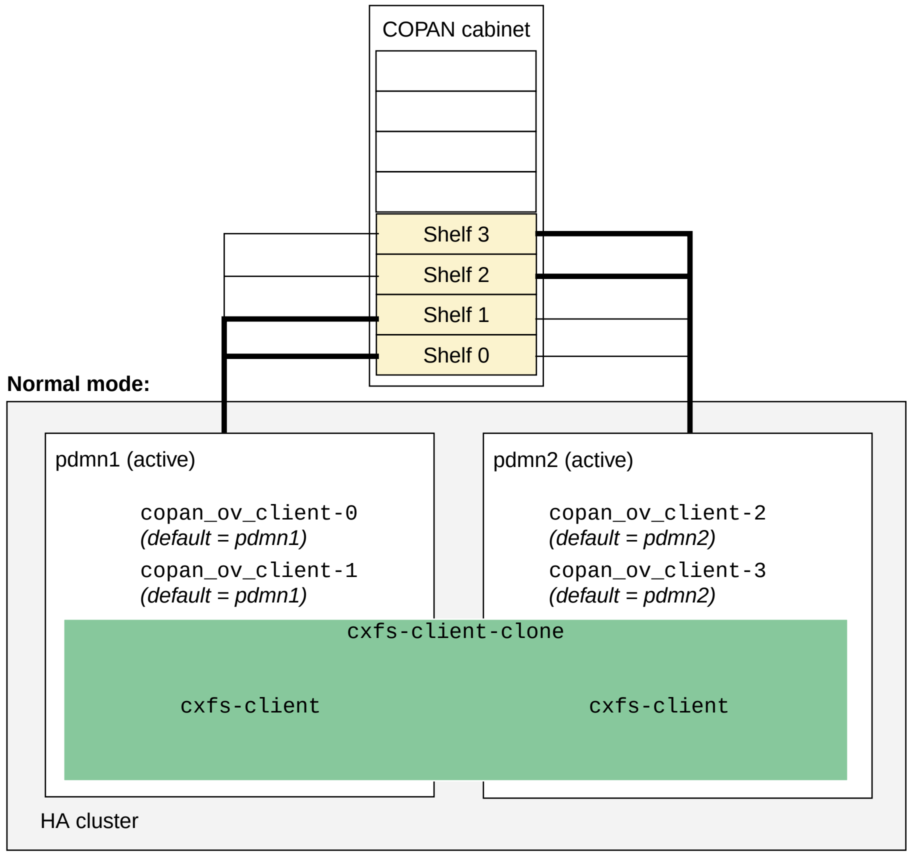

Figure 1-5 COPAN OpenVault Client HA Service for Mover Nodes — Normal State

When pdmn1 fails, its COPAN OpenVault client resources move to pdmn2 and pdmn2 becomes the current owner node of all of the shelves, as shown in Figure 1-6.

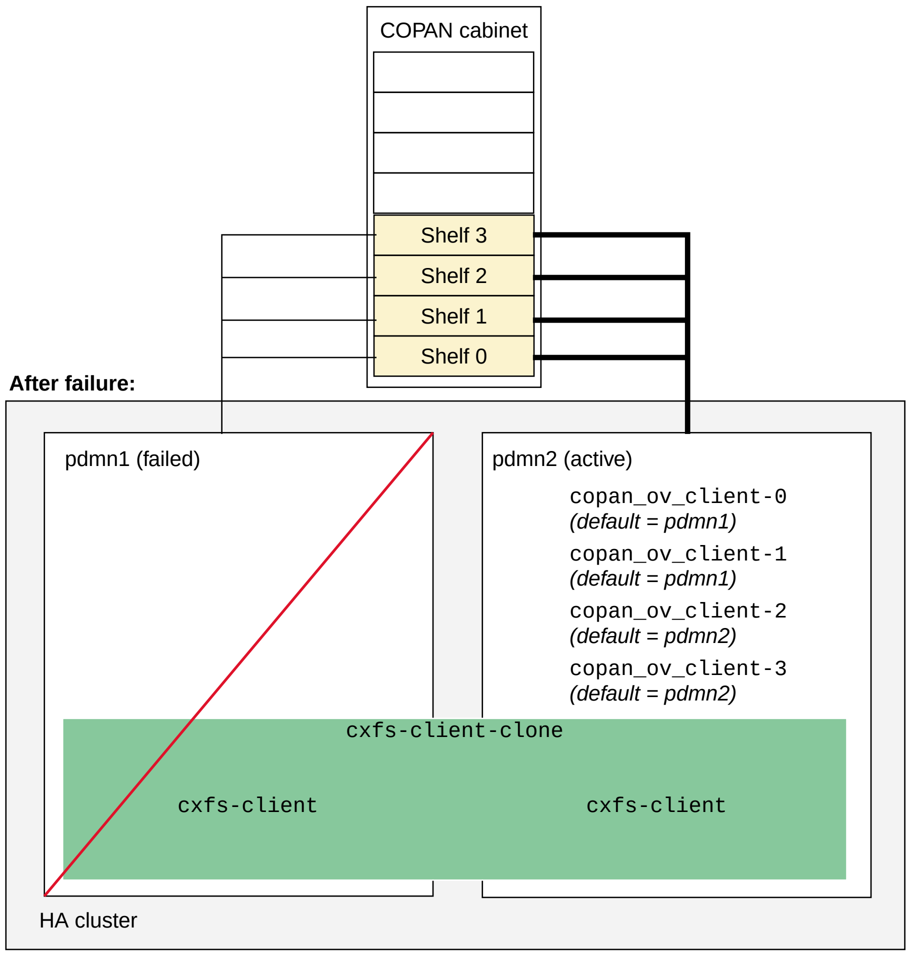

Figure 1-6 COPAN OpenVault Client HA Service for Mover Nodes — After Failover

After pdmn1 recovers and rejoins the HA cluster, you can choose a convenient time to manually move copan_ov_client_0 and copan_ov_client_1 back to pdmn1 to balance the load and return the HA cluster to its normal state (see "Manually Moving a copan_ov_client Resource" on page 178). You should perform this procedure during a time of low shelf activity, because moving a `copan_ov_client` resource involves disabling the active node via the `dmnode_admin` command (which results in stopping all activity to all shelves owned by the node).

Configuration Tools

The procedures in this guide use the following tools as documented in the SUSE *High Availability Guide* and the `crm`(8) online help:

- YaST installation and configuration tool
- Pacemaker graphical user interface (GUI) for HA resource management, accessed with the `crm_gui` command
- Linux HA Management Client command-line administration tools such as `crm`(8), `cibadmin`(8), and `crm_verify`(8)

Chapter 2

Best Practices

The following are best practices when using SGI resource agents:

- "Preliminary Best Practices Before Introducing HA" on page 11
- "HA Configuration and Testing Best Practices" on page 13
- "Administrative Best Practices" on page 18
- "Maintenance Best Practices" on page 22

Preliminary Best Practices Before Introducing HA

The following are best practices for your environment before introducing high availability:

- "Ensure the System is Ready for HA" on page 11
- "Prepare to Use the GUI" on page 12
- "Use a Separate Filesystem for CXFS NFS State Information" on page 12
- "Use Consistent Virtual Hostnames" on page 12
- "Ensure that the Debug RPM Matches the Kernel" on page 12

Ensure the System is Ready for HA

Do the following:

- Fix networking issues first.
- Make your overall system configuration as simple as possible — complexity makes high availability harder to achieve.
- Use redundancy in your system components to avoid single points of failure.
- Perform regular and frequent system backups.
- Configure and test the standard services (like DMF) in normal mode before making them highly available — doing so will make problems easier to diagnose.

Configure and test the base HA cluster before adding the SGI resource group and resource primitives. To do these things, review Chapter 4, "Outline of the Configuration Procedure", and then follow the detailed example steps in the remainder of the guide, as appropriate for your site.

Prepare to Use the GUI

Before using the GUI, you should do one of the following:

- Add the `root` user to the `haclient` group
- Set the password for the `hacluster` user before you start `crm_gui`

You should set an appropriate password for the HA GUI (`crm_gui`). You can log in to the GUI with any user ID that has access to the `haclient` group, but you must know the password for the user ID. By default, the GUI uses the `hacluster` user ID.

Use a Separate Filesystem for CXFS NFS State Information

For CXFS NFS edge-serving, use a separate shared CXFS filesystem on which to store NFS state information. The state is kept on disk and must be available while any edge servers are running. A simple setup where only one filesystem is being served via NFS can keep the state directories on the same filesystem that is being served.

Use Consistent Virtual Hostnames

When configuring OpenVault and DMF, be consistent when specifying virtual hostnames, always using either the short hostname (like `myhost`) everywhere or the fully qualified domain name (like `myhost.mycompany.com`) everywhere.

Ensure that the Debug RPM Matches the Kernel

Ensure that the `kernel-default-debuginfo` RPM that matches the kernel is installed on the system. This will let you make the best use of the recommended SGI Support tools in case you must send a kernel crash dump to SGI for troubleshooting purposes.

Caution: If your HA cluster includes CXFS nodes, you should use a CXFS fail policy that does not include `reset`. If the fail policy does not include `reset`, you must perform a manual reset of a failed node if the node does not reboot automatically.

HA Configuration and Testing Best Practices

The following are best practices for configuring and testing the HA system:

- "Use the Appropriate HA Tools" on page 13
- "Use the Information in this Guide Correctly" on page 14
- "Avoid Unnecessary Failovers" on page 16
- "Use IDs Appropriately" on page 17
- "Always use STONITH" on page 17
- "Use One Group for DMF HA" on page 17
- "Examine the Use of Stickiness and Thresholds" on page 17
- "Consider a `pingd` Resource" on page 17

Use the Appropriate HA Tools

Use the appropriate HA tools:

- Use the HA GUI (`crm_gui`) to initially configure the HA cluster and to make configuration changes, particularly changes to timeout values:
  - Click **Apply** only after you have entered all of the required operations.
  - Resize the GUI as needed to view some fields and tabs. In some cases, the cursor must be positioned on the left-edge of a tab in order to select it.
  - When you enter a time value that you want to be in seconds, you must often include the character “`s`” (as in `30s`) because the default is usually in milliseconds.

Note: If you make configuration changes with the `crm` command, use a shadow environment (`crm cib new`), so that you can verify those changes before applying them to the running cluster information base (CIB). See the `crm`(8) online help for more information.

- Use the `crm`(8) command or the HA GUI to test resource primitives. This guide typically provides the `crm` command-line method.
- Use the following command to verify changes you make to the CIB, with each resource primitive that you define:

```
ha# crm_verify -LV
```

Use the Information in this Guide Correctly

Note the following about using the configuration information in this guide:

- Use values appropriate for your site (most instance attributes and timeouts are site-specific). Values shown in *italic font* in this guide are site-specific and are therefore changeable; suggested starting values that you must enter are shown in `literalfont`. When possible, some guidance is given for determining appropriate values.

  Note: Fields that are unnecessary or for which the GUI provides appropriate defaults are not addressed in this guide.

- **monitor** is the **name** value of the operation that determines if the resource is operating correctly. There are two types of monitor operations:
  - *Standard monitor operations* monitor the operation of the resources at an interval of the specified time (the interval begins at the end of the last monitor completion). Each monitor operation will time-out after the specified number of seconds. If the monitor operation fails, it will attempt to restart the resource.
  - *Probe monitor operations* check to see if the resources are already running.

    Note: Always use a probe operation, even if you do not use a standard monitor operation.

- **start** is the **name** value of the operation that initiates the resource. It will time-out after a specified time. It requires that **fencing** is configured and active in order to start the resource. Using system reset as a fencing method is required in order to preserve data integrity. If the start operation fails, it will attempt to restart the resource.

  Note: *Fencing* in HA terminology (node-level fencing) is not the same as *fencing* in CXFS terminology (I/O-level fencing).

- **stop** is the **name** value of the operation that terminates or gives up control of the resource. It will time-out after the specified time. If the stop operation fails, it will attempt to fence the node on which the failure occurred. The stop fail policy must be set to **fence** and a STONITH facility must be configured according to the requirements for your site (see Chapter 9, "STONITH Examples" on page 165.)

  Note: Longer stop operation timeouts may result in longer failover times, and shorter stop operation timeouts may result in more frequent system reset events.

- **migration-threshold** specifies a count of failures at which the current node will receive a score of `-INFINITY` so that the resource must fail over to another node and is not eligible for restart on the local node, based on the number of start, monitor, or stop failures that this resource has experienced.
- **resource-stickiness** specifies a score for the preference to keep this resource on the node on which it is currently running. A positive value specifies a preference for the resource to remain on the node on which it is currently running. This preference may only be overridden if the node becomes ineligible to run the resource (if the node fails over) or if there is a start, monitor, or stop failure for this resource or another resource in the same resource group.
- Some **Operations** fields are accessed under the **Optional** tab. Those that are required for the SGI implementation of HA are listed in this guide using the format **Optional** > ***Field Name***.

  Note: Although the GUI organizes these items under the **Optional** heading, they are not optional for the SGI implementation of HA; you must provide them in your configuration. Similarly, the **Required** subsections in this chapter refer to those items located under that heading in the GUI; the label does not imply that only those values are required.

- In many cases, there are pull-down lists that contain possible values (shown in **boldface** in this guide). In other cases, you must enter in text (shown in `literalfont`).
- In general, you must use the values shown in this guide for meta attributes and for those values available from a pull-down list.
- **ID** is the unique identification of the clone, resource group, or resource primitive, such as `cxfs`. This can be any name you like as long as it does not include spaces. For the monitor, start, and stop operations, a unique ID will be generated for you based on the primitive name and interval, such as `cxfs_op_monitor_30s`.
- **Class**, **Provider**, and **Type** must use the exact values shown in this guide.

Avoid Unnecessary Failovers

This guide provides some starting points for monitoring values, but you should test these values under typical load for your site to determine if they are appropriate for your use. In some cases, you may wish to avoid monitoring other than probe monitoring (probe monitoring is always appropriate).

To prevent a given resource from being monitored and possibly triggering failovers, do not define a nonprobe monitor operation. This may be particularly useful for those resources that are not critical (such as DMF Manager) but should still move with the rest of the resource group or for those resources that have an alternative method for monitoring. You can also make timeout values larger to decrease the likelyhood of an unnecessary failover.

Do not create explicit location, colocation, or order constraints on individual resource primitives except as directed in this guide.

Caution: Defining constraints on the resource primitives can lead to a deadlock situation in which the group has conflicting constraints that prevent it from starting anywhere.

Use IDs Appropriately

Use unique IDs for all resource clones, groups, and primitives.

Do not use spaces in resource IDs because this may cause HA or other supporting software to behave in a confusing manner.

Always use STONITH

Always use STONITH node-level fencing to protect data integrity in case of failure.

Use One Group for DMF HA

For a DMF HA cluster, place all resource primitives within one resource group. The resource group mechanism incorporates implied colocation constraints as well as resource order constraints. The resource group concept lets you control the resources as a single entity, which greatly simplifies administration.

Examine the Use of Stickiness and Thresholds

You may want to examine the use of the `resource_stickiness` and `migration_threshold` attributes of resources to control how often and to what node failover will occur. The examples in this book result in the resource or resource group failing over to the alternate node on the first failure of any of the resource primitives. In this default setting, there is no automatic failback to the original node. For more information about score calculation, see the SUSE documentation and the following website:

http://www.clusterlabs.org/doc/en-US/Pacemaker/1.0/html/Pacemaker_Explained/index.html

Consider a `pingd` Resource

The `IPaddr2` virtual IP address resource agent monitors the existence of the IP alias address on the interface, but it does not monitor network interface controller (NIC) interface availability. You may want to consider defining a `pingd` resource. For more information, see the information about moving resources due to connectivity changes at the following website:

http://www.clusterlabs.org/doc/en-US/Pacemaker/1.0/html/Pacemaker_Explained/index.html

Administrative Best Practices

The following are best practices for administering the HA cluster:

- "Use the Appropriate Tools" on page 18
- "Make a Backup Copy of the CIB" on page 18
- "Monitor Status Information for Problems" on page 19
- "Get More Information by Increasing Verbosity" on page 19
- "Clear Failcount Values After Resolving the Problem" on page 19
- "Remove Implicit Constraints after Explicitly Moving a Resource" on page 20
- "Set the Core Membership Timeout Value Appropriately" on page 20
- "Use the Default Consensus Value" on page 21
- "Upgrade Appropriately" on page 21
- "Do Not Use an HA CXFS NFS Edge Server as an NFS Client" on page 21
- "Include PID in Core Files" on page 21

Use the Appropriate Tools

Use the `crm`(8) command or the HA GUI to obtain status and perform administrative actions. Use the online help available with the command.

For verification, SGI recommends that you use `crm_verify`(8).

Note: The `crm configure verify` command is not equivalent to the `crm_verify` command.

Make a Backup Copy of the CIB

After you have successfully completed the initial configuration, make a backup copy of the working CIB so that you can return to it if necessary after future changes. See:

- "Backing Up the CIB" on page 174
- "Recovering from a CIB Corruption" on page 206

Before making changes to an existing HA configuration, ensure that you have a good backup copy of the current CIB so that you can return to it if necessary. (If you encounter a corrupted CIB, you must erase it by force and then restore the information about resources, constraints, and configuration from a backup copy of a good CIB.)

After you establish that your changed configuration is good, make a new backup of the CIB.

For more information, see the SUSE *High Availability Guide* and the `crm`(8) man page.

Monitor Status Information for Problems

Periodically watch the output of the following commands for problems:

```
ha# crm status
ha# crm_verify -LV
ha# crm status failcounts
```

Refer to the `/var/log/messages` system log periodically (to ensure that you are aware of operations automatically initiated by HA) and if you notice errors. See "Reviewing the Log File" on page 175.

Get More Information by Increasing Verbosity

To get more information, you may add multiple `-V` options to many of the HA commands in order to increase verbosity. For example:

```
ha# crm_verify -LVVV
```

Clear Failcount Values After Resolving the Problem

After a failure, clear the resource primitive failcount values for a node immediately after resolving the cause of the failure (or reboot the system). See "Clearing the Resource Primitive Failcount" on page 175.

Remove Implicit Constraints after Explicitly Moving a Resource

If you want to move or start a resource or resource group on a specific node, enter the following:

```
ha# crm resource move resource_or_resourceGROUP node
```

The result of this command is to create a location constraint with a score of `INFINITY` for the specified resource or resource group on the specified node.

Note: If conflicting constraints already exist, this preference might not be honored.

You must remember to remove implicit constraints when they are no longer needed, such as after the resource or resource group has successfully moved to the new node. Do the following:

```
ha# crm resource unmove resource_or_resourceGROUP
```

To move the COPAN OpenVault client resource, see "Manually Moving a `copan_ov_client` Resource" on page 178.

Set the Core Membership Timeout Value Appropriately

Set the HA `totem token` (core membership timeout) value to one that is significantly higher than the CXFS heartbeat timeout. The CXFS heartbeat timeout is set by the `mtcp_hb_period` system tunable parameter (which is specified in hundredths of a second). To determine the current setting of `mtcp_hb_period`, use the `sysctl`(8) command. For example:

```
# sysctl -a | grep mtcp_hb_period
kernel.cell.mtcp_hb_period = 500
```

In this case, the CXFS heartbeat timeout is `500` (5 seconds), so you would set the HA `totem token` value to at least `15s`. If `mtcp_hb_period` was set to `6000` (60 seconds), you would use an HA `totem token` value of at least `90s`.

For more information, see the `corosync.conf`(5) man page.

Note: There is an error on the `corosync.conf`(5) man page; the correct file location is `/etc/corosync/corosync.conf`.

Use the Default Consensus Value

By default, the HA `totem consensus` value is 1.2 X the `totem token` value, which is appropriate for most sites. If you choose to set this value explicitly, set it to a value that is no lower than 1.2 X the `totem token` value.

For example, for the `totem token` value of `90s`, the `totem consensus` value is set by default to `108s`. If appropriate, a site could choose to set the value higher (for example, to `135s`).

For more information, see the `corosync.conf`(5) man page.

Upgrade Appropriately

When upgrading the software, follow the procedure in "Performing a Rolling Upgrade" on page 180.

Do Not Use an HA CXFS NFS Edge Server as an NFS Client

Do not use a CXFS NFS edge server node running HA software as an NFS client.

Include PID in Core Files

To better analyze problems, consider adding the following directive to the `/etc/sysctl.conf` file on each node in the HA cluster so that core dump files will include the name of the process ID (PID) that caused the dump and the time when the dump occurred:

```
kernel.core_pattern = core.%p.%t
```

Changes made to `/etc/sysctl.conf` will take effect on the next boot and will be persistent. To make the settings effective immediately for the current session as well, enter the following:

```
ha# echo "core.%p.%t" > /proc/sys/kernel/core_pattern
```

Note: Core files are generally placed in the current working directory (`cwd`) of the process that dumped the core file. For example, to locate the core file for a PID of `25478`:

```
node1# ps -fp 25478
UID        PID  PPID  C STIME TTY          TIME CMD
root     25478 25469  0 Feb02 ?        00:02:40 /usr/lib/heartbeat/stonithd
node1# ls -l /proc/25478/cwd
lrwxrwxrwx 1 root root 0 Mar 2 17:29 /proc/25478/cwd -> /var/lib/heartbeat/cores/root
```

Maintenance Best Practices

This section discusses the following:

- "Questions to Ask Before Performing Maintenance" on page 22
- "Hardware Maintenance" on page 23
- "System Software Updates" on page 23
- "ISSP Software Updates" on page 23
- "Changes Permitted on a Running Resource" on page 23
- "Changes that Require Maintenance Mode" on page 24
- "Changes that Require a Full Cluster Outage" on page 24

Questions to Ask Before Performing Maintenance

Before performing maintenance tasks, answer the following questions:

- How will end users be impacted by the change being proposed?
- Will the change affect the availability of a resource, even briefly?
- How is HA monitoring the resource availability?
- Will the change impact other resources in the HA environment?
- What is the risk of a misstep that could lead to an HA service outage?

- How can the effectiveness of the change be verified?
- What is the change roll-back plan?

Hardware Maintenance

Hardware changes are generally disruptive to the HA environment and always require careful planning. You should consider whether or not the hardware change will also require a software change. In many cases, you must entirely shut down the HA cluster. See "Maintenance with a Full Cluster Outage" on page 189.

System Software Updates

System software updates (such as an operating system upgrade, kernel update, or software patches) are generally disruptive to the HA environment and always require careful planning. In many cases, a full cluster outage is required; see "Maintenance with a Full Cluster Outage" on page 189.

In other cases, an upgrade with the operational HA cluster may be possible; see "Performing a Rolling Upgrade" on page 180.

ISSP Software Updates

Before updating ISSP software, read the release notes and any late-breaking caveats on the Supportfolio download page.

Changes Permitted on a Running Resource

If a resource allows the change without impact to production operation, then the change is generally safe to perform in an HA environment. For example, you can make changes to most DMF configuration parameters or add volumes to an existing OpenVault cartridge group without problems. For more information about which parameters can be changed while DMF is running, see the “Best Practices” chapter of the *DMF 6 Administrator Guide for SGI InfiniteStorage.*

Caution: Changing meta attributes or operation parameters will influence the behavior of the resource or clone and can therefore influence how HA handles the resource or clone. If you make a mistake (such as setting a timeout to `3s` when you meant to change it to `30s`), problems can result.

Changes that Require Maintenance Mode

If a change requires that an individual resource be stopped but does not otherwise impact the rest of the HA cluster, you should put the cluster into maintenance mode before stopping the resource. See "Putting the Cluster into Maintenance Mode" on page 174.

Changes in this category include:

- Any change that requires DMF to be stopped according to the DMF administration guide
- Restarting the OpenVault server when volume usage is inactive

Note: In general, you should not simply unmanage a given resource because that can adversely impact failcounts and cause inappropriate failovers.

Changes that Require a Full Cluster Outage

Many changes that require a resource to be stopped may also be disruptive to the HA cluster and therefore require a full cluster outage. See "Maintenance with a Full Cluster Outage" on page 189.

Changes in this category include:

- Changes to CXFS filesystem mount options
- Changes to NFS export options
- Changes that require extensive testing

Chapter 3

Requirements

This chapter discusses the following requirements for a high-availability (HA) cluster using SGI resource agents:

Note: All of the stop / start requirements for services and resources that are noted in this chapter will be fulfilled if you follow the steps in Chapter 4, "Outline of the Configuration Procedure" on page 39.

- "HA Support Requirements" on page 26
- "Licensing Requirements" on page 26
- "Software Version Requirements" on page 26
- "Hardware Requirements" on page 26
- "System Reset Requirements" on page 27
- "Time Synchronization Requirements" on page 27
- "CXFS NFS Edge-Serving Requirements" on page 27
- "CXFS Requirements" on page 29
- "Local XVM Requirements" on page 31
- "Filesystem Requirements" on page 31
- "Virtual IP Address Requirements" on page 31
- "TMF Requirements" on page 32
- "OpenVault™ Requirements" on page 32
- "COPAN MAID Requirements" on page 33
- "DMF Requirements" on page 35
- "NFS Requirements" on page 37
- "Samba Requirements" on page 37

- "DMF Manager Requirements" on page 38
- "DMF Client SOAP Service Requirements" on page 38

HA Support Requirements

HA may in some cases require the purchase of additional support from SUSE.

Licensing Requirements

All nodes in an HA cluster must have the appropriate software licenses installed. The following software requires licenses if used:

- CXFS
- DMF
- DMF Parallel Data Mover Option

For information about obtaining licenses, see the individual product administration guides.

Software Version Requirements

For any of the SGI resource agents, you must use the corresponding version of SGI software as defined in the *SGI InfiniteStorage Software Platform* release note.

Hardware Requirements

All nodes in an SGI HA cluster must be x86_64 architecture with a BMC supporting the IPMI protocol and administrative privileges.

Note: If you form an HA cluster using only members of a partitioned system with a single power supply, a failure of that power supply may result in failure of the HA cluster. CXFS does not support these members as nodes in the CXFS cluster.

DMF supports only one instance running on a given node in an HA cluster at any given time, thus active/active mode is not a possible configuration. If the cluster also runs CXFS, the DMF server nodes in the cluster must also be CXFS server-capable administration nodes. For additional requirements when using the DMF Parallel Data Mover Option, see *DMF 6 Administrator Guide for SGI InfiniteStorage*.

System Reset Requirements

You must use STONITH node-level fencing to protect data integrity in case of failure. See Chapter 9, "STONITH Examples" on page 165.

STONITH requires the use of a BMC user account with administrative privileges (typically `-U admin -P admin`). For more information, see the `ipmitool`(1) man page and the user guide or quick start guide for your system.

Time Synchronization Requirements

You must configure time synchronization among all cluster nodes.

CXFS NFS Edge-Serving Requirements

CXFS NFS edge-serving in an HA environment has the following requirements:

- NFS version 3.
- An HA cluster of two CXFS client-only nodes. The nodes must run the **SGI CXFS Edge Server** YaST pattern software. See the CXFS release notes for more information.

  Note: There can be multiple two-node HA clusters within one CXFS cluster.

- Due to the way that NLM grace notification is implemented, all of the server-capable administration nodes in the CXFS cluster must run the same version of CXFS in order to use CXFS relocation. This means that if you want to do a CXFS rolling upgrade of the metadata servers while running HA CXFS NFS edge-serving, you must use CXFS recovery and not CXFS relocation.

- On all CXFS NFS edge-server HA cluster systems during HA operation, disable the `cxfs_client` and `nfsserver` services from being started automatically at boot time:

  ```
  ha# chkconfig cxfs_client off
  ha# chkconfig nfsserver off
  ```

  The HA software will control these services.

- The NFS client service on a CXFS NFS edge server does not support monitored locking via `statd`.

- There must be a file located on shared storage on which to keep kernel state information. (In a non-HA cluster, this would be the `/var/lib/nfs/state` file.) The file must be shared as follows:

  – All systems in a single CXFS NFS edge-server HA cluster must share this file

  – If there are multiple CXFS NFS edge-server HA clusters within one CXFS cluster, all of the cluster systems must share this file

  The **statefile** instance attribute for the CXFS client NFS server identifies this file.

- There must be a directory located on shared storage that will be used to store NFS lock state. (In a non-HA cluster, this would be the `/var/lib/nfs/` directory.) The directory must be shared as follows:

  – All systems in a single CXFS NFS edge-server HA cluster must share this directory

  – If there are multiple CXFS NFS edge-server HA clusters within one CXFS cluster, each must have a separate state directory

The **statedir** instance attribute for the CXFS client NFS server identifies this directory.

Also see:

- "Preliminary Best Practices Before Introducing HA" on page 11
- "Instance Attributes for a CXFS Client NFS Server" on page 66

CXFS Requirements

The CXFS resource agent allows you to associate the location of the CXFS metadata server with other products, such as DMF. This section discusses the following:

- "CXFS Server-Capable Administration Nodes" on page 29
- "CXFS Relocation Support" on page 29
- "Applications that Depend Upon CXFS Filesystems" on page 30
- "CXFS and System Reset" on page 30
- "CXFS Start/Stop Issues" on page 30
- "CXFS Volumes and DMF-Managed User Filesystems" on page 31

CXFS Server-Capable Administration Nodes

An HA cluster using the CXFS resource agent must include the server-capable administration nodes that are potential metadata servers for every filesystem that is managed by the CXFS resource agent.

Certain resources (such as DMF) require that the CXFS metadata server and the HA resource be provided by the same node; see "DMF Requirements" on page 35. Other resources (such as NFS and Samba) do not have this requirement, but it may be desirable to enforce it in order to ensure that these resources provide the best performance possible. (Some NFS and Samba workloads can cause significant performance problems when the NFS or Samba resource is located on a node that is not the CXFS metadata server.)

The CXFS server-capable administration nodes in an HA cluster must use a CXFS fail policy of `reset`. You should otherwise configure the CXFS cluster, nodes, and filesystems according to the instructions in the following:

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

CXFS Relocation Support

CXFS relocation is provided automatically by the CXFS resource agent. In a CXFS cluster running HA software, relocation should only be started by using the tools provided with HA software and not by any other method.

Applications that Depend Upon CXFS Filesystems

If an application uses a CXFS filesystem that is managed by HA, that application must also be managed by HA. You must set colocation and start-ordering constraints or ordered resource groups such that:

- The application can only run on the server-capable administration node that is the active CXFS metadata server for the filesystem that it uses.
- The CXFS metadata server will start before the application starts and stop after the application stops

Using a single resource group and configuring in the correct order ensures the proper colocation.

CXFS and System Reset

CXFS server-capable administration nodes must use some sort of system reset in order to prevent conflicts with CXFS I/O fencing methods. You must use one of the following methods:

- Specify the following in the node definition for the server-capable administration nodes:
  - Fail policy that includes `Reset` or `FenceReset`
  - Reset method of `reset`
- Specify a fail policy of `fence,shutdown` and use STONITH for system reset

For more information, see Chapter 9, "STONITH Examples" on page 165.

CXFS Start/Stop Issues

You must start the CXFS cluster `cxfs_cluster` service and CXFS filesystem `cxfs` service before starting the HA `openais` service. The CXFS resource agent will wait for all of the CXFS filesystems to be mounted by CXFS before attempting any relocation. You must adjust the start operation timeout for the CXFS resource agent accordingly.

During failover, resources that colocate with the CXFS metadata server must be stopped before the CXFS resource. If a resource fails to shutdown completely, any files left open on the metadata server will prevent relocation. Therefore, the HA fail policy for any resource that could prevent relocation by holding files open must be

fence and you must configure a STONITH facility according to the requirements for your site. See Chapter 9, "STONITH Examples" on page 165.

In this case, the offending CXFS metadata server will be reset, causing recovery to an alternate node.

CXFS Volumes and DMF-Managed User Filesystems

The CXFS volumes specified for the `cxfs` resource must not include any volumes that represent DMF-managed user filesystems. See "Instance Attributes for CXFS" on page 79.

Local XVM Requirements

All local XVM volumes that are managed by HA must have unique `volname` values.

All local XVM physical volumes (*physvols*) that are managed by HA must have unique `Disk Name` values in their XVM label when compared to all other XVM volumes on the SAN. For example, you cannot have two physvols on the same SAN with the `Disk Name` of `spool`, even if one is foreign.

If you do not have unique values, the following are potential problems:

- HA software may steal the wrong physvol from a system outside of the cluster while I/O is ongoing. This may result in losing data from that system while corrupting the filesystem from the node within the cluster by whom it is stolen.
- General confusion, resulting in node reset.

Filesystem Requirements

For DMF HA purposes, filesystems used by the `Filesystem` resource should use a filesystem type of `xfs`.

Virtual IP Address Requirements

You must allocate a virtual IP address on the subnet used for DMF and OpenVault communication. The address must be a virtual address managed by a community

`IPaddr2` resource within the same resource group as the `openvault` resource. You must also add an associated virtual hostname to your local DNS or to the `/etc/hosts` file on all hosts in the cluster that could be used as a DMF server or as an OpenVault client node.

Each HA node must have a physical Ethernet interface on the same subnet as the virtual IP address defined for the `IPaddr2` resource.

You may use the `IPaddr2` virtual address for other services, such as for accessing DMF Manager or serving NFS. However, if DMF and OpenVault are configured to use a dedicated subnet, you should instead define a second `IPaddr2` address on an appropriate subnet for accessing these services. You should define this `IPaddr2` resource in the same resource group as the `dmfman` resource.

See also "Virtual IP Address Resource" on page 96.

TMF Requirements

All tape devices should be configured as `DOWN` in the `tmf.config` file on all nodes. The loaders should be configured as `UP` but the `tmf` service must be disabled at boot time (`chkconfig tmf off`) for all nodes.The resource agent will start `tmf` and configure the loader up as needed.

Note: If tape drives are defined and used outside of DMF, you must manually start TMF on the inactive server.

OpenVault™ Requirements

If OpenVault is to be used as the DMF mounting service, you must do the following:

- If upgrading to an entirely new root filesystem, as would be required if upgrading from a SLES 10 system, you should create a copy of the OpenVault configuration directory (`/var/opt/openvault`) from the old root before upgrading the OS. You can then reinstall it on the new root so that you do not need to entirely reconfigure OpenVault. See the section about taking appropriate steps when upgrading DMF in the *DMF 6 Administrator Guide for SGI InfiniteStorage*.
- Provide a directory for OpenVault's use within an HA filesystem in the DMF resource group. This is known as the *serverdir directory* (as specified in "OpenVault

Resource" on page 107). The directory will hold OpenVault's database and logs. The directory can be either of the following:

  - Within the root of an HA-managed filesystem dedicated for OpenVault use
  - Within another HA-managed filesystem, such as the filesystem specified by the `HOME_DIR` parameter in the DMF configuration file

  In non-HA configurations, the OpenVault server's files reside in `/var/opt/openvault/server`. During the conversion to HA, OpenVault will move its databases and logs into the specified directory within an HA-managed filesystem and change `/var/opt/openvault/server` to be a symbolic link to that directory.

- Ensure that you **do not** have the `OV_SERVER` parameter set in the `base` object of the DMF configuration file, because in an HA environment the OpenVault server must be the same machine as the DMF server.
- Configure the DMF application instances in OpenVault to use a wildcard ("*") for the hostname and instance name. For more information, see the chapter about mounting service configuration tasks in the *DMF 6 Administrator Guide for SGI InfiniteStorage*.
- On all HA nodes during HA operation, disable the `openvault` service from being started automatically at boot time:

  ```
  ha# chkconfig openvault off
  ```

  The HA software will control this service.

See also "Virtual IP Address Requirements" on page 31.

COPAN MAID Requirements

This section discusses the following:

- "COPAN MAID in Any HA Cluster" on page 34
- "COPAN MAID in a DMF HA Cluster" on page 34
- "COPAN MAID in a Mover-Node HA Cluster" on page 34

COPAN MAID in Any HA Cluster

Using COPAN MAID shelves in any HA cluster requires the following:

- OpenVault must be configured to manage the RAID sets, `lxvm` volumes, and `xfs` filesystems for each shelf
- At any time, only one node (the *owner node*) can manage activity to a given shelf
- Activity to all shelves controlled by a given node must be stopped before moving the control of any one of those shelves to another node

COPAN MAID in a DMF HA Cluster

In addition to the requirements listed in "COPAN MAID in Any HA Cluster" on page 34, using COPAN MAID shelves in an active/passive DMF HA cluster consisting of DMF servers also requires the following:

- All potential DMF server nodes in the HA cluster must have physical connectivity to the shelves
- The active DMF server must be the owner node of all of the shelves
- The OpenVault server resource must be started before the COPAN OpenVault client resource is started
- The COPAN OpenVault client resource must be stopped before the OpenVault server resource is stopped

For suggested resource start/stop order, see Figure 7-1 on page 77.

COPAN MAID in a Mover-Node HA Cluster

In addition to the requirements listed in "COPAN MAID in Any HA Cluster" on page 34, using COPAN MAID shelves in an active-active HA cluster consisting of two parallel data mover nodes also requires the following:

- Both parallel data mover nodes in the HA cluster must have physical connectivity to the shelves
- The parallel data mover nodes that control the shelves cannot also be used for other tape resources

- The CXFS client resource on each parallel data mover node must be started (via a clone) before the COPAN OpenVault client resource is started on those nodes
- The COPAN OpenVault client resource on each parallel data mover node must be stopped before the CXFS client resource is stopped on those nodes
- A parallel data mover node must be configured as the owner node for each shelf. For load-balancing purposes, one mover node will be the default owner of half of the shelves and the other mover node will be the default owner of the remaining shelves.
- On both parallel data mover nodes during HA operation, disable the `cxfs_client` and `openvault` services from being started automatically at boot time:

  ```
  ha# chkconfig cxfs_client off
  ha# chkconfig openvault off
  ```

  The HA software will control these services.

For suggested resource start/stop order, see Figure 7-1 on page 77.

DMF Requirements

Using DMF with HA software requires the following:

- The HA cluster must contain all nodes that could be DMF servers.
- Each DMF server must run the required product and HA software.
- All DMF server nodes in the HA cluster must have connectivity to the same set of libraries and drives. If one node has access to only a subset of the drives, and the DMF server is failed over to that node, DMF would then not be able to access data on volumes left mounted in inaccessible drives.
- All DMF server nodes must have connectivity to all of the CXFS and XFS® filesystems that DMF either depends upon or manages:
  - Each of the local XVM volumes that make up those filesystems must be managed by an `lxvm` resource within the same resource group as the `dmf` resource. Each of the XFS filesystems must be managed by a community `Filesystem` resource in that resource group.

  - Each of the CXFS filesystems (other than DMF-managed user filesystems) must be managed by the `cxfs` resource in that resource group.

  The DMF filesystems to be managed are:

  - The DMF-managed user filesystems (do not include these in the **volnames** attribute list for the `cxfs` resource; see "Instance Attributes for CXFS" on page 79)
  - DMF administrative filesystems specified by the following parameters in the DMF configuration file:
    - `HOME_DIR`
    - `JOURNAL_DIR`
    - `SPOOL_DIR`
    - `TMP_DIR`
    - `MOVE_FS`
    - `CACHE_DIR` for any library servers
    - `STORE_DIRECTORY` for any disk cache manager (DCM) and disk MSPs using local disk storage

  DMF requires independent paths to drives so that they are not fenced by CXFS. The ports for the drive paths on the switch should be masked from I/O fencing in a CXFS configuration.

  The SAN must be zoned so that XVM does not fail over CXFS filesystem I/O to the paths visible through the HBA ports when Fibre Channel port fencing occurs. Therefore, you should use either independent switches or independent switch zones for CXFS/XVM volume paths and DMF drive paths.

  For more information about DMF filesystems, see the *DMF 6 Administrator Guide for SGI InfiniteStorage*.

- The ordering of resources within a resource group containing a `dmf` resource must be such that the `dmf` resource starts after any filesystems it uses are mounted and volume resources it uses are available (and the `dmf` resource must be stopped before those resources are stopped).
- There must be a virtual hostname for use by DMF. See "Virtual IP Address Requirements" on page 31.

- Set the INTERFACE parameter in the node object for each potential DMF server node to the same virtual hostname used for SERVER_NAME in the base object.
- On all HA nodes during HA operation, disable the dmf service from being started automatically at boot time:

  ```
  ha# chkconfig dmf off
  ```

  The HA software will control this service.

Also see:

- "DMF Manager Requirements" on page 38
- "DMF Client SOAP Service Requirements" on page 38

NFS Requirements

On all HA nodes during HA operation, disable the nfsserver service from being started automatically at boot time:

```
ha# chkconfig nfsserver off
```

The HA software will control this service.

Samba Requirements

The /etc/samba and /var/lib/samba directories must be on shared storage. SGI recommends using symbolic links.

On all HA nodes during HA operation, disable the smb and nmb services from being started automatically at boot time:

```
ha# chkconfig smb off
ha# chkconfig nmb off
```

The HA software will control these services.

DMF Manager Requirements

On all HA nodes during HA operation, disable the `dmfman` service from being started automatically at boot time:

```
ha# chkconfig dmfman off
```

The HA software will control this service.

DMF Client SOAP Service Requirements

On all HA nodes during HA operation, disable the `dmfsoap` service from being started automatically at boot time:

```
ha# chkconfig dmfsoap off
```

The HA software will control this service.

Outline of the Configuration Procedure

This chapter summarizes the recommended steps to configure a high-availability (HA) cluster for use with SGI InfiniteStorage products:

- "Prepare for HA" on page 39
- "Install the HA Software" on page 40
- "Enable Multicasting on the Switch" on page 40
- "Configure the Primary Node for HA Using YaST" on page 41
- "Modify the HA Authorization and Configuration Files" on page 42
- "Configure Log and HA Services on Both Nodes" on page 42
- "Configure and Test the SGI Resources" on page 43
- "Put the HA Cluster Into Production Mode" on page 43

This outline uses an example HA cluster with two-nodes: `node1` and `node2`. This procedure uses explicit HA node IDs, but you can choose to automatically generate node IDs if you prefer.

Prepare for HA

To prepare for an HA cluster, do the following:

1. Understand the requirements for the SGI products that you want to include in your HA cluster. See Chapter 3, "Requirements" on page 25.
2. Ensure that you have installed the required SGI products on `node1` and `node2` (from the **SGI ISSP High Availability** YaST pattern) according to the installation procedure in the *SGI InfiniteStorage Software Platform Release Note*.
3. Configure and test each of the standard SGI product services on `node1` before making them highly available. All of the filesystems must be mounted and and all drives and libraries must be accessible on `node1`. See Chapter 5, "Standard Services" on page 45.

4. On both nodes, use the procedures in the following chapters to disable the noted standard services (other than `cxfs` and `cxfs_cluster`) from being started automatically at boot time and then stop the currently running services:

   - Chapter 6, "CXFS NFS Edge-Serving HA Service" on page 53:

     ```
     cxfs_client
     nfsserver
     nmb
     smb
     ```

   - Chapter 7, "DMF HA Service" on page 75:

     ```
     dmf
     dmfman
     dmfsoap
     openvault
     tmf (optional)
     ```

   - Chapter 8, "COPAN MAID HA Service for Mover Nodes" on page 149:

     ```
     cxfs_client
     openvault
     ```

Install the HA Software

Install the OS HA software on `node1` and `node2` as documented in the SUSE *High Availability Guide* provided by the following SUSE website:

http://www.suse.com/documentation/sle_ha/

Enable Multicasting on the Switch

Ensure that the switch supports multicasting and has it enabled. (Some switches disable multicasting by default.)

Configure the Primary Node for HA Using YaST

You must follow the detailed instructions in the SUSE *High Availability Guide* to initialize the cluster and configure `node1`. Also see the information about the setting the password and using the HA GUI (`crm_gui`) in:

- "Preliminary Best Practices Before Introducing HA" on page 11
- "HA Configuration and Testing Best Practices" on page 13

Complete the following set of steps in YaST for `node1`:

1. Set **Bind Network Address** to the network that will support the cluster heartbeat (for example, the CXFS private network, such as `128.162.244.0`), which is different from the IP address to be failed over.
2. Set **Multicast Address** to a multicast address (for example, `226.94.1.1`).
3. Set **Multicast Port** to a multicast port (for example, `5405`).

   Note: Ensure that multiple HA clusters on the same local network use different multicast port numbers.

4. Explicitly set the HA node ID, such as `1` for `node1`. Each node must have a unique HA node ID number. (The HA node ID may be different from the CXFS node ID.)

   Caution: With explicit node IDs, you must not use `csync2` on `node2` (despite the directions in the SUSE *High Availability Guide*) because it will result in duplicate node IDs.

5. Set security on.
6. Select **Generate Auth Key File**. It will be created in `/etc/corosync/authkey`; this process can take several minutes to complete.

Modify the HA Authorization and Configuration Files

Do the following:

1. On `node1`, change the permission on the `/etc/corosync/authkey` and `/etc/corosync/corosync.conf` files to allow read and write permission for the `root` user only:

```
node1# chmod 0600 /etc/corosync/authkey /etc/corosync/corosync.conf
```

2. Copy the `/etc/corosync/authkey` and `/etc/corosync/corosync.conf` files from `node1` to `node2` and preserve their `0600` permission. For example:

```
node1# scp -p /etc/corosync/authkey node2:/etc/corosync/authkey
node1# scp -p /etc/corosync/corosync.conf node2:/etc/corosync/corosync.conf
```

3. Set the `nodeid` in the `/etc/corosync.conf` file on `node2` to a unique value. For example:

```
nodeid:  2
```

Configure Log and HA Services on Both Nodes

On both nodes, enable the `logd` and the `openais` services to be started automatically at boot time and then start them immediately:

- On `node1`:

```
node1# chkconfig logd on
node1# chkconfig openais on

node1# service logd start
node1# service openais start
```

- On `node2`:

```
node2# chkconfig logd on
node2# chkconfig openais on

node2# service logd start
node2# service openais start
```

Configure and Test the SGI Resources

Do the following to configure and test the SGI resources:

1. Test the base HA cluster by running the following command on `node1`, waiting to see both nodes come online (which could take a few minutes):

   ```
   node1# crm status
   ```

2. Disable system reset (which is enabled by default) for testing purposes:

   ```
   node1# crm configure property stonith-enabled=false
   ```

   Note: You will reenable system reset later (in "Put the HA Cluster Into Production Mode" on page 43) after testing all of the SGI resource primitives.

3. Set the correct two-node quorum policy action:

   ```
   node1# crm configure property no-quorum-policy=ignore
   ```

4. Configure and test the resources required for your HA configuration. Proceed to the next resource primitive only if the current resource is behaving as expected, as defined by the documentation. Using the instructions in this guide, you must configure resources in the specific order shown in the following:

   - Chapter 6, "CXFS NFS Edge-Serving HA Service" on page 53
   - Chapter 7, "DMF HA Service" on page 75
   - Chapter 8, "COPAN MAID HA Service for Mover Nodes" on page 149

Put the HA Cluster Into Production Mode

Do the following to put the tested HA cluster into production mode:

1. Create the STONITH facility appropriate for your site, one of:

   - "SGI IPMI STONITH Examples" on page 165
   - "Community IPMI STONITH Examples" on page 168

   Note: The STONITH facility is required to ensure data integrity.

2. Reenable node-level fencing (which was disabled for testing purposes):

   ```
   node1# crm configure property stonith-enabled=true
   ```

3. Ensure that any constraints remaining in the cluster are appropriate for a production environment. To remove any remaining implicit constraints imposed by an administrative `move`, enter the following:

   ```
   node1# crm resource unmove resourceGROUP
   ```

Chapter 5

Standard Services

You should configure and test all standard services before applying high availability. In general, you should do this on one host (known in this guide as *node1* or *pdmn1*). This host will later become a node in the high-availability (HA) cluster, on which all of the filesystems will be mounted and on which all drives and libraries are accessible. If you already have a stable configuration, you can skip the steps in this chapter.

This chapter discusses the following:

- "CXFS NFS Edge-Serving Standard Service" on page 46
- "CXFS Standard Service" on page 47
- "Local XVM Standard Service" on page 47
- "TMF Standard Service" on page 48
- "OpenVault Standard Service" on page 48
- "COPAN MAID Standard Service" on page 49
- "DMF Standard Service" on page 50
- "NFS Standard Service" on page 50
- "Samba Standard Service" on page 51
- "DMF Manager Standard Service" on page 51
- "DMF Client SOAP Standard Service" on page 52

CXFS NFS Edge-Serving Standard Service

Set up the NFS exports in the `/etc/exports` file on both CXFS client-only nodes as you would normally. The `/etc/exports` file should be identical on both nodes.

Note: Be sure to include the `fsid=`*unique_number* export option in order to prevent stale file handles after failover.

To test the CXFS NFS edge-serving standard service, do the following:

1. Run the following command on `node1` to verify that the NFS filesystems are exported:

```
node1# exportfs -v
/work.mynode1   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=8001)
/work.mynode2   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=8002)
/work.mynode3   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=8003)
/work.mynode4   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=8004)
/mirrors        <world>(ro,wdelay,root_squash,no_subtree_check,fsid=8005)
/               <world>(ro,wdelay,root_squash,no_subtree_check,fsid=8006)
```

2. Mount the filesystems on a node that will not be a member of the HA cluster (`otherhost`):

```
otherhost# mount initial:/nfsexportedfilesystem /mnt/test
```

3. Read and write to the NFS-mounted filesystems:

```
otherhost# echo "test data for a test file" > /mnt/test/testFile1A
otherhost# cat /mnt/test/testFile1A
test data for a test file
```

4. Repeat step 1 but execute on `node2`.

5. Repeat steps 2 and 3.

CXFS Standard Service

To configure and test the CXFS standard service before applying high availability, do the following:

1. Configure CXFS on `node1` (which must be a CXFS server-capable administration node), according to the instructions in the following:
   - "CXFS Requirements" on page 29
   - *CXFS 7 Administrator Guide for SGI InfiniteStorage*
   - *CXFS 7 Client-Only Guide for SGI InfiniteStorage*
2. Start the CXFS filesystem service (`cxfs`) and CXFS cluster service (`cxfs_cluster`). For more information, see *CXFS 7 Administrator Guide for SGI InfiniteStorage*.
3. Verify that the filesystem in question mounts on all applicable nodes. For example, use the `cxfs_admin` command:

   ```
   node1# cxfs_admin -c status
   ```

   Note: If you have multiple clusters on the same network, add the `-i` *clustername* option to identify the cluster name. For more information, see the `cxfs_admin`(8) man page.

Local XVM Standard Service

According to the instructions in the *XVM Volume Manager Administrator Guide*, do the following on `node1` for each of the local XVM filesystems that you want to make highly available:

1. Configure the filesystem. Make a note of the name of each physvol that is part of each volume and save it for later.
2. Construct the filesystem using `mkfs`.
3. Mount the filesystem.

To test the local XVM standard service, ensure that you can create and delete files in each of the mounted filesystems.

TMF Standard Service

Configure TMF on `node1` according to the instructions in the *TMF 5 Administrator's Guide for SGI InfiniteStorage* and run the following on `node1`:

```
node1# chkconfig tmf on
```

Note: In the `tmf.config` file, drives in drive groups managed by HA should have `access` configured as `EXCLUSIVE` and should have `status` configured as `DOWN` when TMF starts. Loaders in the `tmf.config` file should have status configured as `UP` when TMF starts.

To test the TMF standard service, do the following:

1. Use `tmstat` to verify that all of the tape drives have a status of `idle` or `assn`:

   ```
   node1# tmstat
   ```

2. Use `tmmls` to verify that all of the loaders have a status of `UP`:

   ```
   node1# tmmls
   ```

OpenVault Standard Service

Configure OpenVault on `node1`, according to the instructions in the *OpenVault Administrator Guide for SGI InfiniteStorage* and, if using the Parallel Data Mover Option, the *DMF 6 Administrator Guide for SGI InfiniteStorage.* This means that you will use the **actual** hostname as reported by the `hostname`(1) command when using `ov_admin`. For the potential DMF servers and any parallel data mover nodes, configure OpenVault library control programs (LCPs) and drive control programs (DCPs) for all local libraries and drives.

Note: Configuration of OpenVault on the alternate DMF server (`node2`) will be done when the conversion to HA is performed.

To test the OpenVault standard service, verify that you can perform operational tasks documented in the OpenVault guide, such as mounting and unmounting of cartridges using the `ov_mount` and `ov_unmount` commands.

For example, in an OpenVault configuration with two drives (`drive0` and `drive1`) where you have configured a volume named `DMF105` for use by DMF, the following sequence of commands will verify that drive `drive0` and the library are working correctly:

```
node1# ov_mount -A dmf -V DMF105 -d drive0
Mounted DMF105 on /var/opt/openvault/clients/handles/An96H0uA3xr0
node1# tsmt status
        Controller: SCSI
        Device: SONY: SDZ-130         0202
        Status: 0x20262
        Drive type: Sony SAIT
        Media : READY, writable, at BOT
node1# ov_stat -d | grep DMF105
drive0              drives     true   false  false    inuse     loaded    ready     true         DMF105S1
node1# ov_unmount -A dmf -V DMF105 -d drive0
Unmounted DMF105
node1# exit
```

Repeat the sequence for `drive1`.

COPAN MAID Standard Service

Configure the following OpenVault components for each COPAN MAID shelf by executing the `ov_shelf`(8) command on `node1` (or `pdmn1`), making `node1` the *owner node* for that shelf, according to the instructions in the *COPAN MAID for DMF Quick Start Guide*:

- One library control program (LCP)
- Up to 16 drive control programs (DCPs)
- One OpenVault drive group

Note: You will not run `ov_shelf` on `node2` at this point. You will do that later in "Creating the OpenVault Components on the Failover Node" on page 119.

If you are using the Parallel Data Mover Option, also see the instructions in *DMF 6 Administrator Guide for SGI InfiniteStorage*.

Note: You will create the OpenVault components on the alternate node later, using the instructions in this guide.

To test the standard service, follow the instructions to test that OpenVault can mount a migration volume, as described in the *COPAN MAID for DMF Quick Start Guide*.

DMF Standard Service

Configure DMF according to the instructions in the *DMF 6 Administrator Guide for SGI InfiniteStorage*.

To test the DMF standard service, do the following:

1. Migrate a few test files:

   ```
   node1# dmput -r files_to_test
   ```

2. Force volumes to be immediately written:

   ```
   node1# dmdidle
   ```

   Wait a bit to allow time for the volume to be written and unmounted.

3. Verify that the volumes are mounted and written successfully.

4. Verify that the volumes can be read and the data can be retrieved:

   ```
   node1# dmget files_to_test
   ```

NFS Standard Service

Set up the NFS exports in the `/etc/exports` file on `node1` as you would normally.

To test the NFS standard service, do the following:

1. Run the following command on `node1` to verify that the NFS filesystems are exported:

```
node1# exportfs -v
/work.mynode1   <world>(rw,wdelay,root_squash,no_subtree_check)
/work.mynode2   <world>(rw,wdelay,root_squash,no_subtree_check)
/work.mynode3   <world>(rw,wdelay,root_squash,no_subtree_check)
/work.mynode4   <world>(rw,wdelay,root_squash,no_subtree_check)
/mirrors        <world>(ro,wdelay,root_squash,no_subtree_check)
/               <world>(ro,wdelay,root_squash,no_subtree_check)
```

2. Mount the filesystems on a node that will not be a member of the HA cluster (`otherhost`):

```
otherhost# mount initial:/nfsexportedfilesystem /mnt/test
```

3. Read and write to the NFS-mounted filesystems:

```
otherhost# echo "test data for a test file" > /mnt/test/testFile1A
otherhost# cat /mnt/test/testFile1A
test data for a test file
```

Samba Standard Service

Set up the Samba standard service on `node1` as you would normally, but place the Samba configuration files and directories on shared storage.

To test the Samba standard service, see the following information:

http://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/install.html

In particular, see the information about the following topics:

- Listing shares available on the server
- Connecting with a UNIX client
- Connecting from a remote SMB client (but not the information about printing)

DMF Manager Standard Service

To verify that standard DMF Manager is operational, start it according to the directions in *DMF 6 Administrator Guide for SGI InfiniteStorage* and access it by pointing your browser to the following address:

`https://`*YOUR_DMF_SERVER*`:1179`

Then verify that you can log in and use DMF Manager, such as by viewing the **Overview** panel.

DMF Client SOAP Standard Service

To verify that the standard DMF client SOAP service is operational, start it according to the directions in *DMF 6 Administrator Guide for SGI InfiniteStorage* and access it by pointing your browser to the following address:

`https://`*YOUR_DMF_SERVER*`:1180/server.php`

Then verify that you can access the GUI and view the WSDL for one of the DMF client functions.

Chapter 6

CXFS NFS Edge-Serving HA Service

This chapter contains the following sections:

- "CXFS NFS Edge-Serving HA Example Procedure" on page 53
- "CXFS Client Resource" on page 62
- "CXFS Client NFS Server Resource" on page 65
- "Virtual IP Address Resource" on page 69
- "CXFS Client NSM Notification Resource" on page 71

Note: The attributes listed in this chapter and the various value recommendations are in support of this example. If you are using the resources in a different manner, you must evaluate whether these recommendations and the use of meta attributes apply to your intended site-specific purpose.

CXFS NFS Edge-Serving HA Example Procedure

Note: In this configuration, each CXFS filesystem is a single point of failure for the HA cluster. Therefore, you may want to consider using a separate HA cluster for each filesystem in order to reduce the possibility of cluster failure while maintaining filesystem bandwidth scalability. However, this also introduces more complexity.

You must determine appropriate monitoring interval and timeout values for your site. See "Avoid Unnecessary Failovers" on page 16.

Figure 6-1 on page 54 shows a map of an example configuration process for CXFS NFS edge-serving in an active/active HA cluster, referring to resource agent type names such as `cxfs-client` and `IPaddr2`.

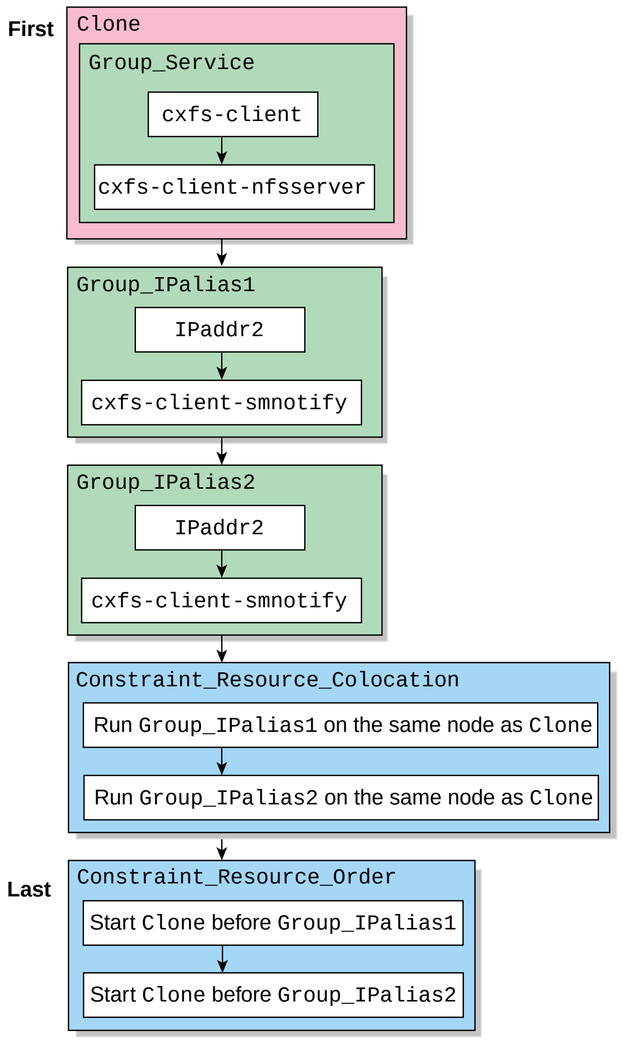
First
Clone
Group_Service
cxfs-client
cxfs-client-nfsserver
Group_IPalias1
IPaddr2
cxfs-client-smnotify
Group_IPalias2
IPaddr2
cxfs-client-smnotify
Constraint_Resource_Colocation
Run Group_IPalias1 on the same node as Clone
Run Group_IPalias2 on the same node as Clone
Last
Constraint_Resource_Order
Start Clone before Group_IPalias1
Start Clone before Group_IPalias2

Figure 6-1 CXFS NFS Edge-Server HA Service Map of Resources

To implement this configuration, use the steps in the following sections:

- "Start the HA GUI" on page 55
- "Create the Clone" on page 55

- "Test the Clone" on page 56
- "Create Two IP Alias Groups" on page 58
- "Create the Constraints" on page 58
- "Test the IP Alias Groups" on page 59

Start the HA GUI

Do the following:

1. Invoke the HA GUI:

   ```
   node1# crm_gui
   ```

2. Log in to the initialized cluster (see Chapter 4, "Outline of the Configuration Procedure" on page 39).

Create the Clone

Do the following to create an anonymous clone containing a group and resources:

1. Select **Resources** in the left-hand navigation panel.
2. Click the **Add** button, select **Clone**, and click **OK**.
3. Enter the ID of the clone, such as `cxfs-nfs-clone`.
4. Select **Stopped** for the **Initial state of resource**, check the **Interleave** option, and click **Forward**.
5. Select the sub-resource **Group** and click **OK**.
6. Enter the ID of the resource group (such as `cxfs-nfs-group`).
7. Select **Defaults to Started or inherit from its parent**.
8. Select the sub-resource **Primitive** and click **OK**.
9. Create the primitives for the following resources:

   Note: Click the **Operations** tab to edit the monitor operations and to add the probe, start, and stop operations as needed for a resource.

   a. "CXFS Client Resource" on page 62

   b. "CXFS Client NFS Server Resource" on page 65

10. Click **Apply** to apply the new group and again to apply the new clone.

Test the Clone

Use the following steps to test the clone:

1. Start the clone. For example:

```
node1# crm resource start cxfs-nfs-clone
```

2. Confirm that the clone has started. For example:

   a. View the status of the cluster on `node1`:

```
node1# crm status
============
Last updated: Tue Mar  8 10:34:02 2011
Stack: openais
Current DC: node1 - partition with quorum
Version: 1.1.2-ecb1e2ea172ba2551f0bd763e557fccde68c849b
2 Nodes configured, 2 expected votes
1 Resources configured.
============
```

   b. Verify that the `cxfs_client` process is running on `node1`:

```
node1# ps -ef | grep cxfs_client
root  11575     1  0 10:32 ?      00:00:00 /usr/cluster/bin/cxfs_client -p /var/run/cxfs_client.pid -i TEST
root  12237  7593  0 10:34 pts/1  00:00:00 grep --color -d skip cxfs_client
```

   Also execute the command on `node2`.

   c. View the status of the NFS daemons on `node1`:

```
node1# rcnfsserver status
Checking for kernel based NFS server: idmapd                      running
 mountd                                                           running
 statd                                                            running
 nfsd                                                             running
```

   Also execute the command on `node2`.

3. Set node2 to standby state to ensure that the resources remain on node1:

   ```
   node1# crm node standby node2
   ```

4. Confirm that node2 is offline and that the resources are off:

   a. View the status of the cluster on node1, which should show that node2 is in standby state:

      ```
      node1# crm status
      ============
      Last updated: Tue Mar  8 10:36:35 2011
      Stack: openais
      Current DC: node1 - partition with quorum
      Version: 1.1.2-ecb1e2ea172ba2551f0bd763e557fccde68c849b
      2 Nodes configured, 2 expected votes
      1 Resources configured.
      ============

      Node node2: standby
      Online: [ node1 ]

        Clone Set: cxfs-nfs-clone [cxfs-nfs-group]
            Started: [ node1 ]
            Stopped: [ cxfs-nfs-group:1 ]
      ```

   b. Verify that the cxfs_client process is not running on node2 by executing the ps(1) command on node2 (that is, there should be no output):

```
node2# ps -ef | grep cxfs_client
node2#
```

   c. View the status of the NFS daemons on node2, which should show that statd is dead and nfsd is unused:

```
node2# rcnfsserver status
Checking for kernel based NFS server: idmapd                      running
 mountd                                                           unused
 statd                                                            dead
 nfsd                                                             unused
```

5. Return node2 to online status:

   ```
   node1# crm node online node2
   ```

6. Confirm that the clone has returned to normal status, as described in step 2.

Create Two IP Alias Groups

Do the following to create two IP alias groups, one for each rack:

1. Select **Resources** in the left-hand navigation panel.
2. Click the **Add** button, select **Group**, and click **OK**.
3. Enter the ID of the group, such as `ipalias-group-1`. Use the default initial state of **Stopped** and click **Forward**.
4. Select the sub-resource **Primitive** and click **OK**.
5. Create the primitives for the following resources:

   Note: You will just create the primitives in this step. You will not test them until later, in "Test the IP Alias Groups" on page 59.

   a. "Virtual IP Address Resource" on page 69

   b. "CXFS Client NSM Notification Resource" on page 71
6. Click **Cancel** to stop adding primitives.
7. Repeat steps 1 through 6 for the second IP alias.
8. After completing both IP aliases, click **Apply** to apply the resource group.

Create the Constraints

Do the following to create the constraints:

1. Select **Constraints** in the left-hand navigation panel.
2. Click the **Add** button, select **Resource Colocation**, and click **OK**.
3. Create two colocation constraints so that the virtual IP addresses must run on a system that has NFS, one constraint for each rack:

   a. Enter the **ID** of the constraint for the IP address, such as `ipalias-rack1-with-nfs`.

b. For **Resource**, select the primitive created in step 5a of "Create Two IP Alias Groups" on page 58 above, such as **ipalias-rack1**.

c. For **With Resource**, select the clone created in step 3 of "Create the Clone" on page 55 above, such as **cxfs-nfs-clone**.

d. For **Score**, select **INFINITY**.

e. Click **OK**.

f. Repeat steps 3a through 3e to create the colocation constraint for the second rack.

4. Click the **Add** button, select **Resource Order**, and click **OK**.

5. Create two resource order constraints so that the clone will be started before the IP addresses and notifications, one constraint for each rack:

   a. Enter the **ID** of the constraint for the IP address, such as `nfs-before-ipalias-rack1`.

   b. For **First**, select the name of the clone created in step 3 of "Create the Clone" on page 55 above, such as **cxfs-nfs-clone**.

   c. For **Then**, select the group name defined in step 3 of "Create Two IP Alias Groups" on page 58 above, such as **ipalias-group-1**.

   d. Under **Optional**, set **score** to **INFINITY**.

   e. Click **OK**.

   f. Repeat steps 5a through 5e to create the resource order constraint for the second rack.

Test the IP Alias Groups

To test the IP alias groups, do the following:

1. Start the group. For example, to start `ipalias-group-1`:

```
node1# crm resource start ipalias-group-1
```

2. Test the virtual IP address resource within the group:

a. Verify that the IP address is configured correctly on `node1`:

```
node1# ip -o addr show | grep 128.162.244.240
4: eth2    inet 128.162.244.240/24 brd 128.162.244.255 scope global secondary eth2
```

b. Verify that `node2` does not accept the IP address packets. For example, run the following command on `node2` (the output should be `0`):

```
node2# ip -o addr show | grep -c 128.162.244.240
0
```

c. Connect to the virtual address using `ssh` or `telnet` and verify that the IP address is being served by the correct system. For example, for the IP address `128.162.244.240` and the machine named `node1`:

```
nfsclient# ssh root@128.162.244.240
Last login: Mon Jul 14 10:34:58 2008 from mynode.mycompany.com
node1# uname -n
node1
```

d. Move the resource group containing the `IPaddr2` resource from `node1` to `node2`:

```
node1# crm resource move ipalias-group-1 node2
```

e. Verify that the IP address is configured correctly on `node2`:

```
node2# ip -o addr show | grep 128.162.244.240
4: eth2    inet 128.162.244.240/24 brd 128.162.244.255 scope global secondary eth2
```

f. Verify that `node1` does not accept the IP address packets by running the following command on `node1` (the output should be `0`):

```
node1# ip -o addr show | grep -c 128.162.244.240
0
```

g. Connect to the virtual address using `ssh` or `telnet` and verify that the IP address is being served by the correct system. For example, for the IP address `128.162.244.240` and the machine named `node2`:

```
nfsclient# ssh root@128.162.244.240
Last login: Mon Jul 14 10:34:58 2008 from mynode.mycompany.com
node2# uname -n
node2
```

h. Move the resource group containing the `IPaddr2` resource back to `node1`:

```
node1# crm resource move ipalias-group-1 node1
```

   i. Test again as in steps 2a-2c above.

   j. Remove the implicit location constraints imposed by the administrative `move` command above:

```
node1# crm resource unmove ipalias-group-1
```

3. Repeat steps 1 and 2 for the other group, such as `ipalias-group-2`.

4. Test the CXFS client NSM notification resource within the group:

   a. Mount the filesystem via each **ipalias hostname** on a system that is outside the HA cluster (for example, `nfsclient`). For example:

```
nfsclient:~ # mount hostalias1://mnt/cxfsvol1 /hostalias1
nfsclient:~ # mount hostalias2://mnt/cxfsvol1 /hostalias2
```

   b. Turn on Network Lock Manager debugging on the NFS client:

```
nfsclient:~ # echo 65534 > /proc/sys/sunrpc/nlm_debug
```

   c. Acquire locks:

```
nfsclient:/hostalias1 # touch file
nfsclient:/hostalias1 # flock -x file -c "sleep 1000000" &
nfsclient:/hostalias2 # touch file2
nfsclient:/hostalias2 # flock -x file2 -c "sleep 1000000" &
```

   d. Check in the shared `sm-notify` **statedir** directory on the NFS server for resources `hostalias1` and `hostalias2` to ensure that a file has been created by `statd`. The name should be the hostname of the node on which you have taken the locks.

      If the file is not present, it indicates a misconfiguration of name resolution. Ensure that fully qualified domain name entries for each NFS client are present in `/etc/hosts` on each NFS server. (If the `/etc/hosts` file is not present, NSM reboot notification will not be sent to the client and locks will not be reclaimed.)

   e. On the NFS clients, check in the `/var/lib/nfs/sm` for a filename that is the fully qualified domain name of each server from which you have requested locks. If this file is not present, NSM reboot notification will be rejected by the client. (The client must mount the **ipalias** node, such as `hostalias1`, by hostname and not by the IP address in order for this to work.)

f. Make `node1` be in `standby` state:

```
node1# crm node standby node1
```

g. Verify that both of the IP aliases are now on `node2`:

```
node2# ip addr
```

h. Verify that the `/var/log/messages` file on the NFS client (`nfsclient`) contains a message about reclaiming locks for every **ipalias hostname** on which you have taken locks via NFS. (The two `statd` processes for the HA cluster share the same state directory, specified by the **statedir** instance attribute. NSM reboot notification will be sent to clients for all IP aliases in the cluster, so you will see messages for all IP aliases that have been mounted by the client.) For example:

```
Jul 30 13:40:46 nfsclient kernel: NLM: done reclaiming locks for host hostalias2
Jul 30 13:40:49 nfsclient kernel: NLM: done reclaiming locks for host hostalias1
```

i. Make `node1` active again:

```
node1# crm node online node1
```

5. Test the other group.

CXFS Client Resource

This section discusses the following:

- "Configuring the CXFS Client for HA" on page 62
- "Creating the CXFS Client Primitive" on page 63

Configuring the CXFS Client for HA

To configure the CXFS client for HA, do the following:

1. On both nodes, disable the `cxfs_client` service from being started automatically at boot time:

   - On `node1`:

```
node1# chkconfig cxfs_client off
```

- On node2:

  ```
  node2# chkconfig cxfs_client off
  ```

2. Add the CXFS client NFS resource primitive. See "Creating the CXFS Client Primitive" on page 63.

Creating the CXFS Client Primitive

Use the values shown in the following sections when adding a CXFS client resource primitive.

Note: There are no meta attributes for this primitive in this example procedure because it is part of a clone resource that should always restart locally.

Required Fields for a CXFS Client

ID	*Unique ID such as* `cxfs-client`
Class	**ocf**
Provider	**sgi**
Type	**cxfs-client**

Instance Attributes for a CXFS Client

volnames	*Comma-separated list of XVM volume names containing CXFS filesystems that are to be served via NFS or used to store the NFS state, such as:* `cxfsvol1,cxfsvol2`

Probe Monitor Operation for a CXFS Client

Note: Click the **Operations** tab to edit the monitor operations and to add the probe, start, and stop operations as needed for a resource.

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**

Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Monitor Operation for a CXFS Client

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `120s`
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

The monitor operation does the following:

- Verifies that each volume in **volnames** is mounted by checking `/proc/mounts`
- Verifies that the volumes in **volnames** are online by executing the following command for each volume:

  ```
  xvm show -v vol/volname
  ```

- Fails if the CXFS client does not start

Start Operation for a CXFS Client

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `600s`
Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Starts the CXFS client by calling the following:

  ```
  /etc/init.d/cxfs_client start
  ```

- Checks the `/proc/mounts` file until all volumes in **volnames** are mounted
- Fails if the CXFS client fails to start

Stop Operation for a CXFS Client

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `600s`
Optional > On Fail	**fence**

The stop operation does the following:

- Stops the CXFS client by calling the following:

  ```
  /etc/init.d/cxfs_client stop
  ```

- Fails if the CXFS client fails to stop

Note: Using the example procedure in this guide, you should go on to add the primitive for "CXFS Client NFS Server Resource" on page 65 before testing the clone. You will test the resources later, after completing the clone.

CXFS Client NFS Server Resource

This section discusses the following:

- "Configuring CXFS Client NFS for HA" on page 65
- "Creating the CXFS Client NFS Server Primitive" on page 66

Configuring CXFS Client NFS for HA

To configure CXFS client NFS for HA, do the following:

1. Copy the `/etc/exports` entries that you would like to make highly available from `node1` to the `/etc/exports` file on `node2`.

   Note: Be sure to include the `fsid=`*unique_number* export option in order to prevent stale file handles after failover. All matching exports should have the same `fsid=`*unique_number* value on all CXFS NFS edge-server nodes.

   ---

2. On both nodes, disable the `nfsserver` service from being started automatically at boot time:

   - On `node1`:

     ```
     node1# chkconfig nfsserver off
     ```

   - On `node2`:

     ```
     node2# chkconfig nfsserver off
     ```

3. Add the CXFS client NFS resource primitive. See "Creating the CXFS Client NFS Server Primitive" on page 66.

Creating the CXFS Client NFS Server Primitive

Use the values shown in the following sections when adding a CXFS client NFS server resource primitive. (There are no meta attributes for this primitive in this example procedure because it is part of a clone resource that should always restart locally.)

Required Fields for a CXFS Client NFS Server

ID	*Unique ID such as* `cxfs-client-nfsserver`
Class	**ocf**
Provider	**sgi**
Type	**cxfs-client-nfsserver**

Instance Attributes for a CXFS Client NFS Server

nfs_init_script	*Location of the NFS initialization script, such as:* `/etc/init.d/nfsserver`
statedir	*Directory located on the NFS filesystem used to store NFS state (equivalent to* `/var/lib/nfs/` *in a nonclustered configuration), such as:* `/mnt/cxfsvol2/statd/nfs2-nfs3`

Note: There must be a unique **statedir** value for each HA cluster within a CXFS cluster.

statefile	*Filename located on the NFS state filesystem (equivalent to* `/var/lib/nfs/state` *in a nonclustered configuration), such as:* `/mnt/cxfsvol2/statd/state` **Note:** A single **statefile** value is shared among all HA clusters within a CXFS cluster.
volnames	*Comma-separated list of XVM volume names containing CXFS filesystems that are to be served via NFS, such as:* `cxfsvol1`

Probe Monitor Operation for a CXFS Client NFS Server

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Monitor Operation for a CXFS Client NFS Server

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `120s`
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

The monitor operation does the following:

- Verifies the status of the NFS server by calling the status action of the **nfs_init_script**, such as:

  ```
  /etc/init.d/nfsserver status
  ```

- Fails if the NFS server is not running

Start Operation for a CXFS Client NFS Server

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `300s`
Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Creates the **statedir** directory and **statefile** as needed
- Updates `/etc/sysconfig.nfs` to set the following:
  - **statd_options** to `-p` *statedir* `-s` *statefile*
  - **start_smnotify** to `no`
- Enables NLM grace notification for all volumes in **volnames**
- Starts the NFS server by calling the start action of **nfs_init_script**, such as:

  ```
  /etc/init.d/nfsserver start
  ```
- Fails if the NFS server does not start or if the NLM grace notification cannot be enabled

Stop Operation for CXFS Client NFS Server

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `600s`
Optional > On Fail	**fence**

The stop operation does the following:

- Stops the NFS server by calling the stop action of **nfs_init_script**, such as:

  ```
  /etc/init.d/nfsserver stop
  ```
- Disables NLM grace notification for all volumes in **volnames**
- Fails if the NFS server does not stop or if the NLM grace notification cannot be disabled

Note: Using the example procedure in this guide, you should return to step 10 of "Create the Clone" on page 55.

Virtual IP Address Resource

This section discusses creating the virtual IP address primitive. You will test it as part of testing the group.

Creating the Virtual IP Address Primitive

Use the values shown in the following sections when adding a virtual IP address resource primitive.

Required Fields for a Virtual IP Address

ID	*Unique ID such as* `ipalias-rack1`
Class	**ocf**
Provider	**heartbeat**
Type	**IPaddr2**

Instance Attributes for a Virtual IP Address

ip	*IP address of the virtual channel, such as* `128.162.244.240`
nic	*(Optional) Network interface card that will service the virtual IP address, such as* `eth0`
cidr_netmask	*(Optional) Short-notation network mask, which should match the subnet size at the site, such as* `24`
broadcast	*(Optional) Broadcast IP address, such as* `128.162.244.255`

Note: If you do not specify values for **nic**, **cidr_netmask**, and **broadcast**, appropriate values will be determined automatically.

Meta Attributes for a Virtual IP Address

resource-stickiness	**1**
migration_threshold	**1**

Probe Monitor Operation for a Virtual IP Address

Note: Defining a monitor operation (other than a probe operation) on an `IPaddr2` resource is normally unnecessary.

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Start Operation for a Virtual IP Address

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `90s`
Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Establishes the IP alias on the specified NIC
- Fails if the IP alias is not established

Stop Operation for a Virtual IP Address

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `100s`

Optional > On Fail	**fence**

The stop operation does the following:

- Removes the IP alias from the specified NIC
- Fails if the IP alias is not removed

Note: Using the example procedure in this guide, you should go on to "Creating the CXFS Client NSM Notification Primitive" on page 71. You will test the resources later.

CXFS Client NSM Notification Resource

This section discusses creating the CXFS client NSM notification primitive. You will test it as part of testing the group.

Creating the CXFS Client NSM Notification Primitive

Use the values shown in the following sections when adding a CXFS client Network Status Monitor (NSM) notification resource primitive.

Required Fields for a CXFS Client NSM Notification

ID	*Unique ID such as* `smnotify-for-rack1`
Class	**ocf**
Provider	**sgi**
Type	**cxfs-client-smnotify**

Instance Attributes for a CXFS Client NSM Notification

ipalias	*IP address of the IP alias associated with the NFS client lock state, which is reclaimed by the NFS client from the NFS server when it receives the NSM reboot notification that is initiated by the* `cxfs-client-smnotify` *resource agent, for example for hostalias1* `128.162.244.244`
statedir	*Identical to the* **statedir** *value specified above for* **cxfs-client-nfsserver** (see "Instance Attributes for a CXFS Client NFS Server" on page 66)

statefile	*Identical to the* **statefile** *value specified above for* **cxfs-client-nfsserver**
pidfile	*Filename located on the NFS state filesystem that specifies the process ID, such as:* `/mnt/cxfsvol2/statd/smnotify-rack1.pid` **Note:** There must be a separate file and unique **pidfile** value for each **cxfs-client-smnotify** primitive.
gracedir	*Directory on the NFS state filesystem that specifies the file containing the grace-period state, such as:* `/mnt/cxfsvol2/grace` **Note:** A single **gracedir** value is shared among all HA clusters within a CXFS cluster.
hostname	*Hostname of* **ipalias**, *which must match what is in* `/etc/hosts` *or be resolvable with DNS*
seconds	*The number of seconds before the grace period expires (must be the same on all cxfs-client-smnotify resources in the cluster), such as* `120` **Note:** This value is always in seconds, unlike other timeout values, so you must not include the `s` identifier.
volnames	*Comma-separated list of all volumes that will be served via* **ipalias**, *such as:* `cxfsvol1`

Meta Attributes for a CXFS Client NSM Notification

resource-stickiness	**1**
migration_threshold	**1**

Probe Monitor Operation for a CXFS Client NSM Notification

ID	*(Generated based on the primitive name and interval)*

name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Monitor Operation for a CXFS Client NSM Notification

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `120s`
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

The monitor operation does the following:

- Verifies that `sm-notify` ran successfully or is still running
- Fails if `sm-notify` exited with an error

Start Operation for a CXFS Client NSM Notification

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `60s`
Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Ends any active NLM grace period for the IP alias
- Runs `sm-notify` to send out an NSM reboot notification to clients
- Fails if `sm-notify` returns an error

Stop Operation for a CXFS Client NSM Notification

ID	*(Generated based on the primitive name and interval)*
name	**stop**

Timeout	*Timeout, such as* `60s`
Optional > On Fail	**fence**

The stop operation does the following:

- Starts an NLM grace period for the IP alias
- Drops all locks associated with the IP alias
- Fails if locks cannot be dropped

Note: Using the example procedure in this guide, you should go back to step 6 of "Create Two IP Alias Groups" on page 58. You will test the resources later.

DMF HA Service

This chapter contains the following sections:

Note: The attributes listed in this chapter and the various value recommendations are in support of this example. If you are using the resources in a different manner, you must evaluate whether these recommendations and the use of meta attributes apply to your intended site-specific purpose.

DMF HA Example Procedure

Note: You must determine appropriate monitoring interval and timeout values for your site. See "Avoid Unnecessary Failovers" on page 16.

When you create the resource group for the DMF HA service, you must also configure and test the first resource primitive at the same time. This first primitive must be for either CXFS or local XVM.

Figure 7-1 on page 77 shows a map of an example configuration process for DMF HA in a two-node active/passive HA cluster (`node1` and `node2`), referring to resource agent type names such as `lxvm` and `IPaddr2`. This map also describes the start/stop order for resources.

You must configure a resource group and then add and test resource primitives in the order shown in this chapter, skipping products that do not apply to your site.

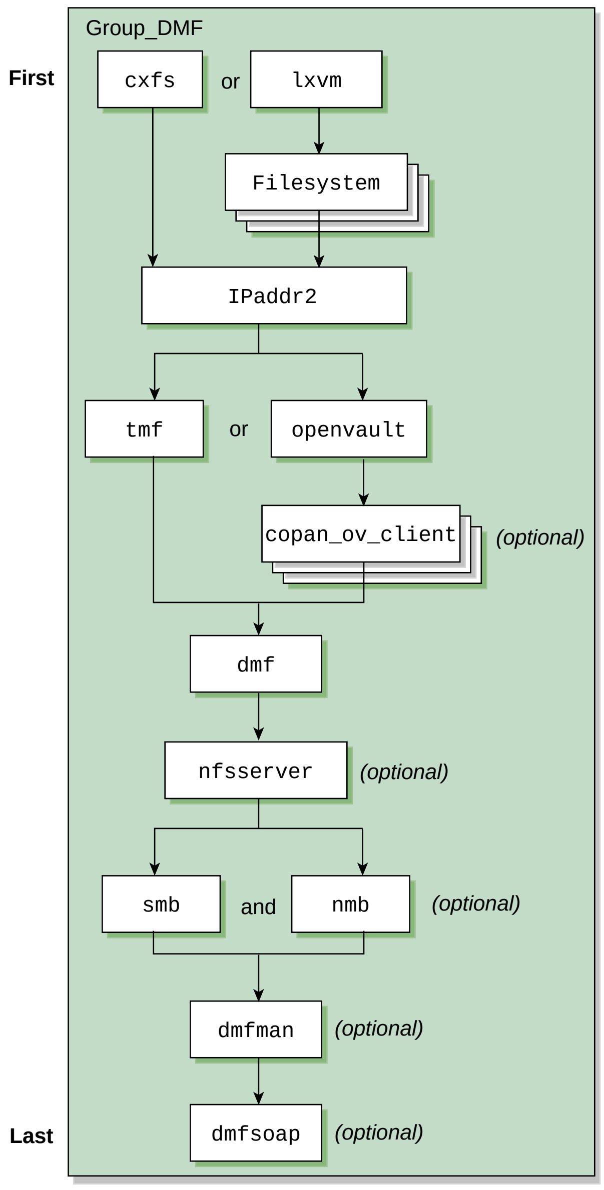

Figure 7-1 DMF HA Service Map of Resources

To create the resource group (referred to in the examples in this guide as `dmfGroup`), do the following:

1. Invoke the HA GUI:

   ```
   node1# crm_gui
   ```

   See the information about setting the password and using the HA GUI in:

   - "Preliminary Best Practices Before Introducing HA" on page 11
   - "HA Configuration and Testing Best Practices" on page 13

2. Log in to the initialized cluster (see Chapter 4, "Outline of the Configuration Procedure" on page 39).
3. Select **Resources** in the left-hand navigation panel.
4. Click the **Add** button, select **Group**, and click **OK**.
5. Enter the ID of the resource group (such as `dmfGroup`).
6. Set the **target-role** meta attribute for `dmfGroup` to **Started** and click **Forward**.
7. Select **OK** to add a **Primitive**. Add and test the CXFS or local XVM primitive, according to the steps described in either:

   - "CXFS Resource" on page 79
   - "Local XVM Resource" on page 82 and "Filesystem Resources" on page 86

   Note: Click the **Operations** tab to edit the monitor operations and to add the probe, start, and stop operations as needed for a resource.

   ---

8. Add additional primitives for the other resources that should be part of `dmfGroup`, in the order shown in this guide:

   a. "Virtual IP Address Resource" on page 96

   b. A mounting service, either:

   - "TMF Resource" on page 100
   - "OpenVault Resource" on page 107 and (optionally) "COPAN MAID OpenVault Client Resource" on page 119

c. "DMF Resource" on page 124

d. "NFS Resource" on page 131 (optional)

e. "Samba Resources" on page 135 (optional)

f. "DMF Manager Resource" on page 141 (optional)

g. "DMF Client SOAP Service Resource" on page 145 (optional)

CXFS Resource

This section discusses examples of the following:

- "Creating the CXFS Primitive" on page 79
- "Testing the CXFS Resource" on page 81

Creating the CXFS Primitive

Required Fields for CXFS

ID	*Unique ID such as* `cxfs`
Class	**ocf**
Provider	**sgi**
Type	**cxfs**

Instance Attributes for CXFS

volnames	*Comma-separated list of CXFS volume names (under* `/dev/cxvm`*, excluding any volumes that represent DMF-managed user filesystems), such as:* `cxfsvol1,cxfsvol2`

Meta Attributes for CXFS

resource-stickiness	**1**
migration_threshold	**1**

Probe Operation for CXFS

Note: Click the **Operations** tab to edit the monitor operations and to add the probe, start, and stop operations as needed for a resource.

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `120s`

The probe operation checks to see if the resource is already running.

Monitor Operation for CXFS

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `120s`
Timeout	*Timeout, such as* `120s`
Optional > On Fail	**restart**

The monitor operation does the following:

- Verifies that each volume in **volnames** is mounted by checking `/proc/mounts`
- Verifies that each volume in **volnames** is owned by the local node according to the `clconf_info` output
- Fails if a volume in **volnames** is not mounted or not owned by the local system

Start Operation for CXFS

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `600s`

Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Waits until all volumes in **volnames** are mounted by checking `/proc/mounts`
- Relocates the metadata server for all volumes in **volnames**
- Waits for all volumes in **volnames** to be owned by the local node according to `clconf_info` output
- Never explicitly fails, but can time out

Stop Operation for CXFS

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `600s`
Optional > On Fail	**fence**

The stop operation never explicitly fails, but can time out.

Testing the CXFS Resource

To test the `cxfs` resource, do the following:

1. Verify that CXFS is working on `node1`. For example:

   a. Verify that all of the CXFS filesystems are mounted and accessible:

   ```
   node1# df -lh
   ```

   b. Display the current metadata server for the filesystems:

   Note: If you have multiple clusters on the same network, add the `-i` *clustername* option to identify the cluster name. For more information, see the `cxfs_admin`(8) man page.

   ```
   node1# /usr/cluster/bin/cxfs_admin -c "show server"
   ```

Note: After a `cxfs` primitive has been added to a resource group's configuration, moving that resource group will unmount the filesystem defined in the primitive. This will result in killing any process that has that filesystem in the path of its current working directory.

---

2. Move the resource group containing the `cxfs` resource to `node2`:

   ```
   node1# crm resource move dmfGroup node2
   ```

3. Verify that CXFS is working on `node2`:

   ```
   node2# df -lh
   node2# /usr/cluster/bin/cxfs_admin -c "show server"
   ```

4. Move the resource group containing the `cxfs` resource back to `node1`:

   ```
   node1# crm resource move dmfGroup node1
   ```

5. Verify that CXFS is working again on `node1`:

   ```
   node1# df -lh
   node1# /usr/cluster/bin/cxfs_admin -c "show server"
   ```

6. Remove the implicit location constraints imposed by the administrative `move` command above:

   ```
   node1# crm resource unmove dmfGroup
   ```

Local XVM Resource

This section discusses examples of the following:

- "Creating the Local XVM Primitive" on page 83
- "Testing the Local XVM Resource" on page 85

Creating the Local XVM Primitive

Required Fields for Local XVM

ID	*Unique ID such as* `local_xvm`
Class	**ocf**
Provider	**sgi**
Type	**lxvm**

Instance Attributes for Local XVM

volnames *Comma-separated list of local XVM volume names (under* `/dev/lxvm`*) to monitor, such as:*

```
openvault,home,journals,spool,move,tmp,diskmsp,dmfusr1,dmfusr3
```

physvols *Comma-separated list of the physical volumes (physvols) for the resource agent to steal, such as:*

```
myCluster,myClusterStripe1,myClusterStripe2
```

Note: physvols must contain all of the physical volumes for every logical volume listed in **volnames**. All physical disks that belong to a logical volume in an HA cluster must be completely dedicated to that logical volume and no other.

Meta Attributes for Local XVM

resource-stickiness	**1**
migration_threshold	**1**

Probe Monitor Operation for Local XVM

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Monitor Operation for Local XVM

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `120s`
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

The monitor operation does the following:

- Verifies that all volumes in **volnames** are online
- Fails if any volume in **volnames** is not online

Start Operation for Local XVM

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `60s`

Note: A 60-second **start** timeout should be sufficient in most cases, but sites with large disk configurations may need to adjust this value. You should usually use the same **timeout** value for **start** and **stop**.

Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Steals all physical volumes in **physvols** that are not already owned by the local system
- Verifies that all volumes in **volnames** are online
- Probes paths for all local XVM devices
- Switches to preferred paths for all local XVM devices
- Fails if any volume in **volnames** does not come online

Stop Operation for Local XVM

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout*, *such as* `60s`
Optional > On Fail	**fence**

The stop operation does the following:

- Gives all physical volumes in **physvols** to a pseudo-cluster whose ID is of the form `OCF-`*host*`-`*pid*, which allows the `lxvm` resource agent to identify the filesystems that it must steal when it becomes active
- Fails if any physical volume in **physvols** could not be given away

Testing the Local XVM Resource

To test the `lxvm` resource, do the following:

1. On `node1`, unmount all of the filesystems for which a `Filesystem` primitive will be defined (in "Filesystem Resources" on page 86).
2. Move the resource group containing the `lxvm` resource from `node1` to `node2`:

   ```
   node1# crm resource move dmfGroup node2
   ```

   Note: If the timeout is too short for a **start** operation, the `crm status` and `crm_verify -LV` output and the `/var/log/messages` file will have an entry that refers to the action being "`Timed Out`". For example (line breaks shown here for readability):

   ```
   node1# crm status | grep Timed
       lxvm_start_0 (node=node1, call=222, rc=-2): Timed Out

   node1# crm_verify -LV 2>&1 | grep Timed
   crm_verify[147386]: 2008/07/23_14:36:34 WARN: unpack_rsc_op:
     Processing failed op lxvm_start_0 on node1: Timed Out
   ```

3. Verify that the local XVM volumes are visible and online on `node2`:

   ```
   node2# xvm -d local show vol
   ```

4. On `node2`, unmount all of the filesystems for which a `Filesystem` primitive will be defined (in "Filesystem Resources" on page 86).

5. Move the resource group containing the `lxvm` resource back to `node1`:

   ```
   node1# crm resource move dmfGroup node1
   ```

6. Verify that the local XVM volumes are visible and online on `node1`:

   ```
   node1# xvm -d local show vol
   ```

7. Remove the implicit location constraints generated by the administrative `move` command above:

   ```
   node1# crm resource unmove dmfGroup
   ```

Filesystem Resources

This section discusses examples of the following:

- "Filesystems Supported" on page 87
- "Configuring a DMF-Managed User Filesystem or DMF Administrative Filesystem for HA" on page 88
- "Creating a DMF-Managed User Filesystem Primitive" on page 88
- "Creating a DMF Administrative Filesystem Primitive" on page 90
- "Creating a Dedicated OpenVault Server Filesystem Primitive *(Optional)*" on page 93
- "Testing Filesystem Resources" on page 95

Filesystems Supported

In this release, SGI supports the following types of filesystems for DMF HA:

- DMF-managed user filesystems

  Note: You must specify the `dmi` and `mtpt` mount options when configuring a DMF-managed user filesystem.

- DMF administrative filesystems specified by the following parameters in the DMF configuration file (`/etc/dmf/dmf.conf`):
  - `HOME_DIR`
  - `JOURNAL_DIR`
  - `SPOOL_DIR`
  - `TMP_DIR`
  - `MOVE_FS` (optional)
  - `CACHE_DIR` for any library servers
  - `STORE_DIRECTORY` for a disk cache manager (DCM) media-specific process (MSP) or disk MSP using local disk storage

  Note: The following DMF administrative filesystems require mount options:

  - *MOVE_FS* requires `dmi` and `mtpt`
  - *STORE_DIRECTORY* for a DCM MSP requires `dirsync`, `dmi`, and `mtpt`
  - *STORE_DIRECTORY* for a disk MSP requires `dirsync`

- *(Optional)* OpenVault server filesystem
- *(Optional)* Any additional HA filesystems that are not managed by DMF; for example, other NFS-exported filesystems that are not under DMF control

All of the above filesystems should be configured as locally mounted XFS filesystems using the following resources:

- Local XVM resource
- Community-provided `Filesystem` resource

Configuring a DMF-Managed User Filesystem or DMF Administrative Filesystem for HA

To configure a DMF-managed user filesystem or DMF administrative filesystem for HA, do the following:

- Edit `/etc/fstab` and remove all of the filesystems that you will manage with HA
- Add a `Filesystem` primitive using the values show in one of the following, as appropriate:
  - "Creating a DMF-Managed User Filesystem Primitive" on page 88
  - "Creating a DMF Administrative Filesystem Primitive" on page 90
  - "Creating a Dedicated OpenVault Server Filesystem Primitive *(Optional)*" on page 93

Creating a DMF-Managed User Filesystem Primitive

Required Fields for a DMF-Managed User Filesystem

ID	*Unique ID such as* `dmfusrlfs`
Class	**ocf**
Provider	**heartbeat**
Type	**Filesystem**

Instance Attributes for a DMF-Managed User Filesystem

device	*The* `/dev/lxvm` *volume name of the DMF filesystem device, such as* `/dev/lxvm/dmfusr1`
directory	*The mount point for the DMF filesystem, such as* `/dmfusr1`
options	*The mount options*

Note: The value of the `mtpt` mount option must match the value used for the **directory** attribute, such as `/dmfusr1`.

fstype	**xfs**

Meta Attributes for a DMF-Managed User Filesystem

resource-stickiness	**1**
migration_threshold	**1**

Probe Monitor Operation for a DMF-Managed User Filesystem

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Monitor Operation for a DMF-Managed User Filesystem

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `120s`
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

The monitor operation does the following:

- Checks `/proc/mounts`, `/etc/mtab`, or the output of the `mount` command for the existence of the filesystem
- Fails if the filesystems is not mounted

Start Operation for a DMF-Managed User Filesystem

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `60s`
Optional > Requires	**fencing**

Optional > On Fail	**restart**

The start operation does the following:

- Mounts the filesystem
- Fails if the mount is unsuccessful

Stop Operation for a DMF-Managed User Filesystem

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**fence**

The stop operation does the following:

- Unmounts the filesystem
- Fails if the unmount is unsuccessful

Creating a DMF Administrative Filesystem Primitive

Required Fields for a DMF Administrative Filesystem

ID	*Unique ID such as* `dmf_spool_fs`
Class	**ocf**
Provider	**heartbeat**
Type	**Filesystem**

Instance Attributes for a DMF Administrative Filesystem

device	*The* `/dev/lxvm` *volume name of the DMF administrative filesystem device, such as* `/dev/lxvm/dmf_spool`
directory	*Mount point for the DMF administrative filesystem, such as* `/dmf/dmf_spool`
options	*Mount options*

Note: You must use the **options** attribute for the following DMF administrative filesystems:

- *MOVE_FS* requires `dmi` and `mtpt` (the value of the `mtpt` mount option must match the value used for the **directory** attribute, such as `/dmf/dmf_spool`)
- *STORE_DIRECTORY* for a DCM MSP requires `dirsync`, `dmi`, and `mtpt`
- *STORE_DIRECTORY* for a disk MSP requires `dirsync`

---

fstype	**xfs**

Meta Attributes for a DMF Administrative Filesystem

resource-stickiness	**1**
migration_threshold	**1**

Probe Monitor Operation for a DMF Administrative Filesystem

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Monitor Operation for a DMF Administrative Filesystem

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `120s`
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

The monitor operation does the following:

- Checks `/proc/mounts`, `/etc/mtab`, or the output of the `mount` command for the existence of the filesystem
- Fails if the filesystems is not mounted

Start Operation for a DMF Administrative Filesystem

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `60s`
Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Mounts the filesystem
- Fails if the mount is unsuccessful

Stop Operation for a DMF Administrative Filesystem

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**fence**

The stop operation does the following:

- Unmounts the filesystem
- Fails if the unmount is unsuccessful

Creating a Dedicated OpenVault Server Filesystem Primitive *(Optional)*

If you choose to have a dedicated filesystem for the OpenVault `serverdir` directory, use the information in the following sections.

Required Fields for an OpenVault Server Filesystem

ID	*Unique ID such as* `openvaultfs`
Class	**ocf**
Provider	**heartbeat**
Type	**Filesystem**

Instance Attributes for an OpenVault Server Filesystem

device	*The* `/dev/lxvm` *volume name of the DMF filesystem device, such as* `/dev/lxvm/openvault`
directory	*Mount point for the DMF filesystem, such as* `/dmf/openvault`
fstype	**xfs**

Meta Attributes for an OpenVault Server Filesystem

resource-stickiness	**1**

migration_threshold	**1**

Probe Monitor Operation for an OpenVault Server Filesystem

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Monitor Operation for an OpenVault Server Filesystem

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `120s`
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

The monitor operation does the following:

- Checks `/proc/mounts`, `/etc/mtab`, or the output of the `mount` command for the existence of the filesystem
- Fails if the filesystems is not mounted

Start Operation for OpenVault Server Filesystem

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `60s`
Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Mounts the filesystem
- Fails if the mount is unsuccessful

Stop Operation for an OpenVault Server Filesystem

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**fence**

The stop operation does the following:

- Unmounts the filesystem
- Fails if the unmount is unsuccessful

Testing Filesystem Resources

To test the filesystem resources for a DMF resource group named `dmfGroup`, do the following:

1. Ensure that all of the mount points required to mount all `Filesystem` resources exist on both nodes.

   Note: After a `Filesystem` primitive has been added to a resource group's configuration, moving that resource group will unmount the filesystem defined in the primitive. This will result in killing any process that has that filesystem in the path of its current working directory.

2. Verify that the filesystems are online on `node1`:

   ```
   node1# df -hl
   ```

3. Move the resource group containing all of the `Filesystem` resources from `node1` to `node2`:

   ```
   node1# crm resource move dmfGroup node2
   ```

4. Verify that the filesystems are correctly mounted on `node2` only:

   - On `node2`, check the mount table and verify that the filesystems are mounted and have the correct mount options. Use the `ls` and `df -lh` commands on the mount point to verify that the filesystem is functional.

- On `node1`, check the mount table and verify that none of the filesystems are mounted.

5. Move the resource group containing all of the `Filesystem` resources back to `node1`:

   ```
   node1# crm resource move dmfGroup node1
   ```

6. Verify that the filesystems are correctly mounted on `node1` only:

   - On `node1`, check the mount table and verify that the filesystems are mounted and have the correct mount options. Use the `ls` and `df -lh` commands on the mount point to verify that the filesystem is functional.
   - On `node2`, check the mount table and verify that none of the filesystems are mounted.

7. Remove the implicit location constraints imposed by the administrative `move` command above:

   ```
   node1# crm resource unmove dmfGroup
   ```

Virtual IP Address Resource

This section discusses examples of the following:

- "Creating the Virtual IP Address Primitive" on page 96
- "Testing the Virtual IP Address Resource" on page 99

Creating the Virtual IP Address Primitive

Required Fields for a Virtual IP Address

ID	*Unique ID such as* `VirtualIP`
Class	**ocf**
Provider	**heartbeat**

Type	**IPaddr2**

Instance Attributes for a Virtual IP Address

ip	*IP address of the virtual channel, such as* `128.162.244.240`
nic	*Network interface card that will service the virtual IP address, such as* `eth0`
cidr_netmask	*Short-notation network mask, which should match the subnet size at the site, such as* `24`
broadcast	*Broadcast IP address, such as* `128.162.244.255`

Meta Attributes for a Virtual IP Address

resource-stickiness	**1**
migration_threshold	**1**

Probe Monitor Operation for a Virtual IP Address

Note: Defining a monitor operation (other than a probe operation) on an `IPaddr2` resource is normally unnecessary.

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Start Operation for a Virtual IP Address

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `90s`
Optional > Requires	**fencing**

Optional > On Fail	**restart**

The start operation does the following:

- Establishes the IP alias on the specified NIC
- Fails if the IP alias is not established

Stop Operation for a Virtual IP Address

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `100s`
Optional > On Fail	**fence**

The stop operation does the following:

- Removes the IP alias from the specified NIC
- Fails if the IP alias is not removed

Testing the Virtual IP Address Resource

To test the `IPaddr2` resource, do the following:

1. Verify that the IP address (the value used for **ip** in "Instance Attributes for a Virtual IP Address" on page 97 above) is configured correctly on `node1`. For example, for the **ip** value `128.162.244.240`:

```
node1# ip -o addr show | grep ’128.162.244.240/’
4: eth2    inet 128.162.244.240/24 brd 128.162.244.255 scope global secondary eth2
```

2. Verify that `node2` does not accept the IP address packets by running the following command on `node2` (there should be no output):

```
node2# ip -o addr show | grep ’128.162.244.240/’
node2#
```

3. Connect to the virtual address using `ssh` or `telnet` and verify that the IP address is being served by the correct system. For example, for the IP address `128.162.244.240` and the machine named `node1`:

```
ha# ssh root@128.162.244.240
Last login: Mon Jul 14 10:34:58 2008 from mynode.mycompany.com
ha# uname -n
node1
```

4. Move the resource group containing the `IPaddr2` resource from `node1` to `node2`:

```
node1# crm resource move dmfGroup node2
```

5. Verify that the IP address is configured correctly on `node2`:

```
node2# ip -o addr show | grep ’128.162.244.240/’
4: eth2    inet 128.162.244.240/24 brd 128.162.244.255 scope global secondary eth2
```

6. Verify that `node1` does not accept the IP address packets by running the following command on `node1` (the output should be no output):

```
node1# ip -o addr show | grep  ’128.162.244.240/’
node1#
```

7. Connect to the virtual address using `ssh` or `telnet` and verify that the IP address is being served by the correct system. For example, for the IP address `128.162.244.240` and the machine named `node2`:

   ```
   ha# ssh root@128.162.244.240
   Last login: Mon Jul 14 10:34:58 2008 from mynode.mycompany.com
   ha# uname -n
   node2
   ```

8. Move the resource group containing the `IPaddr2` resource back to `node1`:

   ```
   node1# crm resource move dmfGroup node1
   ```

9. Test again as in steps 1-3 above.

10. Remove the implicit location constraints imposed by the administrative `move` command above:

   ```
   node1# crm resource unmove dmfGroup
   ```

TMF Resource

This section discusses examples of the following:

- "Configuring TMF for HA" on page 100
- "Creating the TMF Primitive" on page 101
- "Testing the TMF Resource" on page 105

Configuring TMF for HA

To configure TMF for HA and create the TMF resource, do the following:

1. Modify the `/etc/tmf/tmf.config` file so that all tape devices belonging to device groups that are managed by HA are configured `DOWN` in the `status` parameter in the `DEVICE` definition.

2. Copy the following file from `node1` to `node2`:

   ```
   /etc/tmf/tmf.config
   ```

   On `node2`, if the tape drive pathname (the `FILE` parameter in the `DEVICE` definition) for a given drive is not the same as the pathname for the same drive on `node1`, modify the pathname in the `/etc/tmf.config` file on `node2` so that it points to the appropriate pathname.

3. On `node2`, enable the `tmf` service to be started automatically at boot time:

   ```
   node2# chkconfig tmf on
   ```

4. Create the TMF resource primitive with the fields shown in "Creating the TMF Primitive" on page 101.

Creating the TMF Primitive

Required Fields for TMF

ID	*Unique ID such as* `tmf`
Class	**ocf**
Provider	**sgi**
Type	**tmf**

Instance Attributes for TMF

devgrpnames	*Comma-separated list of TMF device groups defined in the* `tmf.config` *file that are to be managed by HA, such as:* `ibm3592,t10ka`
mindevsup	*Comma-separated list of the number of devices, one entry per device group, that must be configured up successfully within the corresponding device group in order to count the group as being highly available, such as:* `1,0`

Note: A value of `0` indicates that failover will never be initiated, even if all the devices in that device group are unavailable. This value is supported for all device groups; however, in order for TMF to be considered up, at least one tape device in some device group must be up. If there are no devices up in all defined device groups, then the resource agent will be considered to be in a stopped state, which will impact the resource monitor and the resource start actions.

devtimeout

Comma-separated list of timeouts in seconds, one entry per device group, that are used to decide how long to wait for a device in that device group to finish configuring up or down, such as:

```
120,240
```

Note: Changing the up / down state of a device may require rewinding and unloading a tape left in the drive by a previous host. Different tape device types have different maximum rewind and unload times, which can be obtained from the vendor's product literature. To calculate the timeout value for a particular device group, add the maximum rewind time for a device in that group to the device's unload time plus add an additional 10 seconds to allow for any required robot hand movement.

For example, 3592 tape drives with a maximum rewind time of 78 seconds and an unload time of 21 seconds require a **devtimeout** value of 78+21+10=109 seconds. 9940B tape drives with a maximum rewind time of 90 seconds and an unload time of 18 seconds require a **devtimeout** of 90+18+10=118.

admin_emails

(Optional) Comma-separated list of administrator email addresses corresponding to the device groups listed in **devgrpnames***, such as:*

```
root,admin1
```

Note: You can use the same email address for more than one device group (such as `admin1,admin1`). The email address will be used to send a message whenever tape drives that were previously available become unavailable, so that the administrator can take action to repair the drives in a timely fashion.

loader_names	*Comma-separated list of loader names configured in* `tmf.config` *that correspond to the device groups listed in* **devgrpnames**, *such as:* `ibm3494,l700a`
loader_hosts	*Comma-separated list of hosts through which the corresponding loaders listed in* **loader_names** *are controlled, such as:* `ibm3494cps,stkacsls`
loader_users	*Comma-separated list of user names that are used to log in to the corresponding hosts listed in* **loader_hosts**, *such as:* `root,acssa`
loader_passwords	*Comma-separated list of passwords corresponding to the user names listed in* **loader_users**, *such as:* `passwd1,passwd2`

Meta Attributes for TMF

resource-stickiness	**1**
migration_threshold	**1**

Probe Monitor Operation for TMF

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**

Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Monitor Operation for TMF

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `120s`
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

The monitor operation does the following:

- Issues a `tmstat` command and parses the output
- Fails if insufficient drives came up in any device group or if TMF is down

Start Operation for TMF

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `236s`

Note: The `tmf` resource agent will try twice to configure each drive up before considering it unusable, so the **start timeout** value should therefore be at least twice the greatest **devtimeout** value. For example, 2*118=236. You should usually use the same **timeout** value for **start** and **stop**.

Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Starts the TMF daemon if necessary
- Configures up the tape loader and all tape drives in each device group
- Preempts reservations

- Forces dismount if necessary
- Fails if insufficient drives come up in any device group

Stop Operation for TMF

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `236s`
Optional > On Fail	**fence**

The stop operation does the following:

- Configures down all tape drives in each device group
- Forces a release of drives allocated to a user job
- Fails if any drive in any device group could not be stopped

Testing the TMF Resource

To test the TMF resource as part of a resource group named `dmfGroup`, do the following:

1. Use `tmmls` to show the loader status.
2. Use `tmstat` to show the drive status. Verify that all of the tape drives in all HA-managed device groups are in `assn` or `idle` status on `node1`.
3. Move the resource group containing the `tmf` resource to `node2`:

   ```
   node1# crm resource move dmfGroup node2
   ```

4. Verify that the state is correct:
   - Use `tmstat` to verify that the tape drives all have a status of `down` or `sdwn` on `node1` and that they have a status of `idle` or `assn` on `node2`
   - Use `tmmls` to verify that all of the loaders on `node1` still have a status of `UP`
5. Verify that the timeout values for the start, stop, and monitor operations are appropriate. Do the following:

a. On `node2`, look in `/var/log/messages` for the time when the resource start operation started and ended. Also capture the start and end times of the monitor operation.

b. On `node1`, look in `/var/log/messages` to find the start and stop times for the stop operation.

c. Subtract the ending time from the starting time in each case to get the required time for each operation.

d. Take the above values and increase them by 10%.

Following are examples of finding the start, stop, and monitor operation durations (line breaks shown here for readability):

```
node1 -> egrep "do_lrm_rsc_op.*Performing.*tmf_start|process_lrm_event.*tmf_start" /var/log/messages
May 11 08:20:53 node1 crmd: [6498]: info: do_lrm_rsc_op: Performing
  key=47:81:0:562726de-a397-4c6c-8501-b273c214eb3f op=tmf_start_0 )
May 11 08:21:10 node1 crmd: [6498]: info: process_lrm_event: LRM operation tmf_start_0 (call=90, rc=0,
  cib-update=88, confirmed=true) ok

node1 -> egrep "do_lrm_rsc_op.*Performing.*tmf_stop|process_lrm_event.*tmf_stop" /var/log/messages
May 11 08:27:39 node1 crmd: [6498]: info: do_lrm_rsc_op: Performing
  key=46:82:0:562726de-a397-4c6c-8501-b273c214eb3f op=tmf_stop_0 )
May 11 08:27:40 node1 crmd: [6498]: info: process_lrm_event: LRM operation tmf_stop_0 (call=92, rc=0,
  cib-update=100, confirmed=true) ok

node1 -> egrep "do_lrm_rsc_op.*Performing.*tmf_monitor|process_lrm_event.*tmf_monitor" /var/log/messages
May 11 08:08:21 node1 crmd: [6498]: info: do_lrm_rsc_op: Performing
  key=16:78:7:562726de-a397-4c6c-8501-b273c214eb3f op=tmf_monitor_0 )
May 11 08:08:21 node1 crmd: [6498]: info: process_lrm_event: LRM operation tmf_monitor_0 (call=69, rc=7,
  cib-update=77, confirmed=true) not running
May 11 08:21:10 node1 crmd: [6498]: info: do_lrm_rsc_op: Performing
  key=48:81:0:562726de-a397-4c6c-8501-b273c214eb3f op=tmf_monitor_30000 )
May 11 08:21:11 node1 crmd: [6498]: info: process_lrm_event: LRM operation tmf_monitor_30000 (call=91,
  rc=0, cib-update=89, confirmed=false) ok
May 11 08:27:39 node1 crmd: [6498]: info: process_lrm_event: LRM operation tmf_monitor_30000 (call=91,
  status=1, cib-update=0, confirmed=true) Cancelled
```

6. If you need to change any values, modify the primitive using the HA GUI (`crm_gui`).

7. Move the resource group containing the `tmf` resource back to `node1`:

   ```
   node1# crm resource move dmfGroup node1
   ```

8. Verify that the state is correct:

   - Use `tmstat` to verify that the tape drives all have a status of `down` or `sdwn` on `node2` and that they have a status of `idle` or `assn` on `node1`
   - Use `tmmls` to verify that all of the loaders on `node2` still have a status of `UP`

9. Remove the implicit location constraints imposed by the administrative `move` command above:

   ```
   node1# crm resource unmove dmfGroup
   ```

OpenVault Resource

This section discusses examples of the following:

- "Configuring OpenVault for HA" on page 107
- "Creating the OpenVault Primitive" on page 114
- "Testing the OpenVault Resource" on page 117

Configuring OpenVault for HA

Do the following to configure OpenVault for HA:

1. Ensure that all of the resources within the resource group are moved back to `node1` (if not already there).

2. Add the primitive using the values shown in "Creating the OpenVault Primitive" on page 114.

3. Run `ov_admin` on `node1`:

   ```
   node1# ov_admin
   ...
   ```

   When asked for the server hostname, specify the virtual hostname (the `virtualhost` value). `ov_admin` will automatically convert the OpenVault configuration to an HA configuration by doing the following:

   a. Stopping the server (if it is running).

   b. Creating the directory specified by `serverdir`.

   c. Moving the OpenVault database and logs into the directory specified by `serverdir`.

   d. Making the host specified by `virtualhost` be the same hostname address used by the OpenVault server and all drive control programs (DCPs) and library control programs (LCPs) on `node1`.

4. Verify that the DCPs and LCPs are running on `node1` by using the `ov_stat`(8) command with the `-ld` options, which should show `ready` in the `LCP State` and `DCP State` fields (output condensed here for readability):

```
node1# ov_stat -ld
Library Name   Broken ...  LCP State
lib1           false  ...  ready

Drive Name     Group  ...          DCP State   Occupied    Cartridge PCL
tape1          drives ...          ready       false
tape2          drives ...          ready       false
```

5. Enable the passive server (`node2`) as a potential OpenVault server:

   a. On `node1`:

      To allow `node2` to access the OpenVault server, run `ov_admin` and answer `yes` when prompted to start the server. Then select the following menus, answering the questions when prompted:

      ```
      node1# ov_admin
      ...
      22 - Manage OpenVault Client Machines
              1 - Activate an OpenVault Client Machine
      ```

      When asked if DCPs will also be configured, answer yes.

b. On `node2`:

   i. Use `ov_admin` to enable the node to issue administrative commands by entering the virtual hostname:

```
node2# **ov_admin**
...
Name where the OpenVault server is listening? [*virtualhostname*]
```

6. If your site contains a physical or virtual tape library, define DCPs and LCPs on the passive node (`node2`). Whenever `ov_admin` asks for the server hostname, use the virtual hostname.

   Note: If your site contains COPAN MAID shelves, you will create their OpenVault components later in "Creating the OpenVault Components on the Failover Node" on page 119. Therefore, you can skip this step if your site contains only COPAN MAID shelves (and no physical tape library or COPAN VTL).

   a. Configure drives by selecting:

```
2 - Manage DCPs for locally attached Drives
...
1 - Create a new SCSI DCP
```

   You must specify the drive for which would you like to add a DCP and the DCP name.

   On `node2`, you must configure at least one DCP for each drive that is already configured on `node1`.

   b. Configure libraries by selecting:

```
1 - Manage LCPs for locally attached Libraries
```

   On `node2`, you must configure at least one LCP for each library that is already configured on `node1`:

   • When asked for the name of the device, use the same library name that was used on `node1`. The LCP instance name will automatically reflect the `node2` name (for example, for the `l700a` library, the LCP instance name on `node1` is `l700a@node1` and the LCP instance name on `node2` will be `l700a@node2`).

- When prompted with `Library ’libname’ already exists in OpenVault catalog; create LCP anyway?`, respond `yes`.

  All DCPs and LCPs have now been configured and started on `node2`, but the server has not yet been configured to allow the LCPs to connect. This will be accomplished in step c.

c. On `node1`, use `ov_admin` to enable remote LCPs on `node2` by selecting:

```
node1# ov_admin
...
21 - Manage remote Libraries and LCPs
```

   You must enable each remote LCP using the same library and LCP names that you used on `node2`:

```
4 - Activate another LCP for an existing Library
```

   Now that server configuration is complete, the LCPs on `node2` will shortly discover that they are able to connect to the server.

d. On `node2`:

   i. Verify that the DCPs are running successfully. For example, the following output shows under `DCPHost` and `DCPStateSoft` columns that the DCP is running and connected to the OpenVault server (`ready`) on the active

HA node (node1) and running in disconnected mode (disconnected) on the failover node (node2):

```
node2# ov_dumptable -c DriveName,DCPName,DCPHost,DCPStateSoft DCP
DriveName  DCPName          DCPHost DCPStateSoft
9940B_25a1 9940B_25a1@node1 node1    ready
9940B_b7ba 9940B_b7ba@node1 node1    ready
9940B_93c8 9940B_93c8@node1 node1    ready
LTO2_682f  LTO2_682f@node1  node1    ready
LTO2_6832  LTO2_6832@node1  node1    ready
LTO2_6835  LTO2_6835@node1  node1    ready
LTO2_6838  LTO2_6838@node1  node1    ready
9940B_25a1 9940B_25a1@node2 node2    disconnected
9940B_93c8 9940B_93c8@node2 node2    disconnected
9940B_b7ba 9940B_b7ba@node2 node2    disconnected
LTO2_682f  LTO2_682f@node2  node2    disconnected
LTO2_6832  LTO2_6832@node2  node2    disconnected
LTO2_6838  LTO2_6838@node2  node2    disconnected
LTO2_6835  LTO2_6835@node2  node2    disconnected
```

Note: All of the alternate DCPs should transition to disconnected state, meaning that they have successfully contacted the server. Do not proceed until they all transition to disconnected. A state of inactive means that the DCP has not contacted the server, so if the state remains inactive for more than a few minutes, the DCP may be having problems connecting to the server.

ii. Verify that the LCPs are running. For example, the following output shows under LCPHost and LCPStateSoft columns that the LCP is running and connected to the OpenVault server (ready) on the active HA node (node1) and running in disconnected mode (disconnected) on the failover node (node2):

```
node2# ov_dumptable -c LibraryName,LCPName,LCPHost,LCPStateSoft LCP
LibraryName LCPName      LCPHost LCPStateSoft
SL500-2     SL500-2@node1 node1    ready
L700A       L700A@node1   node1    ready
SL500-2     SL500-2@node2 node2    disconnected
L700A       L700A@node2   node2    disconnected
```

Note: It may take a minute or two for the LCPs to notice that they are able to connect to the server and activate themselves. All of the alternate LCPs should transition to `disconnected` state, meaning that they have successfully contacted the server. Do not proceed until they all transition to `disconnected`. A state of `inactive` means that the LCP has not contacted the server, so if the state remains `inactive` for more than a couple of minutes, the LCP may be having problems connecting to the server.

---

7. Stop Openvault on `node2`:

   a. Stop all DCPs and LCPs on `node2`:

   ```
   node2# ov_stop
   ```

   b. Disable the `openvault` service from being started automatically at boot time:

   ```
   node2# chkconfig openvault off
   ```

8. Run `ov_admin` on each parallel data mover node:

   a. Enter the OpenVault virtual hostname, the port number, and security key as needed:

```
pdmn# ov_admin
...
Name where the OpenVault server is listening?  [servername] virtualhostname
What port number is the OpenVault server on virtualhostname using? [44444]
What security key would you like the admin commands to use? [none]
```

   b. Update the server name for each DCP using item `6` in the `OpenVault DCP Configuration` menu:

   ```
   2 - Manage DCPs for locally attached Drives
     6 - Change Server Used by DCPs
       a - Change server for all DCPs.
   ```

   c. Restart the DCPs to connect to the OpenVault server using the virtual server name:

   ```
   pdmn# service openvault stop
   pdmn# service openvault start
   ```

d. Update the server name for each LCP using item `8` in the `OpenVault LCP Configuration` menu:

```
1 - Manage LCPs for locally attached Libraries
  8 - Change Server Used by LCPs
    a - Change server for all LCPs.
```

e. Restart the LCPs to connect to the OpenVault server using the virtual server name:

```
pdmn# service openvault stop
pdmn# service openvault start
```

This step may generate errors for COPAN MAID shelf DCPs and LCPs whose default host is not on this host. You can ignore errors such as the following:

```
shelf C02 is owned by owner_nodename
```

9. On `node1`, stop the OpenVault server and any DCPs and LCPs:

```
node1# ov_stop
```

10. On `node1`, disable the `openvault` service from being started automatically at boot time:

```
node1# chkconfig openvault off
```

11. Update the `openvault` resource so that it is managed by HA:

```
node1# crm resource manage OpenVault_resourcePRIMITIVE
```

12. *(Optional)* If you want to have additional OpenVault clients that are not DMF servers, such as for running administrative commands, install the OpenVault software on those clients and run `ov_admin` as shown below. When asked for the server hostname, specify the virtual hostname. This connects the clients to the virtual cluster, rather than a fixed host, so that upon migration they follow the server.

Note: You may wish to set the environment variable `OVSERVER` to the virtual hostname so that you can use the OpenVault administrative commands without having to specify the `-S` parameter on each command.

Do the following for each OpenVault client:

a. On `node1`:

   To allow `node2` to act as an administrative client, run `ov_admin` and select the following menus, answering the questions when prompted:

   ```
   node1# ov_admin
   ...
   22 - Manage OpenVault Client Machines
           1 - Activate an OpenVault Client Machine
   ```

b. On the OpenVault client node, use `ov_admin` to enable the node to issue administrative commands by entering the virtual hostname, the port number, and security key as needed:

```
node2# ov_admin
...
Name where the OpenVault server is listening? [virtualhostname]
What port number is the OpenVault server on virtualhostname using? [44444]
What security key is used for admin commands on the HA OpenVault servers? [none]
```

Creating the OpenVault Primitive

Required Fields for OpenVault

ID	*Unique ID such as* `Openvault`
Class	**ocf**
Provider	**sgi**
Type	**openvault**

Instance Attributes for OpenVault

virtualhost	*Hostname that will resolve as defined in* `/etc/nsswitch.conf` *to the IP address configured in* "Virtual IP Address Resource" on page 96*, such as* `128.162.244.240`
serverdir	*Directory containing the OpenVault server configuration, such as* `/dmf/home/openvault`

Note: serverdir must be a path on a mountable CXFS filesystem that is being managed by a `cxfs` resource or on an XFS filesystem that is being managed by a `Filesystem` resource in the same resource group as the `openvault` resource. The filesystem could be one dedicated for OpenVault use (such as `/dmf/openvault`) or it could be an HA-managed filesystem in the same resource group that has sufficient space (such as `/dmf/home`). As part of the conversion to HA, OpenVault will create this directory and move its database and logs into the directory; OpenVault will fail if the directory already exists.

Meta Attributes for OpenVault

resource-stickiness	**1**
migration_threshold	**1**
is-managed	**false**

Note: The **false** setting facilitates the conversion of OpenVault from a single-system server to HA.

Probe Monitor Operation for OpenVault

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Monitor Operation for OpenVault

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `120s`

Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

The monitor operation does the following:

- Verifies that the `ovroot` process appears in the `ov_procs` output
- Fails if the `ovroot` process is not running

Start Operation for OpenVault

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `300s`
Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Verifies that the OpenVault **serverdir** directory is mounted and that the **virtualhost** IP address is available
- Starts OpenVault with the following command:

  ```
  ov_start server clients
  ```

- Fails if either the **serverdir** value or the **virtualhost** value is unavailable, or if OpenVault does not start

Stop Operation for OpenVault

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `90s`
Optional > On Fail	**fence**

The stop operation does the following:

- Stops OpenVault with the following command:

  ```
  ov_stop server clients
  ```

- Kills any remaining OpenVault processes found by `ov_procs`
- Clears the OpenVault semaphore with the following command:

  ```
  ipcrm -s
  ```

- Fails if OpenVault could not be stopped or if the semaphore could not be cleared

Testing the OpenVault Resource

To test the OpenVault resource as part of a resource group named `dmfGroup`, do the following:

1. Verify that all of the OpenVault libraries and drives become available after a few minutes on `node1`:

```
node1# ov_stat -ld
Library Name   Broken   Disabled   State     LCP State
L700A          false    false      ready     ready
SL500-2        false    false      ready     ready

Drive Name     Group      Access Broken Disabled SoftState HardState DCP State   Occupied  Cartridge PCL
9940B_25a1     9940B_drives true false  false    ready     unloaded  ready       false
9940B_93c8     9940B_drives true false  false    ready     unloaded  ready       false
9940B_b7ba     9940B_drives true false  false    ready     unloaded  ready       false
LTO2_682f      LTO2_drives true  false  false    ready     unloaded  ready       false
LTO2_6832      LTO2_drives true  false  false    ready     unloaded  ready       false
LTO2_6835      LTO2_drives true  false  false    ready     unloaded  ready       false
LTO2_6838      LTO2_drives true  false  false    ready     unloaded  ready       false
```

2. Move the resource group containing the `openvault` resource from `node1` to `node2`:

   ```
   node1# crm resource move dmfGroup node2
   ```

3. Verify that all of the drives become available after a few moments. For example:

```
node2# ov_stat -ld
Library Name   Broken   Disabled   State     LCP State
L700A          false    false      ready     ready
SL500-2        false    false      ready     ready

Drive Name     Group      Access Broken Disabled SoftState HardState DCP State   Occupied  Cartridge PCL
9940B_25a1     9940B_drives true false  false    ready     unloaded  ready       false
9940B_93c8     9940B_drives true false  false    ready     unloaded  ready       false
9940B_b7ba     9940B_drives true false  false    ready     unloaded  ready       false
LTO2_682f      LTO2_drives true  false  false    ready     unloaded  ready       false
LTO2_6832      LTO2_drives true  false  false    ready     unloaded  ready       false
LTO2_6835      LTO2_drives true  false  false    ready     unloaded  ready       false
LTO2_6838      LTO2_drives true  false  false    ready     unloaded  ready       false
```

4. Move the resource group containing the `openvault` resource back to `node1`:

   ```
   node1# crm resource move dmfGroup node1
   ```

5. Verify that all of the drives become available after a few moments. For example:

```
node1# ov_stat -ld
Library Name   Broken   Disabled   State     LCP State
L700A          false    false      ready     ready
SL500-2        false    false      ready     ready

Drive Name     Group      Access Broken Disabled SoftState HardState DCP State   Occupied  Cartridge PCL
9940B_25a1     9940B_drives true false  false    ready     unloaded  ready       false
9940B_93c8     9940B_drives true false  false    ready     unloaded  ready       false
9940B_b7ba     9940B_drives true false  false    ready     unloaded  ready       false
LTO2_682f      LTO2_drives true  false  false    ready     unloaded  ready       false
LTO2_6832      LTO2_drives true  false  false    ready     unloaded  ready       false
LTO2_6835      LTO2_drives true  false  false    ready     unloaded  ready       false
LTO2_6838      LTO2_drives true  false  false    ready     unloaded  ready       false
```

6. Remove the implicit location constraints imposed by the administrative `move` command above:

   ```
   node1# crm resource unmove dmfGroup
   ```

COPAN MAID OpenVault Client Resource

This section discusses the following:

- "Creating the OpenVault Components on the Failover Node" on page 119
- "Creating the COPAN MAID OpenVault Client Primitive" on page 121
- "Testing the COPAN MAID OpenVault Client Resource" on page 123

Creating the OpenVault Components on the Failover Node

When you configured the standard services according to the information in *COPAN MAID for DMF Quick Start Guide*, you executed an `ov_shelf`(8) command for each shelf in order to create the required OpenVault components (see "COPAN MAID Standard Service" on page 49).

In this step, you will create corresponding OpenVault components for the failover node so that it is ready to resume control of OpenVault in case of failover, using the following information for shelf 0 as an example:

- Shelf identifier: `C00` (indicating cabinet 0, shelf 0)
- Active node: `node1`
- Failover node: `node2`

Note: For more information about the shelf identifier, see *COPAN MAID for DMF Quick Start Guide*.

Do the following:

1. On `node1`:

   a. Stop all of the shelf's OpenVault clients:

   ```
   node1# ov_stop C00*
   ```

   b. Export the OCF shelf, hostname, and root environment variables for use by the `copan_ov_client` script:

   ```
   node1# export OCF_RESKEY_shelf_name=C00
   node1# export OCF_RESKEY_give_host=node2
   node1# export OCF_ROOT=/usr/lib/ocf
   ```

c. Transfer ownership of the shelf from `node1` to `node2`:

```
node1# /usr/lib/ocf/resource.d/sgi/copan_ov_client give
```

2. On `node2`:

a. Verify that `node2` now owns the shelf's XVM volumes (`C00A` through `C00Z`, although not necessarily listed in alphabetical order):

```
node2# xvm -d local probe | grep C00
phys/copan_C00M
phys/copan_C00B
phys/copan_C00G
...
```

b. Create the OpenVault components for `node2`:

```
node2# ov_shelf create C00
```

This automatically starts all of the shelf's OpenVault components.

For more information, see *COPAN MAID for DMF Quick Start Guide*.

c. Stop all of the shelf's OpenVault clients:

```
node2# ov_stop C00*
```

d. Export the shelf, hostname, and OCF root environment variables for use by the `copan_ov_client` script:

```
node2# export OCF_RESKEY_shelf_name=C00
node2# export OCF_RESKEY_give_host=node1
node2# export OCF_ROOT=/usr/lib/ocf
```

e. Transfer ownership of the shelf from `node2` back to `node1`:

```
node2# /usr/lib/ocf/resource.d/sgi/copan_ov_client give
```

3. On `node1`:

a. Verify that `node1` once again owns the shelf's XVM volumes (`C00A` through `C00Z`, although not necessarily listed in alphabetical order):

```
node1# xvm -d local probe | grep C00
phys/copan_C00M
phys/copan_C00B
phys/copan_C00G
```

```
...
```

b. Restart all of the shelf's OpenVault clients:

```
node1# ov_start C00*
```

4. Repeat steps 1 through 3 for each shelf.

Creating the COPAN MAID OpenVault Client Primitive

Use the values shown in the following sections when adding a COPAN OpenVault client primitive. There should be one primitive instance for each shelf.

Required Fields for a COPAN MAID OpenVault Client

ID	*Unique ID for the COPAN OpenVault client on each shelf, corresponding to the shelf ID, such as* `copan_C00` for shelf 0, which has a shelf ID of `C00`
Class	**ocf**
Provider	**sgi**
Type	**copan_ov_client**

Meta Attributes for a COPAN MAID OpenVault Client

resource-stickiness	**1**
migration-threshold	**1**
target-role	**Started**

Instance Attributes for a COPAN OpenVault Client

shelf_name	*The three-character shelf ID, using the naming convention described in the COPAN MAID for DMF Quick Start Guide, such as* `C00` *for cabinet 0 shelf 0.*

Probe Monitor Operation for a COPAN MAID OpenVault Client

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**

Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Monitor Operation for a COPAN MAID OpenVault Client

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `120s`
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

The monitor operation does the following:

- Verifies that OpenVault is functional: the LCP and at least one DCP are running and at least one RAID set is accessible on the shelf
- Fails if the COPAN MAID shelf is not functional

Start Operation for a COPAN MAID OpenVault Client

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `120s`
Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Starts the COPAN MAID shelf client by ensuring that RAID sets are available and that the OpenVault LCP and at least one DCP are running
- Fails if any of the above conditions are not met

Stop Operation for a COPAN MAID OpenVault Client

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `120s`

Optional > On Fail   **fence**

The stop operation does the following:

- Stops the COPAN MAID OpenVault client resource
- Fails if the COPAN MAID OpenVault client resource fails to stop

Testing the COPAN MAID OpenVault Client Resource

To test the COPAN MAID OpenVault client resource, do the following, using shelf `C02` as an example:

1. Verify that shelf `C02` becomes available after a few minutes on `node1`:

```
node1# ov_stat -L C02 -D ’C02.*’
Library Name   Broken   Disabled   State     LCP State
C02            false    false      ready     ready

Drive Name     Group      Access Broken Disabled SoftState HardState DCP State    Occupied   Cartridge PCL
C02d00         dg_c02     true   false  false    ready     unloaded  ready        false
C02d01         dg_c02     true   false  false    ready     unloaded  ready        false
C02d02         dg_c02     true   false  false    ready     unloaded  ready        false
C02d03         dg_c02     true   false  false    ready     unloaded  ready        false
C02d04         dg_c02     true   false  false    ready     unloaded  ready        false
C02d05         dg_c02     true   false  false    ready     unloaded  ready        false
C02d06         dg_c02     true   false  false    ready     unloaded  ready        false
```

2. Move the resource group containing the `copan_ov_client` resource from `node1` to `node2`:

   ```
   node1# crm resource move dmfGroup node2
   ```

3. Verify that shelf `C02` becomes available after a few minutes on `node2`:

```
node2# ov_stat -L C02 -D ’C02.*’
Library Name         Broken   Disabled   State     LCP State
C02                  false    false      ready     ready

Drive Name     Group      Access Broken Disabled SoftState HardState DCP State    Occupied   Cartridge PCL
C02d00         dg_c02     true   false  false    ready     unloaded  ready        false
C02d01         dg_c02     true   false  false    ready     unloaded  ready        false
```

```
C02d02         dg_c02     true   false  false    ready     unloaded  ready        false
C02d03         dg_c02     true   false  false    ready     unloaded  ready        false
C02d04         dg_c02     true   false  false    ready     unloaded  ready        false
C02d05         dg_c02     true   false  false    ready     unloaded  ready        false
C02d06         dg_c02     true   false  false    ready     unloaded  ready        false
```

4. Move the resource group containing the `copan_ov_client` resource back to `node1`:

   ```
   node1# crm resource move dmfGroup node1
   ```

5. Verify that shelf `C02` becomes available after a few minutes on `node1`:

```
node1# ov_stat -L C02 -D 'C02.*'
Library Name         Broken   Disabled   State     LCP State
C02                  false    false      ready     ready

Drive Name     Group      Access Broken Disabled SoftState HardState DCP State    Occupied   Cartridge PCL
C02d00         dg_c02     true   false  false    ready     unloaded  ready        false
C02d01         dg_c02     true   false  false    ready     unloaded  ready        false
C02d02         dg_c02     true   false  false    ready     unloaded  ready        false
C02d03         dg_c02     true   false  false    ready     unloaded  ready        false
C02d04         dg_c02     true   false  false    ready     unloaded  ready        false
C02d05         dg_c02     true   false  false    ready     unloaded  ready        false
C02d06         dg_c02     true   false  false    ready     unloaded  ready        false
```

6. Remove the implicit location constraints imposed by the administrative `move` command above:

   ```
   node1# crm resource unmove dmfGroup
   ```

DMF Resource

This section discusses examples of the following:

- "Configuring DMF for HA" on page 125
- "Creating the DMF Primitive" on page 127
- "Testing the DMF Resource " on page 130

Configuring DMF for HA

Note: The following procedure requires that the DMF application instances in OpenVault are configured to use a wildcard (`"*"`) for the hostname and instance name. For more information, see the chapter about mounting service configuration tasks in the *DMF 6 Administrator Guide for SGI InfiniteStorage*.

To configure DMF for HA, do the following:

1. Make the filesystem backup inventory accessible from all DMF servers in the HA cluster.

   The backup of DMF-managed user filesystems and DMF administrative filesystems is always performed on the active DMF server based upon parameters in the DMF configuration file. The `xfsdump` command maintains an inventory of all backups performed within the directory `/var/lib/xfsdump`; in an HA environment, the active DMF server node can change over time. Therefore, in order for `xfsdump` to maintain a consistent inventory, it must be able to access the inventory for all past backups even if those backups were created on another node.

   SGI recommends that you make the inventory accessible to all DMF server nodes by relocating it into an HA-managed DMF administrative filesystem within the same resource group as DMF. For example, create a site-specific directory in DMF's *HOME_DIR*, such as `/dmf/home/site_specific`:

   - On `node1` (which currently contain the inventory), enter the following:

     ```
     node1# cd /var/lib
     node1# cp -r xfsdump /dmf/home/site_specific/xfsdump
     node1# mv xfsdump xfsdump.bak
     node1# ln -s /dmf/home/site_specific/xfsdump xfsdump
     ```

     Note: In a brand-new DMF installation, the `/var/lib/xfsdump` directory will not exist until after a backup has been performed.

   - On `node2`, enter the following:

     ```
     node2# cd /var/lib
     node2# mv xfsdump xfsdump.bak
     node2# ln -s /dmf/home/site_specific/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.

---

2. On `node1`, modify the DMF configuration file as follows:

   - Set the `MAX_MS_RESTARTS` parameter in the appropriate `drivegroup` objects to `0` so that DMF will not restart the mounting service.
   - Set the `DUMP_INVENTORY_COPY` parameter so that it uses a DMF HA administrative filesystem that is on a different disk from the live inventory created above in step 1. If the live inventory in `/dmf/home/site_specific/xfsdump` is lost, you can then recreate it from the inventory backup in `DUMP_INVENTORY_COPY`. For example, you could create the directory `/dmf/journal/site_specific/inventory_copy` for use in `DUMP_INVENTORY_COPY`.
   - If you are using OpenVault, set the `MSG_DELAY` parameter in the `drivegroup` objects to a value of slightly more than 2 minutes.
   - Set the `SERVER_NAME` parameter for the `base` object to the HA virtual hostname of the DMF server.

     Note: If you change this parameter, you must copy the DMF configuration file (`/etc/dmf/dmf.conf`) manually to each parallel data mover node and then restart the services related to DMF. Do not change this parameter while DMF is running.

   -
     - Set the `INTERFACE` parameter in the `node` object for each potential DMF server node to the same virtual hostname used for `SERVER_NAME` in the `base` object.
   - If using the DMF Parallel Data Mover Option, create `node` objects for each parallel data mover node in the HA cluster.

   For more information, see the `dmf.conf`(5) man page and the *DMF 6 Administrator Guide for SGI InfiniteStorage.*

3. Copy the DMF configuration file (`/etc/dmf/dmf.conf`) from `node1` to `node2` and to any parallel data mover nodes in the DMF configuration. You may wish to use a symbolic link on `node1` and on `node2` that points to a shared location in the *HOME_DIR* directory. For example:

   ```
   ha# ln -s /dmf/home/dmf.conf /etc/dmf/dmf.conf
   ```

   Note: You cannot use a symbolic link for parallel data mover nodes because DMF itself keeps the `dmf.conf` file synchronized with the server node.

4. If you are using OpenVault and you explicitly set hostnames when you defined the `ov_keys` file during initial OpenVault setup, edit the `ov_keys` file and replace the hostname in field 1 of the DMF lines with the OpenVault virtual hostname. For example, if the virtual hostname is `virtualhost`:

   ```
   virtualhost   dmf   *  CAPI none
   virtualhost   dmf   *  AAPI none
   ```

   Note: If you used a wildcard hostname (*) when you defined the `ov_keys` file during initial OpenVault setup, there is no need to edit this file.

5. On each potential DMF server node in the HA cluster, disable the `dmf`, `dmfman`, and `dmfsoap` services from being started automatically at boot time:

   ```
   dmfserver# chkconfig dmf off
   dmfserver# chkconfig dmfman off
   dmfserver# chkconfig dmfsoap off
   ```

6. Create the DMF resource with the fields shown in "Creating the DMF Primitive" on page 127.

Creating the DMF Primitive

Required Fields for DMF

ID	*Unique ID, such as* `dmf`
Class	**ocf**
Provider	**sgi**

Type	**dmf**

Instance Attributes for DMF

monitor_level	*Level that specifies whether to check for the existence of the dmfdaemon process only (*`0`*) or also run an additional check using dmdstat (*`1`*)*

Meta Attributes for DMF

resource-stickiness	**1**
migration_threshold	**1**

Probe Monitor Operation for DMF

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Monitor Operation for DMF

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `120s`
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

The monitor operation does the following:

- Verifies that `dmfdaemon` is running by calling the following:

  ```
  ps -C dmfdaemon
  ```

- If **monitor_level** is set to `1`, verifies that `dmfdaemon` is responding via a `dmdstat` command
- Fails if the `dmfdaemon` process does not exist or if it is not responsive

Start Operation for DMF

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `240s`

Note: In a CXFS environment, you must account for the time required for relocation of DMF-managed user filesystems. In this case, you may want to use a **Timeout** value of `600s`.

Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Starts DMF by calling the following:

  ```
  /etc/init.d/dmf start
  ```

- Waits for a successful DMF startup by calling `dmstat` in a loop until `dmfdaemon` responds successfully
- Fails if `dmfdaemon` does not respond to a `dmdstat` query before the resource times out

Stop Operation for DMF

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `240s`
Optional > On Fail	**fence**

The stop operation does the following:

- Stops DMF by calling the following:

  ```
  /etc/init.d/dmf stop
  ```

- Issues a `dmclrmount` command
- Fails if DMF could not be stopped

Testing the DMF Resource

To test the `dmf` resource as part of a resource group named `dmfGroup`, do the following:

1. Verify that DMF has started by using the `dmdstat -v` command and manual `dmput` and `dmget` commands on `node1`:

   ```
   node1# dmdstat -v
   node1# xfs_mkfile size test_file
   node1# dmput -r test_file
   node1# dmdidle
   (wait a bit to allow time for the volume to be written and unmounted)
   node1# dmget test_file
   node1# rm test_file
   ```

2. Move the resource group containing the `dmf` resource to `node2` (because the mounting service is in the same resource group, it must be colocated and thus should failover with DMF to the new node):

   ```
   node1# crm resource move dmfGroup node2
   ```

3. Verify that DMF has started on the new node by using the `dmdstat -v` command and manual `dmput` and `dmget` commands on `node2`:

   ```
   node2# dmdstat -v
   node2# xfs_mkfile size another_test_file
   node2# dmput -r another_test_file
   node2# dmdidle
   (wait a bit to allow time for the volume to be written and unmounted)
   node2# dmget another_test_file
   node2# rm another_test_file
   ```

4. Move the resource group containing the `dmf` resource back to `node1`:

   ```
   node1# crm resource move dmfGroup node1
   ```

5. Verify that DMF has started by using the `dmdstat -v` command and manual `dmput` and `dmget` commands on `node1`:

   ```
   node1# dmdstat -v
   node1# xfs_mkfile size test_file
   node1# dmput -r test_file
   node1# dmdidle
   (wait a bit to allow time for the volume to be written and unmounted)
   ```

```
node1# dmget test_file
node1# rm test_file
```

6. Remove the implicit location constraints imposed by the administrative `move` command above:

   ```
   node1# crm resource unmove dmfGroup
   ```

NFS Resource

This section discusses examples of the following:

- "Configuring NFS for HA" on page 131
- "Creating the NFS Primitive" on page 132
- "Testing the NFS Resource" on page 134

Configuring NFS for HA

To configure NFS for HA, do the following:

1. Copy the `/etc/exports` entries that you would like to make highly available from `node1` to the `/etc/exports` file on `node2`.

   Note: Be sure to include the `fsid=`*unique_number* export option in order to prevent stale file handles after failover.

2. On both nodes, disable the `nfsserver` service from being started automatically at boot time:

   - On `node1`:

     ```
     node1# chkconfig nfsserver off
     ```

   - On `node2`:

     ```
     node2# chkconfig nfsserver off
     ```

3. Add the NFS resource primitive. See "Creating the NFS Primitive" on page 132.

Creating the NFS Primitive

Required Fields for NFS

ID	*Unique ID, such as* `nfs`
Class	**ocf**
Provider	**heartbeat**
Type	**nfsserver**

Instance Attributes for NFS

nfs_init_script	*NFS initialization script, such as* `/etc/init.d/nfsserver`
nfs_notify_cmd	*NFS notification command, such as* `/usr/sbin/sm-notify`
nfs_shared_infodir	*Site-specific NFS shared-information directory, such as a* `/mnt/cxfsvol1/.nfs` *subdirectory in the exported filesystem*
nfs_ip	*The same IP address specified for the* **ip** *field for the virtual IP resource* (see "Instance Attributes for a Virtual IP Address" on page 97)

Meta Attributes for NFS

resource-stickiness	**1**
migration_threshold	**1**

Probe Monitor Operation for NFS

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Monitor Operation for NFS

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `120s`
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

The monitor operation does the following:

- Checks for the existence of the `rpc.mountd`, `rpc.statd`, and `nfsd` processes by calling the **nfs_init_script** status action, such as the following:

```
/etc/init.d/nfsserver status
```

- Fails if the processes do not exist

Start Operation for NFS

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `60s`
Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Starts the NFS server by calling the **nfs_init_script** start action, such as the following:

```
/etc/init.d/nfsserver start
```

- Notifies clients by calling the **nfs_notify_cmd** command
- Fails if the NFS server does not start

Stop Operation for NFS

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `60s`

Optional > On Fail **fence**

The stop operation does the following:

- Stops the NFS server by calling the **nfs_init_script** stop action, such as the following:

  ```
  /etc/init.d/nfsserver stop
  ```

- Fails if the NFS server does not stop

Testing the NFS Resource

To test the `nfsserver` resource, do the following:

1. Run the following command on `node1` to verify that the NFS filesystems are exported:

   ```
   node1# exportfs -v
   /work.mynode1   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=8001)
   /work.mynode2   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=8002)
   /work.mynode3   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=8003)
   /work.mynode4   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=8004)
   /mirrors        <world>(ro,wdelay,root_squash,no_subtree_check,fsid=8005)
   /               <world>(ro,wdelay,root_squash,no_subtree_check,fsid=8006)
   ```

2. Mount the filesystems on a node that will not be a member of the HA cluster (`otherhost`):

   ```
   otherhost# mount node1:/nfsexportedfilesystem /mnt/test
   ```

3. Read and write to the NFS-mounted filesystems:

   ```
   otherhost# echo "data for a test file" > /mnt/test/testFile1A
   otherhost# cat /mnt/test/testFile1A
   test data for a test file
   ```

4. Move the resource group containing the `nfsserver` resource from `node1` to `node2`:

   ```
   node1# crm resource move dmfGroup node2
   ```

5. Run the following command on `node2` to verify that the NFS filesystems are exported:

   ```
   node2# exportfs -v
   /work.mynode1   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=8001)
   /work.mynode2   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=8002)
   /work.mynode3   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=8003)
   /work.mynode4   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=8004)
   /mirrors        <world>(ro,wdelay,root_squash,no_subtree_check,fsid=8005)
   /               <world>(ro,wdelay,root_squash,no_subtree_check,fsid=8006)
   ```

6. Read and write to the NFS-mounted filesystems:

   ```
   otherhost# echo "test data for another test file" > /mnt/test/testFile1B
   otherhost# cat /mnt/test/testFile1B
   test data for another test file
   ```

7. Move the resource group containing the `nfsserver` resource back to `node1`:

   ```
   node1# crm resource move dmfGroup node1
   ```

8. Remove the implicit location constraints imposed by the administrative `move` command executed above:

   ```
   node1# crm resource unmove dmfGroup
   ```

Samba Resources

This section discusses examples of the following:

- "Configuring Samba for HA" on page 135
- "Creating the `smb` Primitive" on page 137
- "Creating the `nmb` Primitive" on page 138
- "Testing the Samba Resources" on page 140

Configuring Samba for HA

To configure Samba for HA, do the following:

1. Use symbolic links to make the `/etc/samba` directory and the files within it available on both `node1` and `node2`. For example:

a. Copy `/etc/samba` to a shared location. For example:

```
node1# cp -a /etc/samba /mnt/data/.ha/etc-samba
```

b. Remove the original `/etc/samba` directory on both nodes:

- On `node1`:

```
node1# rm -r /etc/samba
```

- On `node2`:

```
node2# rm -r /etc/samba
```

c. Make a symbolic link from the shared location to the original name on both nodes:

- On `node1`:

```
node1# ln -s /mnt/data/.ha/etc-samba /etc/samba
```

- On `node2`:

```
node2# ln -s /mnt/data/.ha/etc-samba /etc/samba
```

2. Make the `/var/lib/samba` directory and the files within it available on both nodes. For example:

a. Copy the `/var/lib/samba` directory and its contents to `/mnt/data/.ha/var-lib-samba` on one node. For example, if using `node1`:

```
node1# cp -r /var/lib/samba /mnt/data/.ha/var-lib-samba
```

b. Verify that the following directories exist on both nodes:

```
/mnt/data/.ha/var-lib-samba/usershares
/mnt/data/.ha/var-lib-samba/ncalrpc
```

c. Add the following lines to the `/etc/samba/smb.conf` file on both nodes:

```
lock directory = /mnt/data/.ha/var-lib-samba
state directory = /mnt/data/.ha/var-lib-samba
cache directory = /mnt/data/.ha/var-lib-samba
usershare path = /mnt/data/.ha/var-lib-samba/usershares
ncalrpc dir = /mnt/data/.ha/var-lib-samba/ncalrpc
```

3. Disable the `smb` and `nmb` services from being started automatically at boot time on both nodes:

   - On `node1`:

     ```
     node1# chkconfig smb off
     node1# chkconfig nmb off
     ```

   - On `node2`:

     ```
     node2# chkconfig smb off
     node2# chkconfig nmb off
     ```

4. Add the Samba resource primitives. See:

   - "Creating the `smb` Primitive" on page 137
   - "Creating the `nmb` Primitive" on page 138

Creating the smb Primitive

Note: The Samba resources do not have as many required fields or attributes as other resources.

Required Fields for smb

ID	*Unique ID, such as* `smb`
Class	**lsb**
Type	**smb**

Probe Monitor Operation for smb

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Start Operation for smb

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `60s`
Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Starts the `smbd` service by calling the following:

  ```
  /etc/init.d/smb start
  ```

- Fails if the `smbd` service does not start

Stop Operation for smb

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**fence**

The stop operation does the following:

- Stops the `smbd` service by calling the following:

  ```
  /etc/init.d/smb stop
  ```

- Fails if the `smbd` service does not stop

Creating the nmb Primitive

Required Fields for nmb

ID	*Unique ID, such as* `nmb`
Class	**lsb**

Type	**nmb**

Probe Monitor Operation for nmb

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Start Operation for nmb

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `60s`
Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Starts the `nmbd` service by calling the following:

  ```
  /etc/init.d/nmb start
  ```

- Fails if the `nmbd` service does not start

Stop Operation for nmb

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**fence**

The stop operation does the following:

- Stops the `nmbd` service by calling the following:

  ```
  /etc/init.d/nmb stop
  ```

- Fails if the nmbd service does not stop

Testing the Samba Resources

To test the Samba resources, do the following:

1. Ensure that the resource group containing the smb and nmb resources is on node1:

   ```
   node1# crm resource move dmfGroup node1
   ```

2. Use smbclient from a machine outside of the HA cluster to connect to the Samba server on node1 and copy a file. For example, to log in to node1 as admin (assuming that admin is a valid login name in the homes section of the smb.conf file) copy origfileA to remotefileA on the remote host:

   ```
   otherhost# smbclient //node1/admin
   smb:\> get origfileA remotefileA
   ```

   Note: Depending upon the setting of the security parameter in the smb.conf file, this may involve using a Samba account that already exists.

3. Move the resource group containing the smb and nmb resources from node1 to node2:

   ```
   node1# crm resource move dmfGroup node2
   ```

4. Use smbclient from a machine outside of the HA cluster to connect to the Samba server on node2 and copy a file. For example, to log in to node2 as admin (assuming that admin is a valid login name in the homes section of the smb.conf file) and copy origfileB to remotefileB on the remote host:

   ```
   otherhost# smbclient //node2/admin
   smb:\> get origfileB remotefileB
   ```

5. Move the resource group containing the smb and nmb resources back to node1:

   ```
   node1# crm resource move dmfGroup node1
   ```

6. Remove the implicit location constraints imposed by the administrative move command executed above:

   ```
   node1# crm resource unmove dmfGroup
   ```

DMF Manager Resource

This section discusses examples of the following:

- "Configuring DMF Manager for HA" on page 141
- "Creating the DMF Manager Primitive" on page 142
- "Testing the DMF Manager Resource" on page 144

Configuring DMF Manager for HA

To configure DMF Manager for HA, do the following:

1. On both nodes, disable the `dmfman` service from being started automatically at boot time:

   - On `node1`:

     ```
     node1# chkconfig dmfman off
     ```

   - On `node2`:

     ```
     node2# chkconfig dmfman off
     ```

2. Add a primitive for DMF Manager using the values shown in "Creating the DMF Manager Primitive" on page 142.

3. Run the `dmfman_setup_ha` script to create the required links and directories in a commonly accessible filesystem (such as the DMF *HOME_DIR*) that will allow DMF statistics archives to be accessible across the HA cluster (include the `-u` option only if you are upgrading from a previous release):

   - On `node1`:

     ```
     node1# /usr/lib/dmf/dmfman_setup_ha -d HOME_DIR [-u node2name]
     ```

   - On `node2`:

     ```
     node2# /usr/lib/dmf/dmfman_setup_ha -d HOME_DIR [-u node1name]
     ```

   For example, if the `HOME_DIR` parameter is set to `/dmf/home` in `/etc/dmf/dmf.conf` and you are upgrading from a previous release, you would enter the following:

- On `node1`:

  ```
  node1# /usr/lib/dmf/dmfman_setup_ha -d /dmf/home -u node2
  ```

- On `node2`:

  ```
  node2# /usr/lib/dmf/dmfman_setup_ha -d /dmf/home -u node1
  ```

Creating the DMF Manager Primitive

Required Fields for DMF Manager

ID	*Unique ID, such as* `dmfman`
Class	**ocf**
Provider	**sgi**
Type	**dmfman**

Meta Attributes for DMF Manager

resource-stickiness	**1**
migration_threshold	**1**

Note: You might want to set the **migration-threshold** higher than 1 for this resource so that it will simply restart in place.

Probe Monitor Operation for DMF Manager

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Monitor Operation for DMF Manager

Note: You should only define a monitor operation for the `dmfman` resource if you want a failure of the DMF Manager resource to cause a failover for the entire resource group.

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `120s`
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

The monitor operation does the following:

- Verifies the DMF Manager status by calling the following:

  ```
  /etc/init.d/dmfman status
  ```

- Fails if DMF Manager is not running

Start Operation for DMF Manager

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `60s`
Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Starts DMF Manger by calling the following:

  ```
  /etc/init.d/dmfman start
  ```

- Waits for DMF Manager to start successfully by calling the following in a loop:

  ```
  /etc/init.d/dmfman status
  ```

- Fails if DMF Manager does not start successfully before the resource times out

Stop Operation for DMF Manager

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**fence**

The stop operation does the following:

- Stops DMF Manger by calling the following:

  ```
  /etc/init.d/dmfman stop
  ```

- Verifies the DMF Manager status by calling the following:

  ```
  /etc/init.d/dmfman status
  ```

- Fails if DMF Manager does not stop successfully

Testing the DMF Manager Resource

To test the `dmfman` resource, do the following:

1. Point your browser at `https://`*virtualIPaddress*`:1179` and verify that you can log in and use DMF Manager, such as viewing the **Overview** panel. For more information about using DMF Manager, see *DMF 6 Administrator Guide for SGI InfiniteStorage*.

2. Move the resource group containing the `dmfman` resource from `node1` to `node2`:

   ```
   node1# crm resource move dmfGroup node2
   ```

3. Repeat step 1 to verify that DMF Manager is still available.

4. Move the resource group containing the `dmfman` resource back to `node1`:

   ```
   node1# crm resource move dmfGroup node1
   ```

5. Remove the implicit location constraints imposed by the administrative `move` command executed above:

   ```
   node1# crm resource unmove dmfGroup
   ```

DMF Client SOAP Service Resource

This section discusses examples of the following:

- "Configuring DMF Client SOAP Service for HA" on page 145
- "Creating the DMF Client SOAP Service Primitive" on page 146
- "Testing the DMF Client SOAP Service Resource" on page 148

Configuring DMF Client SOAP Service for HA

To configure DMF client SOAP service for HA, do the following:

1. On both nodes, disable the `dmfsoap` service from being started automatically at boot time:

   - On `node1`:

     ```
     node1# chkconfig dmfsoap off
     ```

   - On `node2`:

     ```
     node2# chkconfig dmfsoap off
     ```

2. Add a primitive for DMF client SOAP resource using the values shown in "Creating the DMF Client SOAP Service Primitive" on page 146.

Creating the DMF Client SOAP Service Primitive

Required Fields for DMF Client SOAP Service

ID	*Unique ID, such as* `dmfsoap`
Class	**ocf**
Provider	**sgi**
Type	**dmfsoap**

Meta Attributes for DMF Client SOAP Service

resource-stickiness	**1**
migration_threshold	**1**

Note: You might want to set the **migration-threshold** higher than 1 for this resource so that it will simply restart in place.

Probe Monitor Operation for DMF Client SOAP Service

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Monitor Operation for DMF Client SOAP Service

Note: You should only define a monitor operation for the `dmfsoap` resource if you want a failure of DMF client SOAP service to cause a failover for the entire resource group.

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `120s`

Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

The monitor operation does the following:

- Verifies the DMF client SOAP service status by calling the following:

  ```
  /etc/init.d/dmfsoap status
  ```

- Fails if DMF client SOAP service is not running

Start Operation for DMF Client SOAP Service

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `60s`
Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Starts the DMF client SOAP service by calling the following:

  ```
  /etc/init.d/dmfsoap start
  ```

- Waits for DMF client SOAP service to start successfully by calling the following in a loop:

  ```
  /etc/init.d/dmfsoap status
  ```

- Fails if DMF client SOAP service does not start successfully before the resource times out

Stop Operation for DMF Client SOAP Service

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**fence**

The stop operation does the following:

- Stops the DMF client SOAP service by calling the following:

  ```
  /etc/init.d/dmfsoap stop
  ```

- Verifies the DMF client SOAP service status by calling the following:

  ```
  /etc/init.d/dmfsoap status
  ```

- Fails if the DMF client SOAP service does not stop successfully

Testing the DMF Client SOAP Service Resource

To test the `dmfsoap` resource, do the following:

1. Point your browser at `https://`*virtualIPaddress*`:1180/server.php` and verify that you can access the GUI and view the WSDL for one of the DMF client functions. For more information, see *DMF 6 Administrator Guide for SGI InfiniteStorage*.
2. Move the resource group containing the `dmfsoap` resource from `node1` to `node2`:

   ```
   node1# crm resource move dmfGroup node2
   ```

3. Repeat step 1 to verify that DMF client SOAP service is still available.
4. Move the resource group containing the `dmfsoap` resource back to `node1`:

   ```
   node1# crm resource move dmfGroup node1
   ```

5. Remove the implicit location constraints imposed by the administrative `move` command above:

   ```
   node1# crm resource unmove dmfGroup
   ```

Chapter 8

COPAN MAID HA Service for Mover Nodes

This chapter contains the following sections:

Note: The attributes listed in this chapter and the various value recommendations are in support of this example. If you are using the resources in a different manner, you must evaluate whether these recommendations and the use of meta attributes apply to your intended site-specific purpose.

COPAN MAID HA for Mover Nodes Example Procedure

Note: You must determine appropriate monitoring interval and timeout values for your site. See "Avoid Unnecessary Failovers" on page 16.

Figure 8-1 on page 150 shows a map of an example configuration process for the COPAN OpenVault client service for COPAN MAID shelves in an active/active HA cluster that consists of two parallel data mover nodes named `pdmn1` and `pdmn2`. (`pdmn1` is the same node as `node1` referred to in Chapter 4, "Outline of the Configuration Procedure" on page 39.)The map refers to resource agent type names such as `cxfs-client` and `copan_ov_client`.

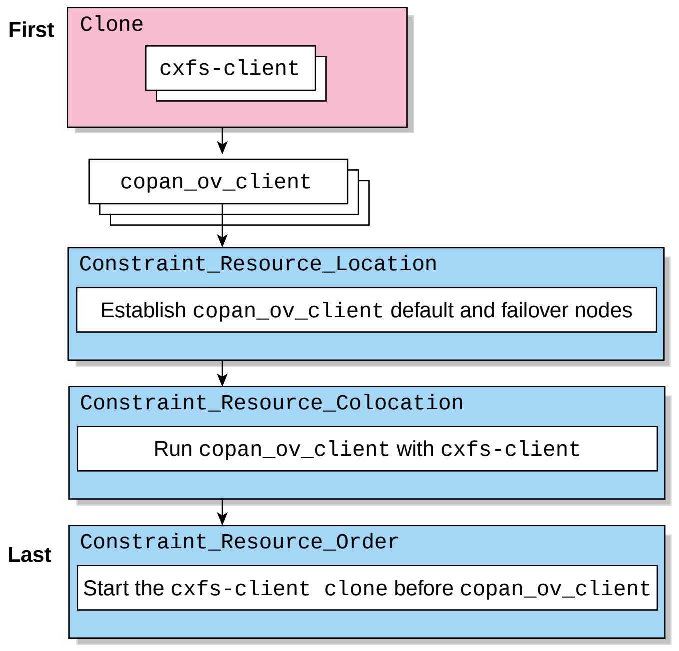
First
Clone
cxfs-client
copan_ov_client
Constraint_Resource_Location
Establish copan_ov_client default and failover nodes
Constraint_Resource_Colocation
Run copan_ov_client with cxfs-client
Last
Constraint_Resource_Order
Start the cxfs-client clone before copan_ov_client

Figure 8-1 COPAN MAID HA Service for Mover Nodes

To manage COPAN MAID shelves in an active/active HA environment, use the steps in the following sections:

- "Disable the Parallel Data Mover Nodes and the Services" on page 151
- "Create the OpenVault Components on the Failover Node" on page 152
- "Start the GUI" on page 154
- "Create the CXFS Client Clone" on page 154
- "Test the Clone" on page 155
- "Create the COPAN OpenVault Client Resources" on page 157
- "Create the Constraints" on page 157
- "Test the `ov_copan_client` Resource" on page 158

Disable the Parallel Data Mover Nodes and the Services

Do the following to ensure that there is no activity to the COPAN MAID shelf:

1. On the DMF server, disable both parallel data mover nodes:

   ```
   dmfserver# dmnode_admin -d pdmn1 pdmn2
   ```

2. Verify that there are no `dmatwc` or `dmatrc` data mover processes running on either parallel data mover node. For example, the output of the following command should be empty on both nodes:

   - On `pdmn1`:

     ```
     pdmn1# ps -ef | egrep 'dmatrc|dmatwc' | grep -v grep
     pdmn1#
     ```

   - On `pdmn2`:

     ```
     pdmn2# ps -ef | egrep 'dmatrc|dmatwc' | grep -v grep
     pdmn2#
     ```

   If the output is not empty, you must wait for the `dmnode_admin -d` action from step 1 to complete (the entire process can take 6 minutes or longer). Rerun the `ps` command until there is no output.

3. Determine which CXFS filesystems are mounted:

   ```
   # ls /dev/cxvm
   ```

   Save the output from this command for use later when you define the **volnames** instance attribute in "Instance Attributes for a CXFS Client" on page 159.

4. On both parallel data mover nodes, disable the `openvault` and `cxfs_client` services from being started automatically at boot time and stop the currently running services:

   - On `pdmn1`:

     ```
     pdmn1# chkconfig openvault off
     pdmn1# chkconfig cxfs_client off

     pdmn1# service openvault stop
     pdmn1# service cxfs_client stop
     ```

- On pdmn2:

  ```
  pdmn2# chkconfig openvault off
  pdmn2# chkconfig cxfs_client off

  pdmn2# service openvault stop
  pdmn2# service cxfs_client stop
  ```

Create the OpenVault Components on the Failover Node

When you configured the standard services according to the information in *COPAN MAID for DMF Quick Start Guide*, you executed an `ov_shelf`(8) command for each COPAN MAID shelf in order to create the required OpenVault components (see "COPAN MAID Standard Service" on page 49).

In this step, you will create corresponding OpenVault components for the failover node so that it is ready to assume control of OpenVault in case of failover, using the following information for shelf 0 as an example:

- Shelf identifier: `C00` (indicating cabinet 0, shelf 0)
- Default node: `pdmn1`
- Failover node: `pdmn2`

Note: For more information about the shelf identifier, see *COPAN MAID for DMF Quick Start Guide*.

Do the following:

1. On `pdmn1`:

   a. Export the shelf, hostname, and OCF root environment variables for use by the `copan_ov_client` script:

   ```
   pdmn1# export OCF_RESKEY_shelf_name=C00
   pdmn1# export OCF_RESKEY_give_host=pdmn2
   pdmn1# export OCF_ROOT=/usr/lib/ocf
   ```

   b. Transfer ownership of the shelf from `pdmn1` to `pdmn2`:

   ```
   pdmn1# /usr/lib/ocf/resource.d/sgi/copan_ov_client give
   ```

2. On pdmn2:

   a. Verify that pdmn2 now owns the shelf's XVM volumes (C00A through C00Z, although not necessarily listed in alphabetical order):

   ```
   pdmn2# xvm -d local probe | grep C00
   phys/copan_C00M
   phys/copan_C00B
   phys/copan_C00G
   ...
   ```

   b. Create the OpenVault components for pdmn2:

   ```
   pdmn2# ov_shelf create C00
   ```

   For more information, see *COPAN MAID for DMF Quick Start Guide*.

   c. Stop the newly created LCP and DCPs for the shelf:

   ```
   pdmn2# ov_stop C00*
   ```

   d. Export the shelf, hostname, and OCF root environment variables for use by the copan_ov_client script:

   ```
   pdmn2# export OCF_RESKEY_shelf_name=C00
   pdmn2# export OCF_RESKEY_give_host=pdmn1
   pdmn2# export OCF_ROOT=/usr/lib/ocf
   ```

   e. Transfer ownership of the shelf from pdmn2 back to pdmn1:

   ```
   pdmn2# /usr/lib/ocf/resource.d/sgi/copan_ov_client give
   ```

3. On pdmn1, verify that pdmn1 once again owns the shelf's XVM volumes:

   ```
   pdmn1# xvm -d local probe | grep C00
   phys/copan_C00M
   phys/copan_C00B
   phys/copan_C00G
   ...
   ```

4. Repeat steps 1 through 3 for each shelf.

Note: For load-balancing purposes, pdmn1 should be the default node for half of the shelves and pdmn2 should be the default node for the remaining shelves.

Start the GUI

Do the following:

1. Invoke the HA GUI:

   ```
   pdmn1# crm_gui
   ```

2. Log in to the initialized cluster (see Chapter 4, "Outline of the Configuration Procedure" on page 39).

Create the CXFS Client Clone

Do the following to create the CXFS client clone:

1. Select **Resources** in the left-hand navigation panel.
2. Click the **Add** button, select **Clone**, and click **OK**.
3. Enter the ID of the clone, such as `cxfs-client-clone`.
4. Select **Stopped** for the **Initial state of resource** and click **Forward**.
5. Select the sub-resource **Primitive** and click **OK**.
6. Create the primitive for the CXFS client resource: See "Creating the CXFS Client Primitive" on page 159.
7. Click **Apply** to apply the new primitive and again to apply the new clone.

Test the Clone

Use the following steps to test the clone:

1. Start the clone. For example:

   ```
   pdmn1# crm resource start cxfs-client-clone
   ```

   It make take several minutes for the filesystems to mount.

2. Confirm that the clone has started. For example:

   a. View the status of the cluster on pdmn1:

   ```
   pdmn1# crm status
   ============
   Last updated: Tue Aug 23 12:14:44 2011
   Stack: openais
   Current DC: pdmn1 - partition with quorum
   Version: 1.1.5-5bd2b9154d7d9f86d7f56fe0a74072a5a6590c60
   2 Nodes configured, 2 expected votes
   2 Resources configured.
   ============
   Online: [ pdmn1 pdmn2 ]

    Clone Set: cxfs-client-clone [cxfs_client]
        Started: [ pdmn1 pdmn2 ]
   ```

   b. Verify that the cxfs_client process is running on pdmn1 and pdmn2. For example:

   - On pdmn1:

```
pdmn1# ps -ef | grep cxfs_client | grep -v grep
root  11575     1  0 10:32 ?      00:00:00 /usr/cluster/bin/cxfs_client -p /var/run/cxfs_client.pid -i TEST
```

   - On pdmn2:

```
pdmn2# ps -ef | grep cxfs_client | grep -v grep
root  11576     1  0 10:32 ?      00:00:00 /usr/cluster/bin/cxfs_client -p /var/run/cxfs_client.pid -i TEST
```

3. Set pdmn2 to standby state to ensure that the resources remain on pdmn1:

   ```
   pdmn1# crm node standby pdmn2
   ```

4. Confirm that `pdmn2` is offline and that the resources are off:

   a. View the status of the cluster on `pdmn1`, which should show that `pdmn2` is in `standby` state:

   ```
   pdmn1# crm status
       ============
       Last updated: Tue Aug 23 12:16:55 2011
       Stack: openais
       Current DC: pdmn1 - partition with quorum
       Version: 1.1.5-5bd2b9154d7d9f86d7f56fe0a74072a5a6590c60
       2 Nodes configured, 2 expected votes
       2 Resources configured.
       ============

       Node pdmn2: standby
       Online: [ pdmn1 ]

       Clone Set: cxfs-client-clone [cxfs_client]
           Started: [ pdmn1 ]
           Stopped: [ cxfs_client:1 ]
   ```

   b. Verify that the `cxfs_client` process is not running on `pdmn2`. For example, executing the following command on `pdmn2` should provide no output:

```
pdmn2# ps -ef | grep cxfs_client | grep -v grep
pdmn2#
```

5. Return `pdmn2` to online status by executing the following on `pdmn1`:

   ```
   pdmn1# crm node online pdmn2
   ```

6. Confirm that the clone has returned to started status, as described in step 2.

   Note: It may take several minutes for all filesystems to mount successfully.

Create the COPAN OpenVault Client Resources

Do the following to create the COPAN OpenVault client resources:

1. Start the HA GUI if it is not already started.
2. Select **Resources** in the left-hand navigation panel.
3. Click the **Add** button, select **Primitive**, and click **OK**.
4. Create the primitives for the COPAN OpenVault client resources. See "Creating the COPAN OpenVault Client Primitive" on page 162.

Create the Constraints

For each shelf being managed by the COPAN MAID HA cluster, you must create the following:

- Two location constraints:
  - A score of `200` for the default node
  - A score of `100` for the failover node
- One colocation constraint
- One order constraint

Do the following:

1. Select **Constraints** in the left-hand navigation panel of the GUI.
2. Create two location constraints (one for the default node and one for the failover node) for each shelf, using the scores described above. Do the following:
   a. Click the **Add** button, select **Resource Location**, and click **OK**.
   b. Enter the constraint ID, such as `C00_on_pdmn1` for the constraint for shelf `C00` managed by `pdmn1`.
   c. Select the name of the COPAN OpenVault client primitive as the **Resource**.
   d. Enter a score based on whether the node is the default node (`200`) or failover node (`100`).
   e. Enter the node name.

   f. Click **OK**.
   g. Repeat steps 2a through 2f to create the remaining location constraints.

3. Create a colocation constraint for each shelf:
   a. Click the **Add** button, select **Resource Colocation**, and click **OK**.
   b. Enter the ID of the constraint for the shelf, such as `C00_with_cxfs` for shelf `C00`.
   c. Select the name of the COPAN OpenVault client primitive as the **Resource**.
   d. Select the name of the CXFS client clone as the **With Resource**.
   e. Select **INFINITY** for **Score**.
   f. Click **OK**.
   g. Repeat steps 3a through 3f to create the colocation constraint for the remaining shelves.

4. Create an order constraint for each shelf:
   a. Click the **Add** button, select **Resource Order**, and click **OK**.
   b. Enter the ID of the constraint for the shelf, such as `C00_cxfs_first` for shelf `C00`.
   c. Select the name of the CXFS client clone as **First**.
   d. Select the name of the COPAN OpenVault client primitive as **Then**.
   e. Open **Optional** and select **true** for **Symmetrical**.
   f. Click **OK**.
   g. Repeat steps 4a through 4f to create the order constraint for the remaining shelves.

Test the `ov_copan_client` Resource

To test the `copan_ov_client` resource, follow the instructions in "Manually Moving a `copan_ov_client` Resource" on page 178 to move a resource from its default node to its failover node, and then return it to the default node.

CXFS Client Resource

This section discusses the following:

- "Creating the CXFS Client Primitive" on page 159

Creating the CXFS Client Primitive

Use the values shown in the following sections when adding a CXFS client resource primitive for the CXFS client clone.

Note: There are no meta attributes for this primitive in this example procedure because it is part of a clone resource that should always restart locally.

Required Fields for a CXFS Client

ID	*Unique ID such as* `cxfs-client`
Class	**ocf**
Provider	**sgi**
Type	**cxfs-client**

Instance Attributes for a CXFS Client

volnames	*Comma-separated list of CXFS XVM volume names that represent the following DMF configuration file parameters and all DMF-managed user filesystems (originally mounted under* `/dev/cxvm` *as determined in step* 3 *of* "Disable the Parallel Data Mover Nodes and the Services" on page 151): *CACHE_DIR* *SPOOL_DIR* *TMP_DIR* *MOVE_FS* *STORE_DIRECTORY (for a DCM, if used)* *All DMF-managed user filesystems*

For example, suppose you have the following output:

```
# ls /dev/cxvm
cache         dmfusr2       move
diskmsp       home          spool
dmfusr1       journal       tmp
```

You would enter the following (everything except `home` *and* `journal` *):*

```
cache,diskmsp,dmfusr1,dmfusr2,move,spool,tmp
```

Probe Monitor Operation for a CXFS Client

Note: Click the **Operations** tab to edit the monitor operations and to add the probe, start, and stop operations as needed for a resource.

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Monitor Operation for a CXFS Client

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `120s`
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

The monitor operation does the following:

- Verifies that each volume in **volnames** is mounted by checking `/proc/mounts`
- Verifies that the volumes in **volnames** are online by executing the following command for each volume:

  ```
  xvm show -v vol/volname
  ```

- Fails if the CXFS client does not start

Start Operation for a CXFS Client

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `600s`
Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Starts the CXFS client by calling the following:

  ```
  /etc/init.d/cxfs_client start
  ```

- Checks the `/proc/mounts` file until all volumes in **volnames** are mounted
- Fails if the CXFS client fails to start

Stop Operation for a CXFS Client

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `600s`
Optional > On Fail	**fence**

The stop operation does the following:

- Stops the CXFS client by calling the following:

  ```
  /etc/init.d/cxfs_client stop
  ```

- Fails if the CXFS client fails to stop

COPAN OpenVault Client Resource

This section discusses the following:

- "Creating the COPAN OpenVault Client Primitive" on page 162

Creating the COPAN OpenVault Client Primitive

Use the values shown in the following sections when adding a COPAN OpenVault client primitive.

Required Fields for a COPAN OpenVault Client

ID	*Unique ID for the COPAN OpenVault client on each shelf, corresponding to the shelf ID, such as* `copan_C00` for shelf 0, which has a shelf ID of `C00`
Class	**ocf**
Provider	**sgi**
Type	**copan_ov_client**

Meta Attributes for a COPAN OpenVault Client

resource-stickiness	**250**
migration-threshold	**1**
target-role	**Started**

Instance Attributes for a COPAN OpenVault Client

shelf_name	*The three-character shelf ID, using the naming convention described in the COPAN MAID for DMF Quick Start Guide, such as* `C00` *for cabinet 0 shelf 0.*

Probe Monitor Operation for a COPAN OpenVault Client

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `60s`

The probe operation checks to see if the resource is already running.

Monitor Operation for a COPAN OpenVault Client

ID	*(Generated based on the primitive name and interval)*
name	**monitor**

Interval	*Interval time, such as* `120s`
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

The monitor operation does the following:

- Verifies that OpenVault is functional: the LCP and at least one DCP are running and at least one RAID set is accessible on the shelf
- Fails if the COPAN MAID shelf is not functional

Start Operation for a COPAN OpenVault Client

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `120s`
Optional > Requires	**fencing**
Optional > On Fail	**restart**

The start operation does the following:

- Starts the COPAN MAID shelf client by ensuring that RAID sets are available and the OpenVault LCP and at least one DCP are running
- Fails if any of the above conditions are not met

Stop Operation for a COPAN OpenVault Client

ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `120s`
Optional > On Fail	**fence**

The stop operation does the following:

- Stops the COPAN OpenVault client resource
- Fails if the COPAN OpenVault client resource fails to stop

STONITH Examples

This chapter discusses the following:

- "Overview of STONITH Resources" on page 165
- "SGI IPMI STONITH Examples" on page 165
- "Community IPMI STONITH Examples" on page 168

Overview of STONITH Resources

STONITH (*"shoot the other node in the head"*) node-level fencing is required in order to protect data integrity in case of failure:

- `sgi-ipmi` for x86_64 systems using baseboard management controller (BMC) and intelligent platform management interface (IPMI) network reset
- `ipmi` for x86_64 systems using baseboard management controller (BMC) and intelligent platform management interface (IPMI) network reset

SGI IPMI STONITH Examples

This section discusses examples of the following:

- "Creating the SGI IPMI STONITH Clone" on page 165
- "Creating the SGI IPMI STONITH Primitive " on page 166
- "Testing the SGI IPMI STONITH Resource" on page 168

Creating the SGI IPMI STONITH Clone

To create the IPMI STONITH clone, do the following:

1. Select **Resources** in the left-hand navigation panel.
2. Click the **Add** button, select **Clone**, and click **OK**.
3. Enter the name of the clone (such as `stonith-sgi-ipmi-set`) in the **ID** field.

4. Use the **target-role** of **Started** and the default options (2 maximum number of copies and 1 number of copies on a single node) and click **Forward**.
5. Select **OK** to add a **Primitive**. Add the STONITH primitive according to the steps described in "Creating the SGI IPMI STONITH Primitive " on page 166.

Creating the SGI IPMI STONITH Primitive

Required Fields for SGI IPMI STONITH

ID	*Unique ID such as* `stonith-sgi-ipmi`
Class	**stonith**
Type	**external/sgi-ipmi**

Instance Attributes for SGI IPMI STONITH

nodelist

Nodes to be acted upon, such as:

```
node1;admin;admin;supermicro;128.162.245.170
node2;admin;admin;supermicro;128.162.245.171
```

You must provide the following information for each node (each field is separated by a semicolon and each node is separated by a space):

nodename`;`*userID*`;`*IPMIpass*`;`*BMCtype*`;`*IPMIipaddr1*`[;`*IPMIipaddrN*`]`

where the fields are:

- Node name (such as `node1`)
- User ID to use on the IPMI device (such as the default `admin`)
- IPMI device password (such as the default `admin`)
- BMC type of the IPMI device:
  - `intel` for Intel® BMC
  - `supermicro` for Supermicro® BMC

- IPMI device IP address (such as `128.162.245.170`)

Probe Monitor Operation for SGI IPMI STONITH

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `300s`

The probe operation checks to see if the resource is already running.

Monitor Operation for SGI IPMI STONITH

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `300s`
Timeout	*Timeout, such as* `300s`
Optional > On Fail	**restart**

The monitor operation calls the defined STONITH resource agent.

Start Operation for SGI IPMI STONITH

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

The start operation does the following:

- Initializes the information required for the `stonithd` daemon to act when necessary
- Fails if the information cannot be initialized

Testing the SGI IPMI STONITH Resource

To test the IPMI STONITH resource, do the following:

1. Reset a node:

   ```
   ha# crm node fence nodename
   ```

   For example, to reset `node1`:

   ```
   ha# crm node fence node1
   ```

2. Verify that the specified node was reset and was able to successfully complete a reboot.

Community IPMI STONITH Examples

This section discusses examples of the following:

- "Creating the Community IPMI STONITH Constraints" on page 168
- "Creating the Community IPMI STONITH Primitive " on page 169
- "Testing the Community IPMI STONITH Resource" on page 170

Creating the Community IPMI STONITH Constraints

To use the community IPMI STONITH facility, do the following for each node in the HA cluster:

1. Create an `ipmi` resource for the node.
2. Create a constraint so that the `ipmi` resource will never run on its corresponding node. Choose one of the following methods:
   - To allow the `ipmi` resource to run on any other node in the HA cluster, create a constraint with a score of `-INFINITY` for the corresponding node. For example, to allow any other node in the cluster to perform STONITH fencing for `node1`:

```
node1# location cli-prefer-node1-stonith stonith-ipmi-node1 -inf: node1
```

- To use a specific node for performing STONITH fencing, create a constraint with a score of `INFINITY` for that specific node. For example, to use `node2` to perform STONITH fencing for `node1`:

```
node1# location cli-prefer-node1-stonith stonith-ipmi-node1 inf: node2
```

Creating the Community IPMI STONITH Primitive

Required Fields for Community IPMI STONITH

ID	*Unique ID such as* `stonith-ipmi-node1`
Class	**stonith**
Type	**external/ipmi**

Instance Attributes for Community IPMI STONITH

hostname	The hostname of the node to be managed by this BMC
ipaddr	The IP address of the BMC
userid	The user name used for logging in to the BMC
passwd	The password used for logging in to the BMC
interface	The IPMI interface to use, one of: • `lanplus` for most SGI systems • `lanplus -o intelplus` for older Altix XE210 or Altix XE240 systems

Probe Monitor Operation for Community IPMI STONITH

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `15s`

The probe operation checks to see if the resource is already running.

Monitor Operation for Community IPMI STONITH

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	*Interval time, such as* `300s`
Timeout	*Timeout, such as* `15s`
Optional > On Fail	**restart**

The monitor operation calls the defined STONITH resource agent.

Start Operation for Community IPMI STONITH

ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `15s`
Optional > On Fail	**restart**

The start operation does the following:

- Initializes the information required for the `stonithd` daemon to act when necessary
- Fails if the information cannot be initialized

Testing the Community IPMI STONITH Resource

To test the community IPMI STONITH resource, do the following:

1. Reset a node:

```
ha# crm node fence nodename
```

For example, to reset `node1`:

```
ha# crm node fence node1
```

2. Verify that the specified node was reset and was able to successfully complete a reboot.

Chapter 10

Administrative Tasks and Considerations

This chapter discusses various administrative tasks and considerations for an HA cluster:

- "Putting the Cluster into Maintenance Mode" on page 174
- "Viewing the Contents of the CIB" on page 174
- "Backing Up the CIB" on page 174
- "Understanding CIFS and NFS in an HA Cluster" on page 175
- "Reviewing the Log File" on page 175
- "Clearing the Resource Primitive Failcount" on page 175
- "Clearing the Resource State on a Node" on page 175
- "Controlling the Number of Historical Files" on page 176
- "Changing DMF Configuration Parameters" on page 177
- "Restarting the OpenVault Server" on page 177
- "Manually Moving a `copan_ov_client` Resource" on page 178
- "Performing a Rolling Upgrade" on page 180
- "Stopping the `openais` Service" on page 186
- "Manually Resetting a Node" on page 186
- "Hardware Maintenance on a Cluster Node" on page 186
- "Maintenance with a Full Cluster Outage" on page 189

Putting the Cluster into Maintenance Mode

You must put the cluster into maintenance mode before manually stopping or restarting any cluster components. To put the cluster into maintenance mode, enter the following:

```
ha# crm configure property maintenance-mode=true
```

You can then manually stop and restart individual resources as needed.

To return the cluster to managed status, enter the following:

```
ha# crm configure property maintenance-mode=false
```

Viewing the Contents of the CIB

If you want to view the contents of the cluster information base (CIB), you can use the following `cibadmin`(8) command to show configuration and status information:

```
ha# cibadmin -o modifier -Q
```

The *modifier* value can be one of the following:

```
constraints
crm_config
nodes
resources
status
```

Backing Up the CIB

You should make a backup copy of the configuration in the cluster information base (CIB) after making and verifying changes, so that you can easily recover in case of future CIB corruption (see "Recovering from a CIB Corruption" on page 206). Do the following to save only static configuration information to a plain text file labeled with the current date and time:

```
ha# crm configure save CIB.$(date +%Y%m%d-%H%M%S)
```

You can view the resulting text file with any text tool, such as `cat`(1) or `vi`(1).

Understanding CIFS and NFS in an HA Cluster

CIFS failover requires that the client application reissue the I/O after the failover occurs. Applications such as XCOPY will do this, but many other applications will not. Applications that do not retry may abort when CIFS services are moved between nodes.

NFS failover is handled by the kernel, so no changes are required for an NFS client application; applications doing I/O on NFS will pause while the failover is occurring.

Reviewing the Log File

You will find information in the `/var/log/messages` log file. To turn on debug messages, see "Increase the Verbosity of Error Messages" on page 196.

Clearing the Resource Primitive Failcount

To clear resource primitive failcounts, either reboot the nodes or enter the following on each node for each resource primitive:

```
ha# crm resource failcount resourcePRIMITIVE delete nodename
```

Clearing the Resource State on a Node

Caution: Do not clear the resource state on the node where a resource is currently running.

After you resolve the cause of `action` error messages in the `crm status` output, you should enter the following to clear the resource state from a given node:

```
ha# crm resource cleanup resourcePRIMITIVE nodename
```

Note: Sometimes, the resource state can be cleared automatically if the same action for the same resource on the same node subsequently completes successfully.

Controlling the Number of Historical Files

Each time the configuration is updated, a new version of the CIB is created and the older version is saved. These files reside in `/var/lib/heartbeat/crm`. SGI recommends that you keep the number of files manageable by setting the following `crm` properties as appropriate for your site:

```
pe-error-series-max
pe-input-series-max
pe-warn-series-max
```

To set the properties, use the following command line:

```
# crm configure property propertyname=value
```

For example, to set a maximum of 50 error, input, and warning files:

```
ha# crm configure property pe-error-series-max=50
ha# crm configure property pe-input-series-max=50
ha# crm configure property pe-warn-series-max=50
```

For more information, see the SUSE *High Availability Guide*.

Changing DMF Configuration Parameters

You can change many DMF configuration file parameters while DMF is running, but others require that DMF be stopped. For more information, see the "Best Practices" chapter in *DMF 6 Administrator Guide for SGI InfiniteStorage*). For those parameters that required DMF to be stopped, do the following:

1. Put the cluster into maintenance mode:

   ```
   ha# crm configure property maintenance-mode=true
   ```

2. Stop the DMF service:

   ```
   ha# service dmf stop
   ```

3. Make the required changes to the DMF configuration file according to the instructions in the DMF administrator's guide, such as by using DMF Manager.

4. Verify the parameter changes by using DMF Manager or the following command:

   ```
   ha# dmcheck
   ```

5. Start the DMF service:

   ```
   ha# service dmf start
   ```

6. Verify DMF functionality, such as by running the following command and other DMF commands (based on the changes made):

   ```
   ha# dmdstat -v
   ```

7. Return the cluster to managed status:

   ```
   ha# crm configure property maintenance-mode=false
   ```

Restarting the OpenVault Server

To restart the OpenVault server, do the following:

1. Put the HA cluster into maintenance mode:

   ```
   ha# crm configure property maintenance-mode=true
   ```

2. Stop the OpenVault service:

   ```
   ha# service openvault stop
   ```

3. Start the OpenVault service:

   ```
   ha# service openvault start
   ```

4. Return the HA cluster to managed status:

   ```
   ha# crm configure property maintenance-mode=false
   ```

Manually Moving a `copan_ov_client` Resource

You may want to manually move a `copan_ov_client` resource as followings:

- To its failover node when you want to perform maintenance on its default node
- Back to its default node, after maintenance is complete on the default node
- Back to its default node, after the formerly failed default node rejoins the HA cluster

Before you can move the resource from one node to another, you must ensure that you stop any activity occurring on the COPAN MAID shelf that is managed by the resource. This requires that you disable the mover capability on the currently active node (which stops activity for all shelves owned by that node).

You must also ensure that the new node is ready to receive the resource:

- The node must be online
- The CXFS client and STONITH services must be operational on the node

For example, to move the `copan_maid_C01` resource from parallel data mover node `pdmn1` to parallel data mover node `pdmn2`:

1. Verify that `pdmn2` is ready to receive the resource by examining its status output with the `crm`(8) command:

   ```
   pdmn2# crm status
   ============
   Last updated: Tue Aug 30 07:46:58 2011
   Stack: openais
   Current DC: pdmn2 - partition with quorum
   Version: 1.1.5-5bd2b9154d7d9f86d7f56fe0a74072a5a6590c60
   2 Nodes configured, 2 expected votes
   6 Resources configured.
       ============
   ```

```
Online: [ pdmn1 pdmn2 ]

Clone Set: cxfs-client-clone [cxfs_client]
    Started: [ pdmn1 pdmn2 ]
copan_maid_C00 (ocf::sgi:copan_ov_client):     Started pdmn1
copan_maid_C01 (ocf::sgi:copan_ov_client):     Started pdmn1
Clone Set: stonith-sgi-ipmi-set [stonith-sgi-ipmi]
    Started: [ pdmn1 pdmn2 ]
```

In the above output, note the following:

- The `copan_maid_C01` resource is `Started` on `pdmn1`
- `pdmn2` is `Online`
- The `cxfs-client` and `stonith-sgi-ipmi` clones have a status of `Started` on `pdmn2`

2. On the DMF server, disable `pdmn1` so that it will cease all COPAN MAID shelf activity:

```
dmfserver# dmnode_admin -d pdmn1
```

3. Verify that there are no `dmatwc` or `dmatrc` data mover processes running on `pdmn1`. For example, the output of the following command should be empty:

```
pdmn1# ps -ef | egrep ’dmatrc|dmatwc’ | grep -v grep
pdmn1#
```

If the output is not empty, you must wait for the `dmnode_admin -d` action from step 2 to complete (the entire process can take 6 minutes or longer). Rerun the `ps` command until there is no output.

4. Clear any failcounts and move the resource to `pdmn2`:

```
pdmn2# crm resource failcount copan_maid_C01 delete pdmn2
pdmn2# crm resource move copan_maid_C01 pdmn2
```

This make take a few moments to complete.

5. Verify that the resource has moved to `pdmn2`:

   ```
   pdmn2# crm status
   ============
   Last updated: Tue Aug 30 07:46:58 2011
   Stack: openais
   Current DC: pdmn2 - partition with quorum
   Version: 1.1.5-5bd2b9154d7d9f86d7f56fe0a74072a5a6590c60
   2 Nodes configured, 2 expected votes
   6 Resources configured.
   ============

   Online: [ pdmn1 pdmn2 ]

   Clone Set: cxfs-client-clone [cxfs_client]
       Started: [ pdmn1 pdmn2 ]
   copan_maid_C00 (ocf::sgi:copan_ov_client):     Started pdmn1
   copan_maid_C01 (ocf::sgi:copan_ov_client):     Started pdmn2
   Clone Set: stonith-sgi-ipmi-set [stonith-sgi-ipmi]
       Started: [ pdmn1 pdmn2 ]
   ```

   In the above output, note that `copan_maid_C01` resource is started on `pdmn2`.

6. Remove the constraint to run `copan_maid_C01` on `pdmn2`:

   ```
   pdmn2# crm resource unmove copan_maid_C01
   ```

7. Repeat steps 3 through 6 for any other `copan_ov_client` resources requiring a move from `pdmn1` to `pdmn2`.

8. On the DMF server, reenable `pdmn1` so that it can resume COPAN MAID shelf activity:

   ```
   dmfserver# dmnode_admin -e pdmn1
   ```

Performing a Rolling Upgrade

Note: Some software may not allow a rolling upgrade. Such a situation might require an extended outage window with all resources down and the `openais` service turned off, which would permit more thorough testing (similar to that done during the initial installation). See "Maintenance with a Full Cluster Outage" on page 189.

Assuming that you have a two-node production HA environment in place and want to perform a rolling upgrade of appropriate software with minimal testing, use the procedures in the following sections:

- "CXFS NFS Edge-Serving HA Rolling Upgrade" on page 181
- "DMF HA Rolling Upgrade" on page 183
- "COPAN MAID HA Service for Mover Nodes Rolling Upgrade" on page 185

CXFS NFS Edge-Serving HA Rolling Upgrade

Do the following for a rolling upgrade in a CXFS NFS edge-serving HA cluster with two nodes (`node1` and `node1`):

1. Read the release notes for the software you intend to upgrade.

   For ISSP software, read the *SGI InfiniteStorage Software Platform Release Note* and any late-breaking caveats on Supportfolio™ Online:

   https://support.sgi.com/login

2. Ensure that the HA cluster, the underlying CXFS cluster, and all other hardware and software components are in a healthy state. Ensure that all resource failcounts are cleared and that no constraints are present other than permanent constraints.

3. Ensure that the resource groups are running on `node1`:

   ```
   node1# crm resource move ipalias-group-1 node1
   node1# crm resource move ipalias-group-2 node1
   ```

4. Set the node you intend to upgrade to `standby` state. (Putting a node in `standby` state will move, if possible, or stop any resources that are running on that node.) For example, if you intend to upgrade `node2`:

   ```
   node1# crm node standby node2
   ```

   This will shut down `cxfs_client` on `node2` automatically.

5. Disable the `openais` service from being started automatically at boot time and stop the currently running service:

   ```
   node2# chkconfig openais off
   node2# service openais stop
   ```

6. Upgrade the software on `node2`.

7. Reboot `node2`.

8. Enable `openais` to be started automatically at boot time and immediately start it on `node2`:

   ```
   node2# chkconfig openais on
   node2# service openais start
   ```

9. Make `node2` active again:

   ```
   node1# crm node online node2
   ```

10. Move the resource groups from `node1` to `node2`:

    ```
    node1# crm resource move ipalias-group-1 node2
    node1# crm resource move ipalias-group-2 node2
    ```

11. *(Optional)* Allow the resource groups to run on `node2` for a period of time as a test.

12. Repeat steps 4 through 11 above but switching the roles for `node1` and `node2`.

Note: In most cases, you will want to leave the resource groups running on `node2` in order to avoid any unnecessary interruptions to the services that would have to be restarted if they were moved to `node1`. However, if you prefer to have the resource groups run on `node1` despite any potential interruptions, do the following:

1. Move the appropriate resource group from `node2` back to `node1`. For example:

   ```
   node1# crm resource move ipalias-group-1 node1
   ```

2. Remove the implicit location constraints imposed by the administrative `move` command above:

   ```
   node1# crm resource unmove ipalias-group-1
   node1# crm resource unmove ipalias-group-2
   ```

DMF HA Rolling Upgrade

Do the following for a rolling upgrade in a DMF HA cluster with two nodes (`node1` and `node2`):

1. Read the release notes for the software you intend to upgrade.

   For ISSP software, read the *SGI InfiniteStorage Software Platform Release Note* and any late-breaking caveats on Supportfolio Online:

   https://support.sgi.com/login

2. Ensure that the HA cluster, the underlying CXFS cluster (if applicable), and all other hardware and software components are in a healthy state. Ensure that all resource failcounts are cleared and that no constraints are present other than intended constraints.

3. Disable the services from being started automatically at boot time:

   ```
   node2# chkconfig openais off
   node2# chkconfig cxfs off
   node2# chkconfig cxfs_cluster off
   ```

4. Ensure that the resource groups are running on `node1`.

   Note: Moving the `dmfGroup` resource group will involve CXFS relocation of the DMF administrative filesystems and DMF managed user filesystems. However, you cannot use CXFS relocation if your CXFS cluster also includes a CXFS NFS edge-server HA pair and the CXFS server-capable administration nodes are running different software levels. If that is the case, you must move the `dmfGroup` resource group via CXFS recovery by resetting the node that is running the `dmfGroup` resource.

   Do one of the following, as appropriate for your site:

   - Using CXFS relocation:

     Note: Stopping `openais` will cause a failover if there are resources running on the node. Depending on how things are defined and whether the resource `stop` actions succeed, it might even cause the node to be reset.

```
node2# crm resource move dmfGroup node1
node2# service openais stop
node2# service cxfs stop
node2# service cxfs_cluster stop
```

- Using CXFS recovery:

  ```
  node2# crm node fence node1
  ```

5. Upgrade the software on node2 and reboot.
6. Add node2 back into the CXFS cluster (if present) by enabling the services to be started automatically at boot time and then starting them immediately:

   ```
   node2# chkconfig cxfs_cluster on
   node2# chkconfig cxfs on
   node2# service cxfs_cluster start
   node2# service cxfs start
   ```

7. Verify that node2 is fully back in the CXFS cluster with filesystems mounted.
8. Enable the openais service to be started automatically at boot time and then start it immediately on node2:

   ```
   node2# chkconfig openais on
   node2# service openais start
   ```

9. Repeat steps 3 through 8 above but executed for node1.

Note: In most cases, you will want to leave the resource group running on node2 in order to avoid any unnecessary interruptions to the services that would have to be restarted if they were moved to node1. However, if you prefer to have the resource group run on node1 despite any potential interruptions, do the following:

1. Move the appropriate resource groups from node2 back to node1:

   ```
   node1# crm resource move dmfGroup node1
   ```

2. Remove the implicit location constraints imposed by the administrative move command above:

   ```
   node1# crm resource unmove dmfGroup
   ```

   The cluster is now back to normal operational state.

COPAN MAID HA Service for Mover Nodes Rolling Upgrade

Do the following for a rolling upgrade in an HA cluster that consists of two parallel data mover nodes (`pdmn1` and `pdmn1`):

1. Read the release notes for the software you intend to upgrade.

   For ISSP software, read the *SGI InfiniteStorage Software Platform Release Note* and any late-breaking caveats on Supportfolio™ Online:

   https://support.sgi.com/login

2. Ensure that the HA cluster, the underlying CXFS cluster, and all other hardware and software components are in a healthy state. Ensure that all resource failcounts are cleared and that no constraints are present other than permanent constraints.

3. Move any `copan_ov_client` resources running on `pdmn2` to `pdmn1` according to the directions in "Manually Moving a `copan_ov_client` Resource" on page 178.

4. Stop all services on `pdmn2` and ensure that they will not restart on the next boot of `pdmn2`:

   ```
   pdmn2# service openais stop
   pdmn2# chkconfig openais off
   ```

5. Upgrade the software on `pdmn2`.

6. Reboot `pdmn2`.

7. Ensure that the services will restart upon reboot and start services immediately:

   ```
   pdmn2# chkconfig openais on
   pdmn2# service openais start
   ```

8. Move all `copan_ov_client` resources running on `pdmn1` to `pdmn2` according to the directions in "Manually Moving a `copan_ov_client` Resource" on page 178.

9. Repeat step 4 through step 7 above but executed for `pdmn1`.

10. Move the `copan_ov_client` resources than normally run on `pdmn1` back to that node, according to the directions in "Manually Moving a `copan_ov_client` Resource" on page 178.

11. Remove the constraints required to move the resources by executing the following command for each `copan_ov_client` resource

    ```
    pdmn2# crm resource unmove copan_ov_client_name
    ```

The cluster is now back to normal operational state.

Stopping the `openais` Service

Note: Stopping `openais` will cause a failover if there are resources running on the node. Depending on how things are defined and whether the resource `stop` actions succeed, it might even cause the node to be reset.

To stop the `openais` service on the local node, enter the following:

```
ha# service openais stop
```

Manually Resetting a Node

To manually issue a system reset of a given node, execute the following from a different node in the HA cluster:

```
ha# crm node fence node_to_be_reset
```

Note: This command requires that the `stonith` resource is defined in the CIB.

For example, to reset `node1` from `node2`:

```
node2# crm node fence node1
```

Hardware Maintenance on a Cluster Node

If you must perform maintenance on one node in the cluster, do the following:

1. On the node that requires maintenance (`downnode`), disable the `openais` service from being started automatically at boot time:

   ```
   downnode# chkconfig openais off
   ```

2. If downnode is a CXFS server-capable administration node, disable the cxfs and cxfs_cluster services from being started automatically at boot time:

   ```
   downnode# chkconfig cxfs off
   downnode# chkconfig cxfs_cluster off
   ```

3. As a precaution, run the sync(8) command to flush disk buffers, synchronizing the data on disk with memory:

   ```
   downnode# sync
   ```

4. Reset downnode in order to force resources to be moved to upnode:

   ```
   upnode# crm node fence downnode
   ```

5. Verify that resources are running on upnode:

   ```
   upnode# crm status
   ```

6. Perform the required maintenance on downnode.

7. Reboot downnode and ensure that is stable before proceeding.

8. If downnode was a CXFS server-capable administration node, do the following:

   a. Enable the cxfs and cxfs_cluster services to start automatically at boot time and start them immediately on downnode:

      ```
      downnode# chkconfig cxfs_cluster on
      downnode# chkconfig cxfs on
      downnode# service cxfs_cluster start
      downnode# service cxfs start
      ```

   b. Verify that CXFS is functioning properly on downnode, such as if the node joined the cluster and mounted filesystems.

9. Enable the openais service to be started automatically at boot time and start it immediately on downnode:

   ```
   downnode# chkconfig openais on
   downnode# service openais start
   ```

10. Verify that downnode rejoins the HA cluster:

    ```
    downnode# crm status
    ```

Note: In most cases, you will want to leave the resources running on `upnode` in order to avoid any unnecessary interruptions to the services that would have to be restarted if they were moved to `downnode`. However, if you prefer to have the resources run on `downnode` despite any potential interruptions, do the following:

1. Restart resources on `downnode`:

   ```
   downnode# crm resource move ResourceName downnode
   ```

2. Remove the implicit location constraints imposed by the administrative `move`:

   ```
   downnode# crm resource unmove ResourceName
   ```

Maintenance with a Full Cluster Outage

This section discusses the following:

- "Full Outage for CXFS NFS Edge-Serving HA" on page 189
- "Full Outage for DMF HA" on page 191
- "Full Outage for COPAN MAID OpenVault Client HA on Mover Nodes" on page 193

Full Outage for CXFS NFS Edge-Serving HA

Do the following to perform a full outate for a CXFS NFS edge-serving HA cluster:

1. Schedule the outage and notify users well in advance.
2. Stop all resources in the proper order (bottom up). For example, using the example procedures in this guide, you would stop the IP alias resource groups and the clone:

   ```
   ha# crm resource stop ipalias-group-2
   ha# crm resource stop ipalias-group-1
   ha# crm resource stop cxfs-nfs-clone
   ```

3. Disable the services related to HA and CXFS from being started automatically at boot time:

   - On all HA servers:

     ```
     ha# chkconfig openais off
     ```

   - On all CXFS servers:

     ```
     cxfsserver# chkconfig cxfs off
     cxfsserver# chkconfig cxfs_cluster off
     ```

   - On all CXFS clients:

     ```
     cxfsclient# chkconfig cxfs_client off
     ```

4. Shut down all of the HA cluster systems and the CXFS cluster systems.
5. Perform the required maintenance.
6. Perform component-level testing associated with the maintenance.

7. Reboot all of the HA cluster systems and the CXFS cluster systems.
8. Enable the services related to CXFS to be started automatically at boot time and start them immediately as follows:
   - On all CXFS servers:

     ```
     cxfsserver# chkconfig cxfs_cluster on
     cxfsserver# chkconfig cxfs on
     cxfsserver# service cxfs_cluster start
     cxfsserver# service cxfs start
     ```

   - On all CXFS clients:

     ```
     cxfsclient# chkconfig cxfs_client on
     cxfsclient# service cxfs_client start
     ```

   - Verify CXFS cluster functionality:

     ```
     cxfsserver# /usr/cluster/bin/cxfs_admin -c status
     ```

9. On the NFS edge servers, disable the `cxfs_client` service from being started automatically at boot time and stop the currently running service immediately:

   ```
   edge# chkconfig cxfs_client off
   edge# service cxfs_client stop
   ```

10. On the HA servers, disable the `openais` service from being started automatically at boot time and stop the currently running service immediately:

    ```
    ha# chkconfig openais on
    ha# service openais start
    ```

11. Verify the HA cluster status:

    ```
    ha# crm status
    ```

12. Start resources in the correct order (top-down). For example:

    ```
    ha# crm resource stop cxfs-nfs-clone
    ha# crm resource stop ipalias-group-1
    ha# crm resource stop ipalias-group-2
    ```

13. Move the resources to the correct locations. For example:

    ```
    ha# crm resource move ipalias-group-1 node1
    ha# crm resource move ipalias-group-2 node2
    ```

14. Remove the implicit location constraints imposed by the administrative `move` command above. For example:

    ```
    ha# crm resource unmove ipalias-group-1
    ha# crm resource unmove ipalias-group-2
    ```

Full Outage for DMF HA

Do the following to perform a full outage for a DMF HA cluster::

1. Schedule the outage and notify users well in advance.
2. Stop all resources in the proper order (bottom-up). Using the example procedures in this guide, you would stop the group:

   ```
   ha# crm resource stop dmfGroup
   ```

3. Disable the services related to HA and CXFS (if applicable) from being started automatically at boot time:

   - On all HA servers:

     ```
     ha# chkconfig openais off
     ```

   - On all CXFS servers:

     ```
     cxfsserver# chkconfig cxfs off
     cxfsserver# chkconfig cxfs_cluster off
     ```

4. Shut down all of the HA cluster systems and any CXFS cluster systems.
5. Perform the required maintenance.
6. Perform component-level testing associated with the maintenance.
7. Reboot all of the HA cluster systems and any CXFS cluster systems.
8. Enable the following services related to CXFS (if applicable) to be started automatically at boot time and start them immediately:

- On all CXFS servers:

  ```
  cxfsserver# chkconfig cxfs_cluster on
  cxfsserver# chkconfig cxfs on
  cxfsserver# service cxfs_cluster start
  cxfsserver# service cxfs start
  ```

- On all CXFS clients:

  ```
  cxfsclient# chkconfig cxfs_client on
  cxfsclient# service cxfs_client start
  ```

- Verify CXFS cluster functionality:

  ```
  cxfsserver# /usr/cluster/bin/cxfs_admin -c status
  ```

9. On the HA servers, enable the `openais` service to be started automatically at boot time and start the service immediately

   ```
   ha# chkconfig openais on
   ha# service openais start
   ```

10. Verify HA cluster status:

    ```
    ha# crm status
    ```

11. Start resources in the correct order (top-down). For example:

    ```
    ha# crm resource start dmfGroup
    ```

12. Move the resources to the correct locations:

    ```
    ha# crm resource move dmfGroup node1
    ```

    Note: Keep in mind any relocation restrictions.

13. Remove the implicit location constraints imposed by the administrative `move` command above:

    ```
    ha# crm resource unmove dmfGroup
    ```

Full Outage for COPAN MAID OpenVault Client HA on Mover Nodes

Do the following to perform a full outage for a COPAN MAID OpenVault HA cluster on parallel data mover nodes:

1. On the DMF server, disable both parallel data mover nodes so that they will cease all COPAN MAID shelf activity:

   ```
   dmfserver# dmnode_admin -d pdmn1
   dmfserver# dmnode_admin -d pdmn2
   ```

2. Verify that there are no `dmatwc` or `dmatrc` data mover processes running on either node. For example, the output of the following command should be empty on each parallel data mover node:

   ```
   ha# ps -ef | egrep 'dmatrc|dmatwc' | grep -v grep
   ha#
   ```

   If the output is not empty, you must wait for the `dmnode_admin -d` action from step 1 to complete (the entire process can take 6 minutes or longer). Rerun the `ps` command until there is no output.

3. On both parallel data mover nodes, disable services related to HA:

   ```
   ha# chkconfig openais off
   ```

4. On both nodes, stop HA services (do this simultaneously):

   ```
   ha# service openais stop
   ```

5. Shut down both nodes.

6. Perform the required maintenance.

7. Perform component-level testing associated with the maintenance.

8. Reboot both nodes

9. On both nodes, enable services related to HA:

   ```
   ha# chkconfig openais on
   ```

10. On both nodes, start HA services (do this simultaneously):

    ```
    ha# service openais start
    ```

11. On either node, verify cluster status:

```
ha# crm status
```

12. On the DMF server, reenable both nodes for COPAN MAID activity:

```
dmfserver# dmnode_admin -e pdmn1
dmfserver# dmnode_admin -e pdmn2
```

Troubleshooting

This chapter discusses the following:

- "Diagnosing Problems" on page 195
- "Failover Testing Strategies" on page 202
- "Corrective Actions" on page 205

For details about troubleshooting High Availability Extension (HAE), see the SUSE *High Availability Guide*.

Diagnosing Problems

If you notice problems, do the following:

- "Monitor the Status Output" on page 195
- "Verify the Configuration in Greater Detail" on page 196
- "Increase the Verbosity of Error Messages" on page 196
- "Match Status Events To Error Messages" on page 196
- "Verify `chkconfig` Settings" on page 197
- "Diagnose the Problem Resource" on page 197
- "Examine Application-Specific Problems that Impact HA" on page 197
- "Test the STONITH Capability" on page 198
- "Gather Troubleshooting Data" on page 198
- "Use SGI Knowledgebase" on page 201

Monitor the Status Output

Use the `crm status` command to determine the current status of the cluster and monitor it for problems.

Verify the Configuration in Greater Detail

Execute the `crm_verify`(8) command with increasing numbers of `-V` options for more detail, such as:

```
ha# crm_verify  -LVVVVVV
```

Note: If you run `crm_verify` before STONITH is enabled, you will see errors. Errors similar to the following may be ignored if STONITH is intentionally disabled and will go away after STONITH is reenabled (line breaks shown here for readability):

```
crm_verify[182641]: 2008/07/11_16:26:54 ERROR: unpack_operation:
Specifying on_fail=fence and
stonith-enabled=false makes no sense
```

Increase the Verbosity of Error Messages

For additional information, turn on debug messages in the `logging` stanza of the `/etc/corosync/corosync.conf` file:

```
logging{
...
        debug:on|off
...
 }
```

The default for `debug` is `off`.

Match Status Events To Error Messages

Match the events listed in the `crm_verify` output with the failed action and the host on which the action failed. To find the specific problem, view messages in `/var/log/messages`.

Verify chkconfig Settings

Verify the `chkconfig` settings for the following services when used in an HA cluster:

- Must be `off` if used (most services):

  ```
  cxfs_client
  dmf
  dmfman
  dmfsoap
  nfsserver
  openvault
  smb
  nmb
  ```

- Must be `on` if used:

  ```
  cxfs
  cxfs_cluster
  openais
  logd
  ```

- Optionally may optionally be `on`:

  ```
  tmf
  ```

Diagnose the Problem Resource

To diagnose problems at the application level, put the cluster into maintenance mode. You might have to stop the `openais` service on all nodes and start/stop resources manually.

Caution: Ensure that you do not start a resource on multiple nodes. Verify that a resource is not already up on another node before you start it.

Examine Application-Specific Problems that Impact HA

Using HA can highlight existing problems for the applications that are being managed. For more information about diagnosing application-specific problems, see the manuals listed in "About this Guide".

Test the STONITH Capability

To test the STONITH capability, do the following:

1. Reset a given node from another node, using one of the following methods:
   - Requires that STONITH is defined in the CIB:

```
ha# crm node fence node_to_be_reset
```

     For example, to reset node1 from node2:

```
node2# crm node fence node1
```

   - Independent of the CIB configuration:

```
ha# ipmitool -I lanplus -U admin -P admin -H bmc_IP_of_node_to_be_reset power reset
```

     For example, to reset node1 (whose BMC has an IP address of 128.162.232.79) from node2:

```
node2# ipmitool -I lanplus -U admin -P admin -H 128.162.232.79 power reset
```

2. Verify that the specified node was reset and was able to successfully complete a reboot.

Gather Troubleshooting Data

If you need to report problems to SGI Support, do the following to gather important troubleshooting data:

- "Collect System Configuration Information" on page 199
- "Collect System Logs" on page 199
- "Collect HA Cluster Information" on page 199
- "Collect SGI Service-Specific Information" on page 200
- "Generate a Kernel Crash Dump" on page 200

When you contact SGI Support, you will be provided with information on how and where to upload the collected information files for SGI analysis.

Collect System Configuration Information

Run the following commands as `root` on every node in the cluster in order to gather system configuration information:

```
ha# /usr/sbin/system_info_gather  -A  -o node.out
ha# /sbin/supportconfig
```

Collect System Logs

Collect any other system log files that may contain information about the `openais` service or the services included in the HA configuration (if not otherwise gathered by the above tools).

Collect HA Cluster Information

Collect HA cluster information by using the `hb_report` command:

```
ha# hb_report -f priortime  destination_directory
```

where:

- *priortime* specifies a time prior to when the problem began (specify *priortime* in `Date::Parse` Perl module format)
- *destination_directory* is the absolute pathname of a nonexistent directory that will be created as a compressed `bunzip2` tarball in the format:

  destination_directory`.tar.bz2`

For example, if run at 3:06 PM on June 2, 2010, the following command line will create a report starting from 1:00 AM (0100) that day and place the output in `/tmp/hb_report.20100602-1506.tar.bz2`:

```
ha# hb_report -f 1am /tmp/hb_report.$(date+%Y%m%d-%H%M)
```

For example, to collect data starting at 1:00 PM on April 4, 2012:

```
ha# hb_report -f "2012/04/04 13:00" /tmp/hb_report.$(date+%Y%m%d-%H%M)
```

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

Collect SGI Service-Specific Information

Collect service-specific information. For example, run `dmcollect` for a resource group that contains DMF and `cxfsdump` for a resource group that contains CXFS. See the `dmcollect`(8) and `cxfsdump`(8) man pages for more information.

Generate a Kernel Crash Dump

Note: This procedure assumes that the appropriate `kernel-default-debuginfo` RPM is installed and that any CXFS node in the HA cluster has a fail policy that does not include `reset`.

To generate a kernel crash dump, do the following:

1. If the `totem token` value is not long enough to allow a dump to take place (which is usually the case), avoid system resets by disabling STONITH:

   ```
   ha# crm configure property stonith-enabled=false
   ```

   Note: Be aware of the following:

   - If you disable STONITH, the resources may not fail over properly and the cluster may enter an unpredictable state.
   - With a `totem token` value that is long enough for a dump to take place, it is possible that the system may restart before the failover system recognizes the situation, which can be problematic.

   ---

2. Force a crash dump to occur. For example:

   ```
   # echo 1 > /proc/sysrq-trigger
   ```

3. If you disabled STONITH in step 1, reenable it:

   ```
   ha# crm configure property stonith-enabled=true
   ```

4. Package up the kernel crash dump:

   ```
   ha# /usr/sbin/sgi_collect_dump
   ```

Use SGI Knowledgebase

If you encounter problems and have an SGI support contract, you can use the SGI Knowledgebase tool to search for information:

1. Log in to Supportfolio Online:

   https://support.sgi.com/login

2. Click on **Search the SGI Knowledgebase**.
3. Select the type of search you want to perform.

If you need further assistance, contact SGI Support.

Failover Testing Strategies

Before performing any sort of failover testing, do the following so that you can predict the expected results and examine the actual results:

1. Verify the state of the HA cluster:

   a. Check the resource fail counts:

   ```
   ha# crm resource failcount resourcePRIMITIVE show nodename
   ```

   Repeat this for each resource primitive on each node.

   b. If there has been a failure, ensure that the issue that caused a resource failure has been resolved.

   c. Reset the resource fail counts on the required nodes:

   ```
   ha# crm resource failcount resourcePRIMITIVE delete nodename
   ```

   d. Clear any implicit location constraints that may have been created by a previous administrative `move` command:

   ```
   ha# crm resource unmove resourceGROUP
   ```

2. Clearly delineate the start of each test in the logs by using the `logger`(1) command. For example:

   ```
   ha# logger "TEST START - testdescription".
   ```

Note: The HBA tests presume that the system has redundant Fibre Channel HBA paths to storage.

Table 11-1 describes failover testing strategies.

Table 11-1 Failover Tests

Test Type	Action	Expected Result
Administrative failover	Move the individual resource or resource group to the failover node: `ha#` **`crm resource move`** ***resourcePRIMITIVE failover_node*** or: `ha#` **`crm resource move`** ***resourceGROUP failover_node***	The individual resource or the resource group moves to the failover node. Occasionally, filesystems may fail to dismount cleanly or in a timely fashion, thus preventing an administrative move from occurring cleanly. In this case, the active node will likely be reset when a stop operation passes its timeout limit. **Note:** Remember that longer resource stop operation timeouts may result in longer failover times, and shorter resource stop operation timeouts may result in more frequent system reset events.
System reboot	Reboot the active node: `active#` **`reboot`**	All resources running on the rebooted server should move to the failover node
Simulated system crash	Reset the active node	All resources running on the node that was reset should move to the failover node
Simulated NFS daemon failure	Stop the NFS server: `ha#` **`service nfsserver stop`**	The resources should move to the failover node due to a monitor operation failure for the `nfsserver` resource
Simulated filesystem failure	Unmount the filesystem: `ha#` **`umount`** ***filesystem***	The resources should move to the failover node due to a monitor operation failure for the `Filesystem` resource

Test Type	Action	Expected Result
Single simulated HBA failure	Disable the port for the Fibre Channel HBA. For example: `brocade>` **`portdisable`** ***portnumber***	A device failover will not actually occur until I/O is attempted via the failed HBA path. An XVM failover to an alternate path should occur after I/O is performed on the system. **Note:** Remember to reenable the port after the test. For example: `brocade>` **`portenable`** ***portnumber***
Multiple simulated HBA failures	Disable the port for the Fibre Channel HBA. For example: `brocade>` **`portdisable`** ***portnumber*** Repeat for every HBA port on the system.	The server should be reset after I/O is performed on the system. There will likely be multiple monitor operation failures for various resources followed by a stop operation failure, which will result in a system reset and a forced XVM failover. **Note:** Remember to reenable the port after the test. For example: `brocade>` **`portenable`** ***portnumber***

Corrective Actions

The following are corrective actions:

- "Recovering from an Incomplete Failover" on page 205
- "Recovering from a CIB Corruption" on page 206
- "Clearing the Failcounts After a Severe Error" on page 206

Recovering from an Incomplete Failover

After an incomplete failover, in which one or more of the resource primitives are not started and the cluster can no longer provide high availability, you must do the following to restore functionality and high availability:

1. Put the cluster into maintenance mode:

   ```
   ha# crm configure property maintenance-mode=true
   ```

2. Determine which resource primitives have failcounts:

   ```
   ha# crm resource failcount resourcePRIMITIVE show node
   ```

   Repeat for each resource primitive on each node.

3. Troubleshoot the failed resource operations. Examine the `/var/log/messages` system log and application logs around the time of the operation failures in order to deduce why they failed. Then deal with those causes.

4. Ensure that all of the individual resources are working properly according to the information in:

   - Chapter 6, "CXFS NFS Edge-Serving HA Service" on page 53
   - Chapter 7, "DMF HA Service" on page 75
   - Chapter 8, "COPAN MAID HA Service for Mover Nodes" on page 149

5. Remove the failcounts found in step 2:

   ```
   ha# crm resource failcount failed_resourcePRIMITIVE delete node
   ```

   Repeat this for each failed resource primitive on each node.

6. Remove error messages:

   ```
   ha# crm resource cleanup failed_resourcePRIMITIVE node
   ```

   Repeat this for each failed resource primitive on each node.

7. Return the cluster to managed status, ether the following:

   ```
   ha# crm configure property maintenance-mode=false
   ```

Recovering from a CIB Corruption

Note: This procedure assumes that you have a good backup copy of the CIB that contains only static configuration information, as directed in "Backing Up the CIB" on page 174.

Do the following to recover from a CIB corruption:

1. Erase the existing corrupt CIB:

   ```
   ha# crm configure erase
   ```

2. Load a new CIB from the backup copy:

   ```
   ha# crm configure load replace CIB.xxxxxx
   ```

   For example, for the copy made on August 24 (`CIB.20100824-130236`):

   ```
   ha# crm configure load replace CIB.20100824-130236
   ```

For more information, see the SUSE *High Availability Guide* and the `cibadmin`(8) man page.

Clearing the Failcounts After a Severe Error

Under certain circumstances, a severe failure will cause the failcount for the resource primitives to be set to `INFINITY`. This means that the resource primitives cannot run on a specific node again until the failcount is cleared, which requires administrative action. See "Clearing the Resource Primitive Failcount" on page 175.

Differences Among FailSafe®, Heartbeat, and Pacemaker/Corosync

Table A-1 summarizes the differences among the following, for those readers who may be familiar with the older products:

- FailSafe®
- Linux-HA Heartbeat
- Pacemaker and Corosync

Note: These products do not work together and cannot form an HA cluster.

Table A-1 Differences Among HA Products

Topic	FailSafe	Heartbeat	Pacemaker & Corosync
Operating system	IRIX	Can be built and run on most operating systems based on UNIX. The version of Heartbeat packaged by SGI is part of the ISSP media distribution and runs on the base OS for ISSP as defined in the ISSP release notes.	SLES 11
Terminology	node resource	node resource	node resource
Size of cluster	8 nodes	8+ nodes (Specific resource agents may have cluster size limitations. DMF can run on only 2 nodes in active/passive mode.)	16 nodes in active/passive mode for DMF, but 2 nodes recommended. 2 nodes for CXFS NFS edge-serving in active/active mode. 2 parallel data mover nodes for COPAN OpenVault client in active/active mode.

Topic	FailSafe	Heartbeat	Pacemaker & Corosync
Node/member name	Hostname or private network address	Hostname and private network address	Hostname and private network address
NFS lock failover	Supported	Not supported by the operating system	Supported in active/passive configurations
Network tiebreaker	A node that is participating in the cluster membership. FailSafe tries to include the tiebreaker node in the membership in case of a split cluster.	You can configure Heartbeat to use a variety of methods to provide tiebreaker functionality.	You can configure HA to use a variety of methods to provide tiebreaker functionality.
Rolling upgrade	Supported	Supported	Supported
Configuration information storage	Information is stored in the cluster database. The cluster database is replicated on all nodes automatically and kept in synchronization.	The `/etc/ha.d/ha.cf` file contains bootstrap information and must be manually replicated across the cluster when changed. Other cluster configuration is stored in the cluster information base (CIB), which is a replicated database. You can use `cibadmin`(8) to query and update the CIB.	The `/etc/corosync/corosync.conf` file contains bootstrap information. Other cluster configuration is stored in the replicated CIB. You can use `cibadmin`(8) to query and update the CIB.
Making changes while the service is enabled	Depends upon the plug-in and the configuration device parameter.	Service parameters can be changed while a service is running. Depending on the service and parameter, a change may cause a `stop`/`start` or a trigger a `restart` action. SGI recommends that you do not make any changes that could stop or restart DMF and CXFS.	Service parameters can be changed while a service is running. Depending on the service and parameter, a change may cause a `stop`/`start` or a trigger a `restart` action. SGI recommends that you do not make any changes that could stop or restart DMF and CXFS.

Topic	FailSafe	Heartbeat	Pacemaker & Corosync
Heartbeat interval and timeout	You can specify cluster membership heartbeat interval and timeout (in milliseconds).	Provides a number of parameters to tune node status monitoring and failure actions.	Provides a number of parameters to tune node status monitoring and failure actions.
Heartbeat networks	Allows multiple networks to be designated as heartbeat networks. You can choose a list of networks.	You can configure Heartbeat to communicate over one or more private or public networks.	You can configure HA to communicate over one or more private or public networks.
Action scripts	Separate scripts named `start`, `stop`, `monitor`, `restart`, `exclusive`.	Open Cluster Framework (OCF) resource agent specification, which may support `start`, `monitor`, `stop`, and `restart` actions as well as other more-specialized actions.	OCF resource agent specification, which may support `start`, `monitor`, `stop`, and `restart` actions as well as other more-specialized actions.
Resource timeouts	Timeouts can be specified for each action (`start`, `stop`, `monitor`, `restart`, `exclusive`) and for each resource type independently.	Timeouts and failover actions are highly configurable.	Timeouts and failover actions are highly configurable.
Resource dependencies	Resource and resource type dependencies are supported and can be modified by the user.	Provides great flexibility to configure resource dependencies.	Provides great flexibility to configure resource dependencies.
Failover policies	The ordered and round-robin failover policies are predefined. User-defined failover policies are supported.	Heartbeat provides great flexibility to configure resource failover policies.	HAE provides great flexibility to configure resource failover policies.

Glossary

This glossary lists terms and abbreviations used within this guide. For a more information, see the SUSE *High Availability Guide*:

http://www.suse.com/documentation/sle_ha/

active/active mode

An HA cluster in which multiple nodes are able to run disjoint sets of resources, with each node serving as a backup for another node's resources in case of node failure.

active/passive mode

An HA cluster in which all of the resources run on one node and one or more other nodes are the failover nodes in case the first node fails.

active node

The node on which resources are running.

BMC

Baseboard management controller, a system controller used in resetting x86_64 systems.

CIB

Cluster information base, used to define the HA cluster.

clone

A resource that is active on more than one node.

COPAN MAID

Power-efficient long-term data storage based on an enterprise massive array of idle disks (MAID) platform.

Corosync

The underlying HA architecture that provides core messaging and membership functionality

CXFS

Clustered XFS.

CXFS NFS edge-serving

A configuration in which CXFS client nodes can export data with NFS.

DCP

Drive control program.

DMF

Data Migration Facility, a hierarchical storage management system for SGI environments.

DMF Manager

A web-based tool you can use to deal with day-to-day DMF operational issues and focus on work flow.

edge-serving

See *CXFS NFS edge-serving*.

failover node

The node on which resources will run if the active node fails or if they are manually moved by the administrator. Also known as the *passive node* or *standby node*.

fencing

The method that guarantees a known cluster state when communication to a node fails or actions on a node fail. (This is *node-level fencing* , which differs from the concept of *I/O fencing* in CXFS.)

HA

High availability, in which resources fail over from one node to another without disrupting services for clients.

HA fail policy

A parameter defined in the CIB that determines what happens when a resource fails.

HA-managed filesystem

A filesystem that will be made highly available according to the instructions in this guide.

HA service

The set of resources and resource groups that can fail over from one node to another in an HA cluster. The HA service is usually associated with an IP address.

High Availability Extension (HAE)

SUSE Linux Enterprise product for high availability.

IPMI

Intelligent Platform Management Interface, a system reset method for x86_64 systems.

ISSP

InfiniteStorage Software Platform, an SGI software distribution.

LCP

library control program.

LSB

Linux Standard Base.

node1

In the examples in this guide, the initial host (which will later become a node in the HA cluster) on which all of the filesystems will be mounted and on which all tape drives and libraries are accessible. See also *alternate node*.

NSM

Network Status Monitor.

node2

In the examples in this guide, the alternate host in the HA cluster other than the first node (node1). See also *node1*.

OCF

Open Cluster Framework.

OpenAIS

The underlying HA product that provides an HA API for certain applications, using the `openais` service

OpenVault

A tape mounting service used by DMF.

owner node

The node that DMF will use to perform migrations and recalls on a given shelf. The node on which you run `ov_copan` becomes the owner node of that shelf. In an HA environment, ownership is transferred as part of HA failover.

Pacemaker

The underlying HA architecture that provides cluster resource management

physvol

XVM physical volume.

primitive

Used to define a resource in the CIB.

resource

An application that is managed by HA.

resource agent

The software that allows an application to be highly available without modifying the application itself.

resource group

A set of resources that are colocated on the same node and ordered to start and stop serially. The resources in a resource group will fail over together as a set.

resource stickiness

An HA concept that determines whether a resource should migrate to another node or stay on the node on which it is currently running.

serverdir directory

A directory dedicated to holding OpenVault's database and logs within a highly available filesystem in the DMF resource group.

SOAP

Simple Object Access Protocol

split cluster

A situation in which cluster membership divides into multiple clusters, each claiming ownership of the same filesystems, which can result in filesystem data corruption. Also known as *split-brain syndrome*.

standard service

An application before HA has been applied to it.

STONITH

Shoot the other node in the head, the facility that guarantees cluster state by fencing non-responsive or failing nodes.

TMF

Tape Management Facility, a tape mounting service used by DMF.

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.

WSDL

Web Service Definition Language

XVM

Volume manager for XFS filesystems (local XVM).

Index