sgi®

High Availability Guide for SGI® InfiniteStorage™

007–5617–011

New Features in this Guide

This revision includes the following changes:

- A new resource agent `dmcopytool` for the DMF copytool for Lustre. It is used to support a single instance of the `lhsmtool_dmf`(8) command:
  - "SGI Resource Agents and RPMs" on page 2
  - "DMF Copytool Requirements" on page 84
  - "Configure and Test Lustre Before Applying the DMF Copytool an HA Environment" on page 92
  - "Add the `dmcopytool` Resources *(Optional)*" on page 154
- Due to a change in the way that the `pacemaker` command-line tools work, the status command should now be the following in order to include the `inactive` argument so that all resources appear:

  ```
  # crm status inactive
  ```
- New section: "Avoid UID/GID Conflicts" on page 35
- Reorganized for better clarity: resource templates are discussed in the order in which they are to be added within the procedures in Chapters 5–7 as well as in the new Appendix A, "Resource Reference" on page 235.

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
008	February 2014 Revision to support ISSP 3.1
009	May 2014 Revision to support ISSP 3.2
010	November 2014 Revision to support ISSP 3.3
011	July 2015 Revision to support ISSP 3.4

Contents

Figures

Tables

About This Guide

This publication provides information about creating a high-availability (HA) cluster using SGI ® resource agents released with SGI InfiniteStorage Software Platform (ISSP) products. The resource agents are used with the Corosync and Pacemaker products, which are provided with the following:

- Red Hat® Enterprise Linux High Availability Add-On
- SUSE® Linux® Enterprise High Availability Extension

Scope of this Guide

This guide describes the use of SGI resource agents. It does not provide details about configuring an HA cluster; for those details, see the following:

- Pacemaker information for either RHEL or SLES clusters:

http://clusterlabs.org/doc/en-US/Pacemaker/1.1-plugin/html-single/Pacemaker_Explained/index.html

- RHEL only:
  - Clusterlabs quick start guide for RHEL:

    http://clusterlabs.org/quickstart-redhat.html

  - RHEL Add-On documentation:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/High_Availability_Add-On_Overview/index.html

> **Note:** The information about RGmanager does not apply to the ISSP implementation, which uses Pacemaker.

- SUSE only:

  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*
- *DMF 6 Filesystem Audit Guide*
- *OpenVault Administrator Guide for SGI InfiniteStorage*
- *SGI InfiniteStorage Software Platform* (ISSP) release note (`README.txt`)
- *TMF 6 Administrator 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

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.

The following prompts are used in examples to help clarify the machine on which the command is executed:

`ha#`	Any node that is or will be in the HA cluster
`node1#`	The first node that is or will be in the HA cluster
`node2#`	The second node that is or will be in the HA cluster
`mover#`	DMF parallel data-mover node
`mover1#`	The first DMF parallel data-mover node that is or will be in the HA cluster (analogous to `node1`)
`mover2#`	The second DMF parallel data-mover node that is or will be in the HA cluster (analogous to `node2`)

`dmfserver#`	DMF server
`cxfsclient#`	CXFS client-only node
`cxfsserver#`	CXFS server-capable administration node
`downnode#`	HA node that requires maintenance
`upnode#`	HA node that does not require maintenance
`nfsclient#`	NFS client outside of the HA cluster
`otherhost#`	Machine outside of the HA cluster

Reader Comments

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

You can contact SGI in either of the following ways:

- Send e-mail to the following address:

  techpubs@sgi.com

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

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

SGI values your comments and will respond to them promptly.

Chapter 1

Introduction

This chapter discusses the following:

- "High Availability Overview" on page 1
- "SGI Resource Agents and RPMs" on page 2
- "Supported HA Services" on page 3
- "Outline of the Configuration Procedure" on page 11

High Availability Overview

For SGI InfiniteStorage Software Platform (ISSP) products, the Corosync and Pacemaker products provide the infrastructure to keep managed resources highly available. The high-availability (HA) software is designed to fail over resources from one node to another, so that they survive a single point of failure:

- A *resource* is an application that is managed by HA software according to its definition in the HA *cluster information base* (CIB) and the use of a resource agent
- A *resource agent* is software that allows an application to be highly available without modifying the application itself
- A *resource group* is a set of resources that must be managed and failed over as a set from one node to another
- A *clone* allows a resource to run simultaneously on multiple nodes
- An *HA service* is the aggregate functionality provided by individual resources, resource groups, and clone resources that are configured together, and is usually associated with an IP address

The HA software starts, monitors, and stops resources and entire HA services. The underlying *control services* differ by operating system:

- Red Hat Enterprise Linux (RHEL):
  - CMAN
  - Corosync
  - Pacemaker
- SUSE Linux Enterprise Server (SLES®):
  - OpenAIS
  - Corosync
  - Pacemaker

Each HA service is actively owned by one node. The HA software uses the IP address associated with each HA service to direct clients to the node currently running the service. 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.

SGI Resource Agents and RPMs

Table 1-1 lists the Open Cluster Framework (OCF) resource agents specific to SGI ISSP products and the associated templates that help you configure individual resources.

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

Resource Agent	Description
`copan_ov_client`	COPAN MAID OpenVault client HA service for parallel data-mover nodes and SGI DMF™ tiered-storage virtualization servers
`cxfs`	CXFS™ filesystems whose metadata server location must follow the location of another resource, such as DMF or NFS

Resource Agent	Description
`cxfs-client`	CXFS 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
`cxfs-client-nfsserver`	NFS server on a CXFS client-only node
`cxfs-client-smnotify`	Network Status Monitor (NSM) lock reclaim notification on a CXFS client-only node
`dmf`	DMF server
`dmcopytool`	DMF copytool for Lustre, used to support a single instance of the `lhsmtool_dmf` (8) command
`dmfman`	DMF Manager tool
`dmfsoap`	DMF client Simple Object Access Protocol (SOAP) service
`lxvm`	Local XVM volume manager
`openvault`	OpenVault mounting service for DMF
`tmf`	Tape Management Facility (TMF) mounting service for DMF

SGI provides templates for the SGI resource agents, plus additional community resources that may be of interest, in `/usr/share/doc/sgi-ha/templates`. See Appendix A, "Resource Reference" on page 235.

For information about software installation, see the ISSP release note and Chapter 4, "Create the Base HA Cluster" on page 35.

Supported HA Services

Although the SGI resource agents can be used independently, this guide provides example procedures to configure the following HA services:

- 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 a two-node active/active HA cluster (the nodes are two DMF parallel data-mover nodes). See "COPAN MAID OpenVault Client HA Service Failover " on page 8.

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

Note: The attributes and the value recommendations listed 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.

CXFS NFS Edge-Serving Failover

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

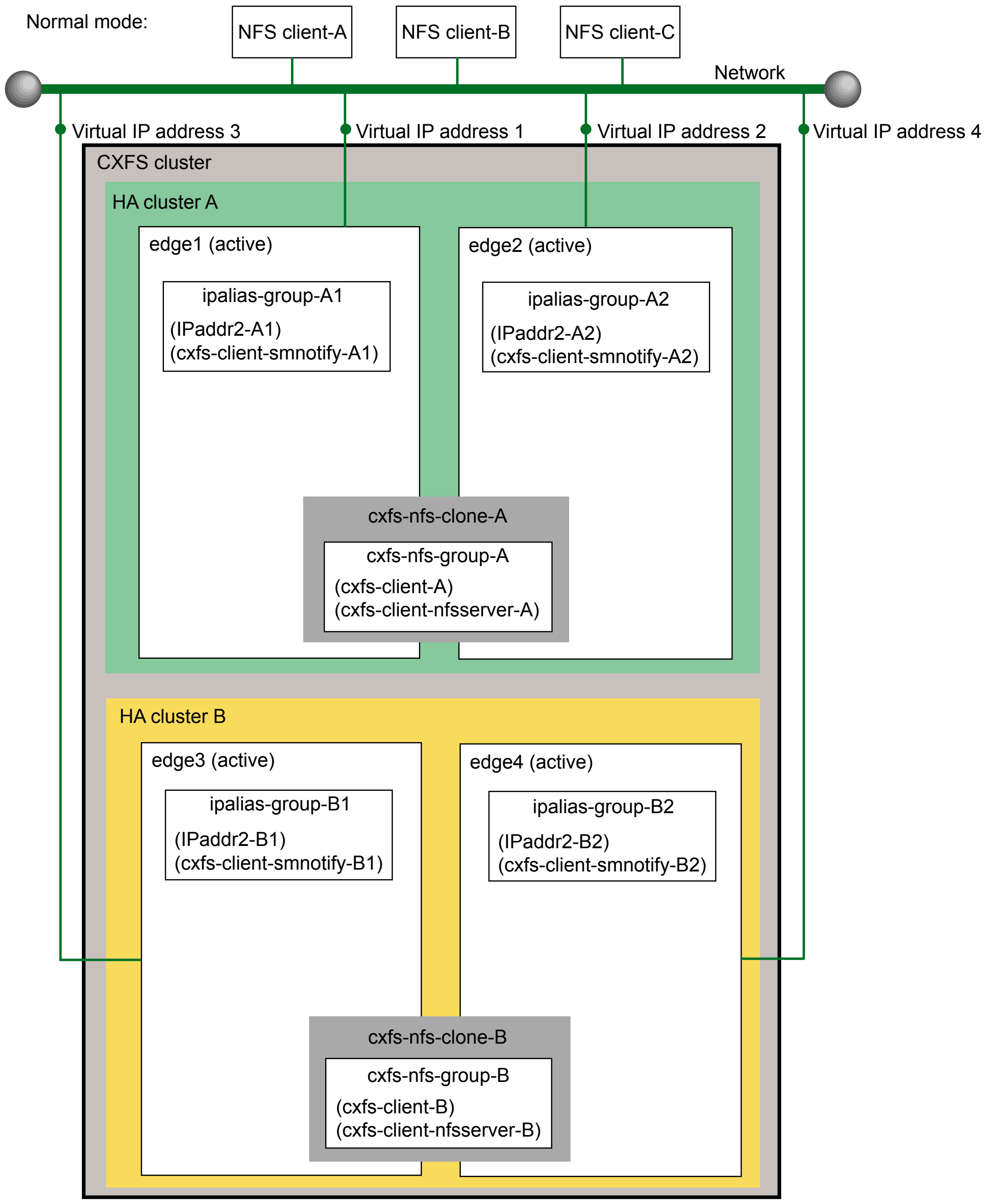

Figure 1-1 CXFS NFS Edge-Serving HA Service — Normal State

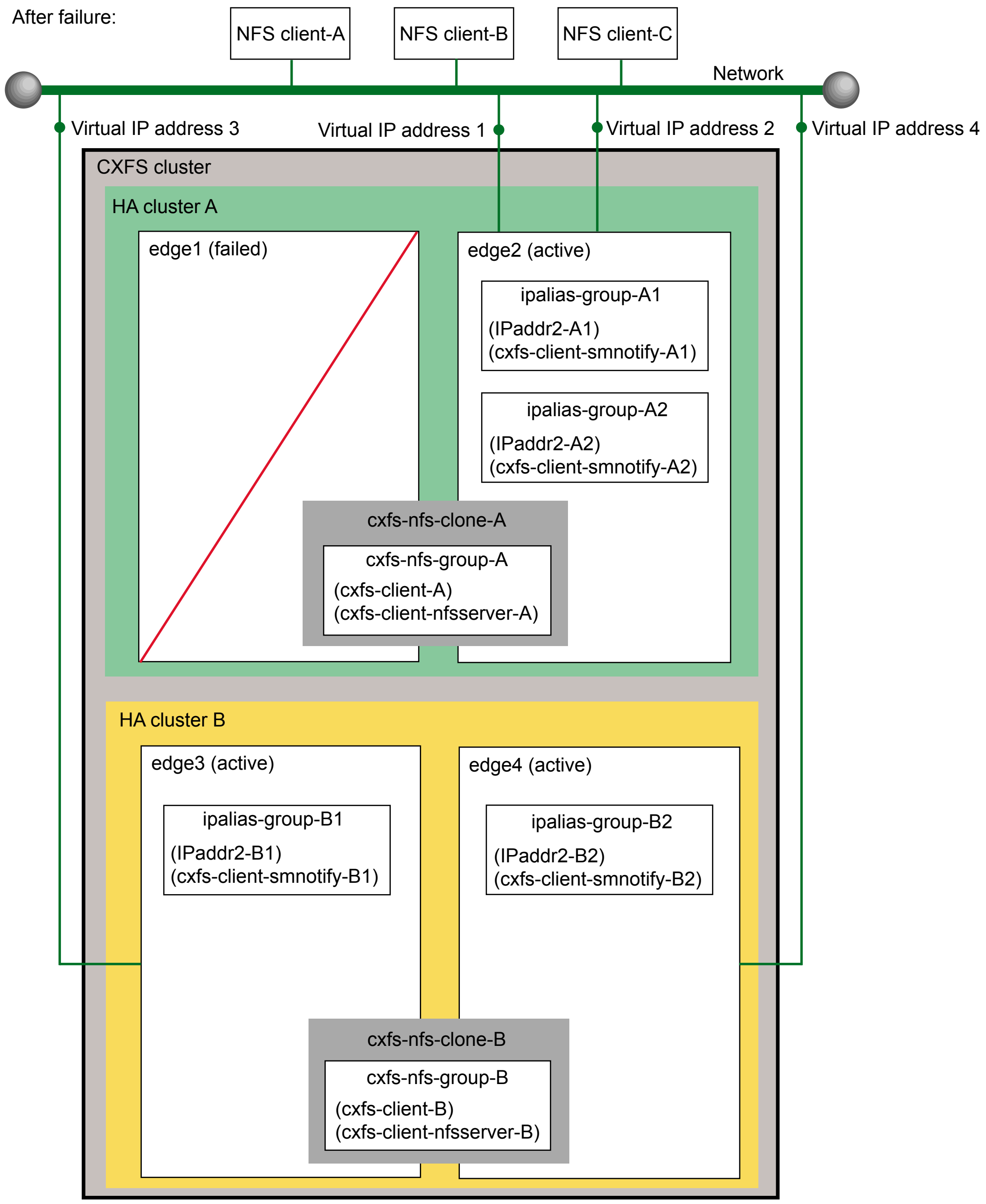

Figure 1-2 CXFS NFS Edge-Serving HA Service — After Failover

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.

Note: If you have multiple CXFS NFS edge-serving HA clusters within a single CXFS cluster, certain values must be unique to each HA cluster while others must be common across all of them. See "Requirements for a Second Edge-Serving HA Cluster" on page 72.

DMF Failover

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

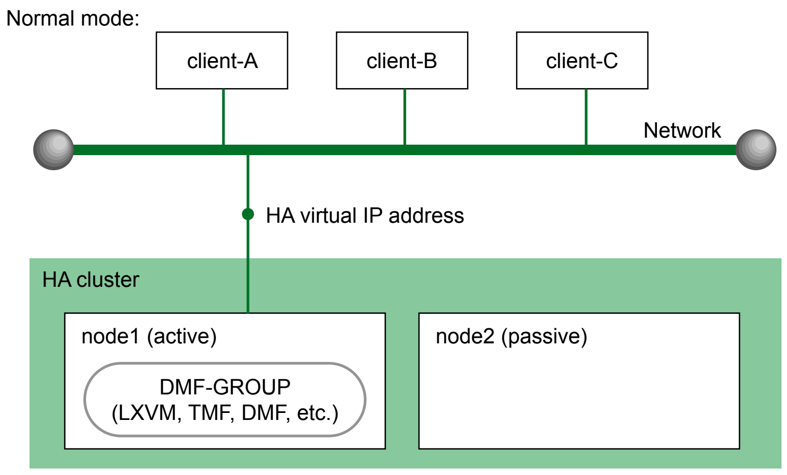

Figure 1-3 DMF HA Service — Normal State

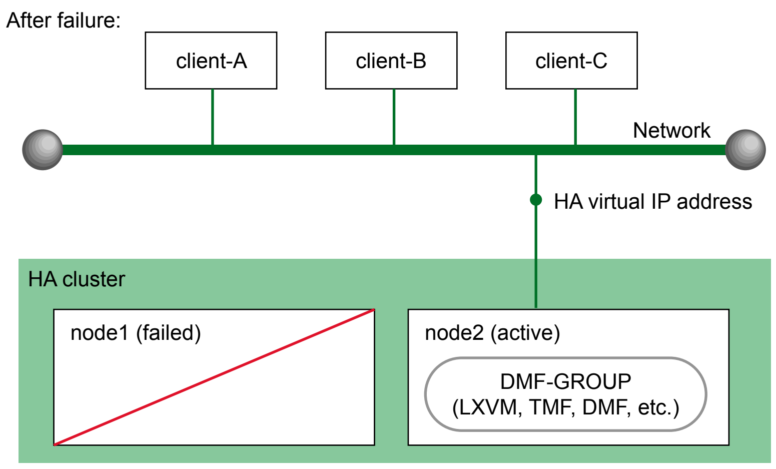

Figure 1-4 DMF HA Service — After Failover

COPAN MAID OpenVault Client HA Service Failover

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 active/active HA cluster (consisting of two parallel data-mover nodes).

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 `mover1` is the owner node of shelves 0 and 1, and `mover2` is the owner node of shelves 2 and 3).

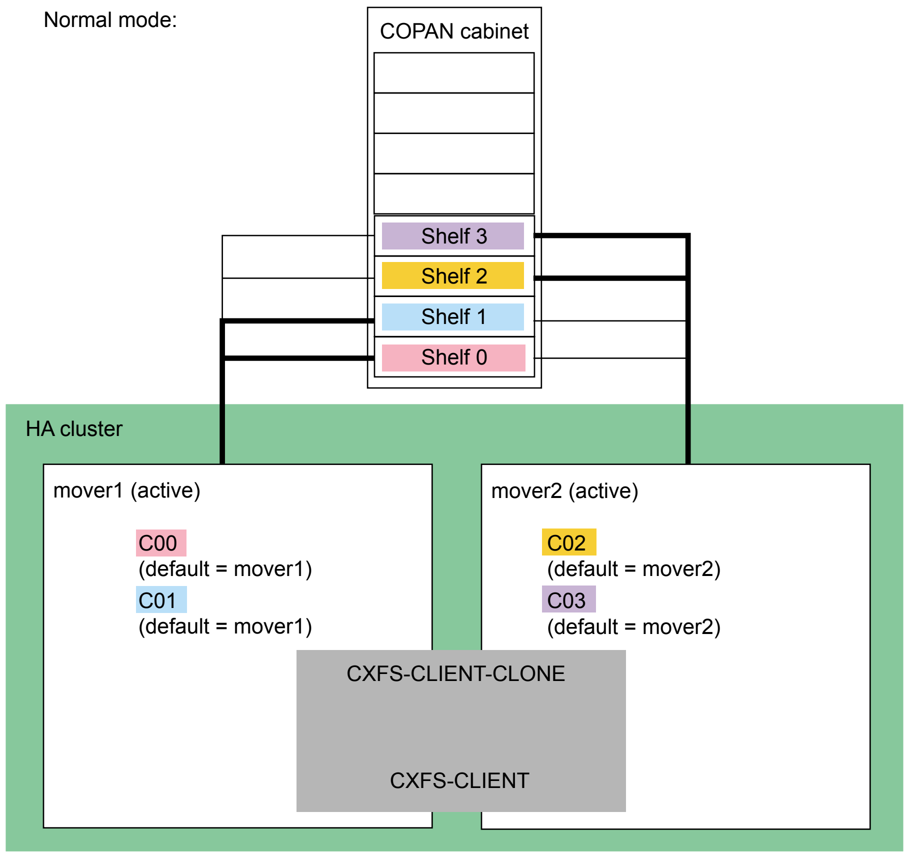

Figure 1-5 COPAN OpenVault Client HA Service for Mover Nodes — Normal State

When `mover1` fails, its COPAN OpenVault client resources move to `mover2` and `mover2` becomes the current owner node of all of the shelves, as shown in Figure 1-6.

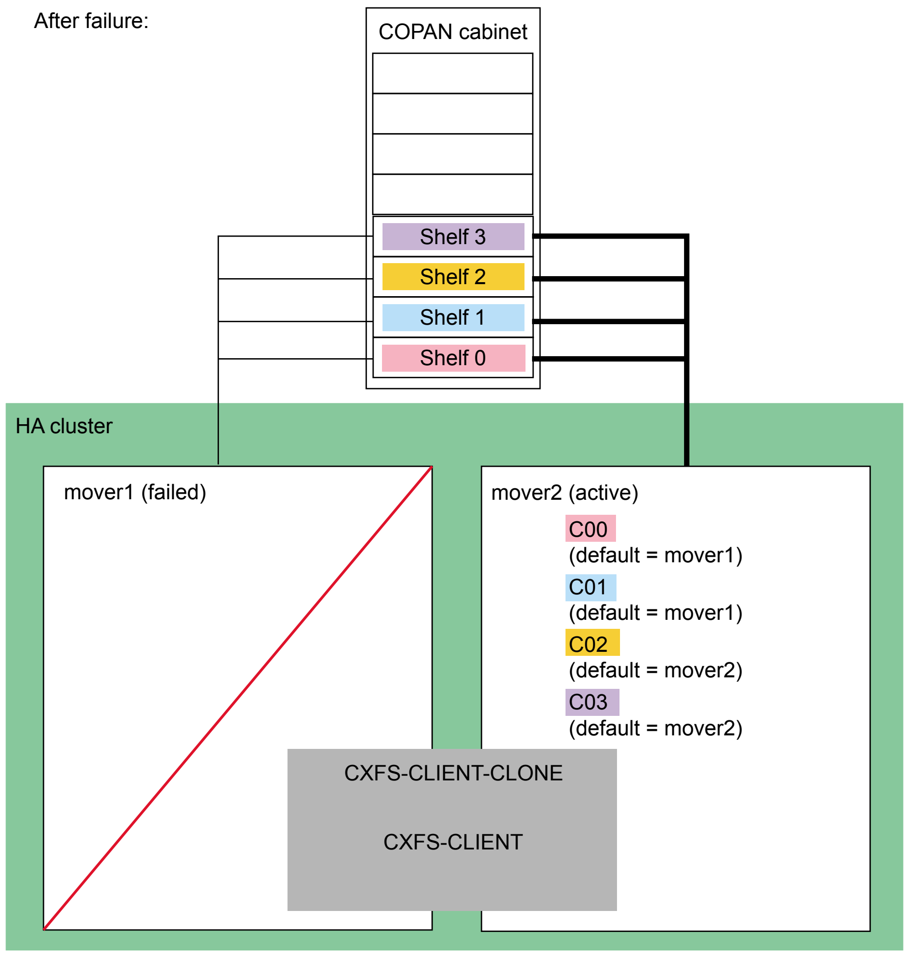

Figure 1-6 COPAN OpenVault Client HA Service for Mover Nodes — After Failover

After `mover1` recovers and rejoins the HA cluster, you can choose a convenient time to manually move `C00` and `C01` back to `mover1` to balance the load and return the HA cluster to its normal state (see "Manually Moving a `copan_ov_client` Resource" on page 195). 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).

Outline of the Configuration Procedure

Figure 1-7 summarizes the recommended steps to configure a two-node HA cluster for use with SGI InfiniteStorage products, pointing to the release notes or other chapters in this guide that provide the details:

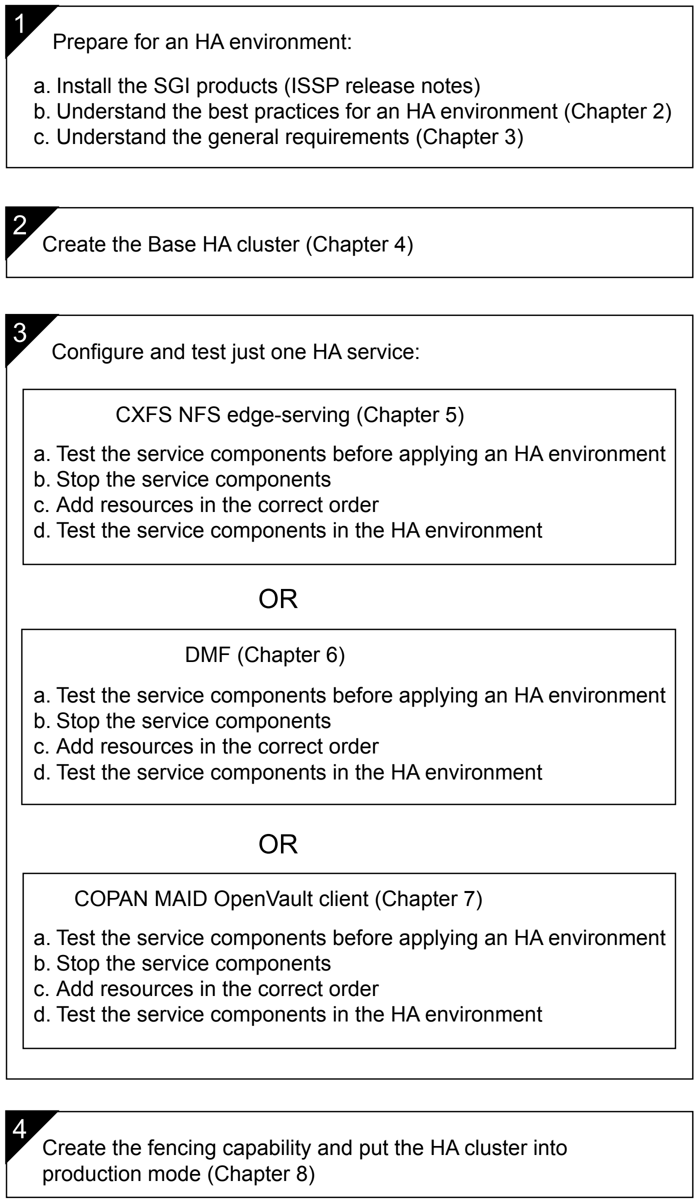
1
Prepare for an HA environment:
a. Install the SGI products (ISSP release notes)
b. Understand the best practices for an HA environment (Chapter 2)
c. Understand the general requirements (Chapter 3)
2
Create the Base HA cluster (Chapter 4)
3
Configure and test just one HA service:
CXFS NFS edge-serving (Chapter 5)
a. Test the service components before applying an HA environment
b. Stop the service components
c. Add resources in the correct order
d. Test the service components in the HA environment
OR
DMF (Chapter 6)
a. Test the service components before applying an HA environment
b. Stop the service components
c. Add resources in the correct order
d. Test the service components in the HA environment
OR
COPAN MAID OpenVault client (Chapter 7)
a. Test the service components before applying an HA environment
b. Stop the service components
c. Add resources in the correct order
d. Test the service components in the HA environment
4
Create the fencing capability and put the HA cluster into production mode (Chapter 8)

Figure 1-7 Map of the Configuration Procedure

Do the following:

1. Prepare for an HA environment:

   a. Ensure that you have installed the required SGI products on both nodes according to the installation procedure in the *SGI InfiniteStorage Software Platform Release Note*.

   b. Understand the recommendations for preparation, configuration, testing, and administration of an HA environment. See Chapter 2, "Best Practices" on page 15.

   c. Understand the general requirements for an HA environment. See Chapter 3, "General Requirements" on page 33.

2. Install the HA software and create the base HA cluster. See Chapter 4, "Create the Base HA Cluster" on page 35.

3. Configure and test each of the individual resources required for **just one** of the following HA services:

   - Chapter 5, "Create the CXFS NFS Edge-Serving HA Service" on page 47
   - Chapter 6, "Create the DMF HA Service" on page 75
   - Chapter 7, "Create the COPAN MAID OpenVault Client HA Service for Mover Nodes" on page 165

   Note: If you are implementing multiple HA services, you should complete the entire process for just one HA service before adding another HA cluster for another HA service. You must configure individual resources for a given HA service in the order shown in the procedure.

   In general, the procedure to create an HA service is as follows:

   a. Configure and test the service components **before applying the HA environment** (to reduce complexity)

   b. Stop the service components (so that they can be controlled by the HA software instead)

   c. Add the resources in the correct order, as documented

   d. Test the service components in the HA environment

4. Put the HA cluster into production mode by creating the STONITH (*shoot the other node in the head*) facility, reenabling node-level fencing, testing, and removing any resulting constraints. See Chapter 8, "Create the Fencing Capability and Put Into Production Mode" on page 185.

Chapter 2

Best Practices

The following are best practices when using SGI resource agents:

- "Best Practices Before Applying an HA Environment" on page 15
- "Best Practices for Configuring and Testing" on page 17
- "Best Practices for Administration" on page 26
- "Best Practices for Maintenance" on page 30

Best Practices Before Applying an HA Environment

The following are best practices for your environment before introducing HA:

- "Ensure the System is Ready" on page 15
- "Configure and Test the Components Before Applying an HA Environment" on page 16
- "Ensure that the Debug RPM Matches the Kernel" on page 16
- "Use a Separate Filesystem for CXFS NFS State Information for Edge Servers" on page 16
- "Use Consistent Virtual Hostnames" on page 16
- "Use the Correct CXFS Fail Policy in a DMF HA Service" on page 16

Ensure the System is Ready

Do the following:

- Fix networking issues first.
- Make your overall system configuration as simple as possible — complexity makes HA 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 Components Before Applying an HA Environment

Configure and test the services (like DMF) before making them highly available, which will make problems easier to diagnose.

In general, you should configure and test on one host (generally known in this guide as *edge1*, *node1*, or *mover1*). This host will later become a node in the HA cluster, on which all of the filesystems will be mounted and on which all drives and libraries are accessible.

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.

Use a Separate Filesystem for CXFS NFS State Information for Edge Servers

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-serving nodes 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.

Use the Correct CXFS Fail Policy in a DMF HA Service

In a DMF HA service, the STONITH facility must manage the server reset capability. Therefore, you must set the CXFS `failpolicy` for the CXFS server-capable administration nodes to `Fence,Shutdown` so that CXFS will not issue its own reset commands. For more information about CXFS `failpolicy` settings, see *CXFS 7 Administrator Guide for SGI InfiniteStorage*.

Best Practices for Configuring and Testing

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

- "Use the Resource Templates" on page 17
- "Use the Appropriate Tools" on page 22
- "Examine Log Files" on page 24
- "Avoid Unnecessary Failovers" on page 24
- "Always use STONITH" on page 25
- "Use One Group for a DMF HA Environment" on page 25
- "Examine the Use of Stickiness and Thresholds" on page 25
- "Additional Resources to Consider" on page 26

Use the Resource Templates

This section discusses the following:

- "Template Location" on page 18
- "Use the Templates as Building Blocks" on page 18
- "Resource Format" on page 18
- "Values You Can Change" on page 19
- "Values You Cannot Change" on page 19
- "Conventions for Resource Instance IDs" on page 19
- "Operations" on page 21
- "Timeout Values" on page 22
- "Meta Attributes" on page 22

Template Location

SGI provides templates of the resources required to configure an HA cluster in the following location:

```
/usr/share/doc/sgi-ha/templates/
```

Use the Templates as Building Blocks

You should use the templates as building blocks to construct your HA configuration file, adding one resource at a time and testing it before adding another resource.

You can create a copy of a template in a partial configuration file (referred to as the *workfile*). In general, you should use the values shown in the templates except for the italicized uppercase site-specific variables (most parameters and some timeouts are site-specific). In many cases, you can use the defaults provided for the variables. The comments in the templates provide information about these variables; you must remove all of these comments before loading the *workfile*. You will then run the following command to update the cluster information base (CIB) database:

```
node1# crm configure load update workfile
```

In general, use a new *workfile* for each resource or as directed in this guide.

Resource Format

In general, the individual resources take the following format:

```
# Comments_about_using_this_resource
#
primitive ID class:provider:type
          op monitor_operation_that_probes_to_see_if_the_resource_is_already_running
          op monitor_operation_that_probes_to_see_if_the_resource_continues_to_run
          op start_operation
          op stop_operation
          params parameters_specific_to_the_resource
          meta attributes_specific_to_the_resource
```

Caution: You must remove all of the comments before loading the work file.

Values You Can Change

Site-specific values that you may want to change are in uppercase in the template files. In this guide, they are highlighted in bold-italic uppercase (such as ***REPLACE_THIS***). In many cases, you can use the defaults provided; see the comments for more information.

Values You Cannot Change

The *class*, *provider*, and *type* shown in "Resource Format" on page 18 must use the exact values provided in the templates. Most *type* names are lowercase, but some are mixed case. The names are listed under `/usr/lib/ocf/resource.d/`*provider*.

Conventions for Resource Instance IDs

The suggested IDs for all resource `primitive` instances, `clone` instances, and `group` instances provided in the templates are site-changeable. You should choose IDs that are meaningful to your site; for simplicity, you may want to retain the defaults provided in the templates unless otherwise directed. By convention, the templates use the example instance IDs shown in Table 2-1.

All IDs should be unique within a given HA cluster; if there are multiple instances of the same *type*, each ID must be unique (such as two resources of type `IPaddr2` with individual instance IDs of `IPaddr2-A1` and `IPaddr2-A2`). Within this guide and in the templates, individual instance names are shown in uppercase by convention.

If you change an ID, you must ensure that the new name is used correctly elsewhere within constraints or groups as needed. You should use a unique ID for each resource, whether it is a `clone`, `group`, or `primitive` resource. Do not use spaces in resource IDs because this may cause HA software or other supporting software to behave in a confusing manner.

Table 2-1 Example Instance Names

Primitive / Group / Clone Type	Example Instance Name
`copan_ov_client`	*SHELF*, such as `C00` for the COPAN MAID cabinet 0, shelf 0 (the bottom shelf)
`cxfs`	`CXFS`
`cxfs-client`	`CXFS-CLIENT`
`clone`	`CXFS-CLIENT-CLONE`
`cxfs-client-nfsserver`	`EDGENFS-A` for cluster `A` and `EDGENFS-B` for cluster `B`
`cxfs-client-smnotify`	`CXFS-CLIENT-SMNOTIFY-`*XX*, such as `CXFS-CLIENT-SMNOTIFY-A1` and `CXFS-CLIENT-SMNOTIFY-A2` for cluster `A` and `CXFS-CLIENT-SMNOTIFY-B1` and `CXFS-CLIENT-SMNOTIFY-B2` for cluster `B`
`dmf`	`DMF`
`dmfman`	`DMFMAN`
`dmfsoap`	`DMFSOAP`
`group`	DMF: `DMF-GROUP` Edge-serving: `IPALIAS-GROUP-`*XX*, such as `IPALIAS-GROUP-A1` and `IPALIAS-GROUP-A2` for cluster `A` and `IPALIAS-GROUP-B1` and `IPALIAS-GROUP-B2` for cluster `B`
`fence-ipmilan`	`STONITH-`*NODENAME*, such as `STONITH-node1` (RHEL only)
`Filesystem`	*FILESYSTEM*, such as `dmfusr1`, `spool`, or `etc_samba`
`IPaddr2`	DMF: `IPaddr2` Edge—serving: `IPaddr2-A1` and `IPaddr2-A2` for cluster `A` and `IPaddr2-B1` and `IPaddr2-B2` for cluster `B`
`ipmi`	`STONITH-`*NODENAME*, such as `STONITH-node1` (SLES only)
`lxvm`	`LXVM`
`MailTo`	`NOTIFY`
`nfsserver`	`NFSSERVER`
`nmb`	`NMB`
`openvault`	`OV`

Primitive / Group / Clone Type	Example Instance Name
`ping`	`PING`
`smb`	`SMB`
`tmf`	`TMF`
`winbind`	`WINBIND`

Operations

Note the following:

- A `monitor` operation determines if the resource is operating correctly:
  - A *probe* `monitor` operation (which always uses an `interval` value of `0`) checks to see if the resource is already running.
  - A *standard* `monitor` operation periodically verifies that the resource continues to run. Each operation will time-out after the specified number of seconds. If the operation fails, it will attempt to restart the resource.

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

- The `start` operation initiates a 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 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).

- A `stop` operation terminates or gives up control of a resource. It will time-out after the specified time. If the operation fails, it will attempt to fence the node on which the failure occurred. The fail policy must be set to `fence` and a STONITH ("*shoot the other node in the head*") facility must be configured according to the requirements for your site. See:
  - RHEL: "`fence_ipmilan` Template" on page 276
  - SLES: "`ipmi` Template" on page 287

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

Timeout Values

The various `timeout` values provided in the templates are good starting points, but you should evaluate their use at your site.

Meta Attributes

Note the following:

- `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.

Use the Appropriate Tools

This section discusses the following:

- "Use the `crm`(8) Command for Configuration, Status, and Control" on page 23
- "Use the `crm_verify` Command to Verify Configuration" on page 24
- "Get More Information by Increasing Verbosity" on page 24

Use the `crm`(8) Command for Configuration, Status, and Control

Use the `crm`(8) command to configure, to obtain status, and to perform administrative actions. To invoke the command, do the following:

```
#  /usr/sbin/crm
```

For more details, enter `help` or `help` *subcommand*. For example:

```
# /usr/sbin/crm
crm(live)# help

This is crm shell, a Pacemaker command line interface.

Available commands:

        cib              manage shadow CIBs
        resource         resources management
        configure        CRM cluster configuration
        node             nodes management
        options          user preferences
        history          CRM cluster history
        site             Geo-cluster support
        ra               resource agents information center
        status           show cluster status
        help,?           show help (help topics for list of topics)
        end,cd,up        go back one level
        quit,bye,exit    exit the program
crm(live)# help cib

A shadow CIB is a regular cluster configuration which is kept in
a file. The CRM and the CRM tools may manage a shadow CIB in the
same way as the live CIB (i.e. the current cluster configuration).
A shadow CIB may be applied to the cluster in one step.
```

Note: Red Hat does not supply the `crm`(8) command. Therefore, the SGI ISSP HA software supplies this tool for RHEL customers. RHEL customers must use this version of `crm` for cluster resource configuration.

Use the `crm_verify` Command to Verify Configuration

Use the following command to verify changes you make to the CIB, after each resource primitive that you define:

```
ha# crm_verify -LV
```

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

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

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
```

Examine Log Files

You should regularly examine the following log files:

- RHEL:

  `/var/log/cluster/corosync.log` (primary RHEL HA log file)
  `/var/log/messages`

- SLES:

  `/var/log/messages`

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 standard `monitor` operation (see "Operations" on page 21). 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 an individual resource primitive except as directed in this guide.

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

Always use STONITH

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

- RHEL: "`fence_ipmilan` Template" on page 276
- SLES: "`ipmi` Template" on page 287

Use One Group for a DMF HA Environment

For a DMF HA cluster, place each resource `primitive` within one resource `group`. The resource `group` mechanism incorporates implied colocation constraints as well as resource order constraints. This 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 individual 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 following website:

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

Additional Resources to Consider

This section discusses the following:

- "`MailTo` Resource" on page 26
- "`ping` Resource" on page 26

`MailTo` Resource

You may want to consider defining a `MailTo` resource to implement notification when a given resource starts, stops, or fails over. For more information, see "`MailTo` Template" on page 293.

`ping` Resource

The `IPaddr2` virtual IP address resource agent monitors the existence of the IP address alias on the interface, but it does not monitor network interface controller (NIC) interface availability. You may want to consider defining a `ping` resource. For more information, see "`ping` Template" on page 304, and 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

Best Practices for Administration

The following are best practices for administering the HA cluster:

- "Make a Backup Copy of the CIB" on page 27
- "Monitor Status Information for Problems" on page 27
- "Clear Failcount Values After Resolving the Problem" on page 27
- "Remove Implicit Constraints after Explicitly Moving a Resource" on page 28
- "Set the Core Membership Timeout Value Appropriately" on page 28
- "Upgrade Appropriately" on page 29
- "Do Not Use an Edge-Serving Node as an NFS Client" on page 29
- "Include the PID in Core Files" on page 29

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 192
- "Recovering from a CIB Corruption" on page 233

Before making changes to an existing HA environment, ensure that you have a good backup copy of the current CIB. (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.

Monitor Status Information for Problems

Periodically watch the output of the following commands for problems:

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

For more information, see "Viewing the Cluster Status " on page 190.

Refer to the logs in case of error and periodically to ensure that you are aware of operations automatically initiated by HA software. See "Examine Log Files" on page 24.

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 Failcount" on page 192.

Remove Implicit Constraints after Explicitly Moving a Resource

If you want to move or start a resource (either an individual `primitive` or a `group`) on a specific node, enter the following:

```
ha# crm resource move resource 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 individual resource or resource group has successfully moved to the new node. Do the following:

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

To move the COPAN OpenVault client resource, see "Manually Moving a `copan_ov_client` Resource" on page 195.

Set the Core Membership Timeout Value Appropriately

Set the HA core membership timeout (`totem token`) 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). For more information, see *CXFS 7 Administrator Guide for SGI InfiniteStorage*.

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 `totem token` value of at least `90s`.

The default `totem consensus` value is 1.2 times the `totem token` value, which is appropriate for most sites. For example, for the `totem token` value of `90s`, the `totem consensus` value is set by default to `108s`.

See "Configure the Nodes for an HA Environment" on page 38.

Upgrade Appropriately

When upgrading the software, follow the procedure in "Performing a Rolling Upgrade" on page 197.

Do Not Use an Edge-Serving Node as an NFS Client

Do not use a CXFS NFS edge-serving node running HA software as an NFS client. (The NFS client service on a CXFS NFS edge-serving node does not support monitored locking.)

Include the 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
```

Best Practices for Maintenance

This section discusses the following:

- "Questions to Ask Before Performing Maintenance" on page 30
- "Hardware Maintenance" on page 30
- "System Software Updates" on page 31
- "ISSP Software Updates" on page 31
- "Changes Permitted on a Running Resource" on page 31
- "Changes that Require Maintenance Mode" on page 31
- "Changes that Require a Full Cluster Outage" on page 32

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 the HA software 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 213.

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 213.

In other cases, an upgrade with the operational HA cluster may be possible; see "Performing a Rolling Upgrade" on page 197.

ISSP Software Updates

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

https://support.sgi.com/login

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.*

⚠ **Caution:** Changing meta attributes or operation parameters will influence the behavior of the resource or clone and can therefore influence how the HA software 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 191.

Changes in this category include:

- Any change that requires DMF to be stopped according to the *DMF 6 Administrator 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 213.

Changes in this category include:

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

Chapter 3

General Requirements

This chapter discusses the following requirements for an HA cluster using SGI resource agents:

- "HA Support Requirements" on page 33
- "Licensing Requirements" on page 33
- "Software Version Requirements" on page 34
- "Hardware Requirements" on page 34
- "System Reset Requirements" on page 34
- "Time Synchronization Requirements" on page 34

Note: Resource-specific requirements are documented in the resource-specific chapters later in this guide.

HA Support Requirements

An HA environment may in some cases require the purchase of additional support from Red Hat or 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 an active/active cluster is not possible for DMF. 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*.

System Reset Requirements

You must use STONITH node-level fencing to protect data integrity in case of failure. The resource agent differs by operating system. See:

- RHEL: "`fence_ipmilan` Template" on page 276
- SLES: "`ipmi` Template" on page 287

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.

Create the Base HA Cluster

This chapter discusses the following:

- "Avoid UID/GID Conflicts" on page 35
- "Install the HA Software" on page 37
- "Enable Multicasting on the Ethernet Switch" on page 38
- "Configure the Nodes for an HA Environment" on page 38
- "Test the Base HA Cluster" on page 45
- "Create the ISSP HA Service" on page 46

Avoid UID/GID Conflicts

The `pacemaker` software automatically creates user `hacluster` and `haclient` if they do not already exist, using the following UID/GID values:

- RHEL: `189`
- SLES: `90`

Before installing the `pacemaker` software, verify that no other user or group is already using the default UID/GID values

- RHEL:
  - On `node1:`

    ```
    node1# getent passwd 189
    node1# getent group 189
    ```

  - On `node2:`

    ```
    node2# getent passwd 189
    node2# getent group 189
    ```

- SLES:

– On `node1`:

```
node1# getent passwd 90
node1# getent group 90
```

– On `node2`:

```
node2# getent passwd 90
node2# getent group 90
```

If there is no output, you can continue on to "Install the HA Software" on page 37.

However, if the output indicates that the UID/GID is already in use at your site, you must use `groupadd`(8) and `useradd`(8) commands to explicitly create the user/group with an unused UID/GID value. For example:

- RHEL: an example value of `190`:
  – On RHEL `node1`:

```
node1# groupadd -r haclient -g 190
node1# useradd -r -g haclient -u 190 -s /sbin/nologin -c "cluster user" hacluster
```

  – On RHEL `node2`:

```
node2# groupadd -r haclient -g 190
node2# useradd -r -g haclient -u 190 -s /sbin/nologin -c "cluster user" hacluster
```

- SLES: an example value of `91`:
  – On SLES `node1`:

```
node1# groupadd -o -r -g 91 haclient
node1# useradd -r -g haclient -c "heartbeat processes" -d /var/lib/heartbeat/cores/hacluster -o -u 91 hacluster
```

  – On SLES `node2`:

```
node2# groupadd -o -r -g 91 haclient
node2# useradd -r -g haclient -c "heartbeat processes" -d /var/lib/heartbeat/cores/hacluster -o -u 91 hacluster
```

Install the HA Software

The ISSP release note provided in the `noarch/sgi-isspdocs` RPM is installed in the following location:

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

Note: For simplicity, this procedure refers to the first node in the HA cluster as `node1` (which also implies either `edge1` or `mover1`) and the second node as `node2` (which also implies either `edge2` or `mover2`) .

Install the HA software on `node1` and `node2`, if not already done:

- RHEL:

  1. Install the following:

     ```
     rhel# yum groupinstall "High Availability"
     rhel# yum install pacemaker
     rhel# yum groupinstall "SGI ISSP High Availability"
     ```

  2. Create a RHEL update repository according to the directions in the ISSP release note.

- SLES:

  1. Install the SUSE Linux Enterprise High Availability Extension by following the instructions in the SUSE *High Availability Guide* section about installation as an add-on (section 3.3):

     https://www.suse.com/documentation/sle_ha/book_sleha/data/sec_ha_installation_add-on.html

     ---

     Note: You will only use the instructions to install the High Availability Extension add-on; further configuration will be done using the instructions in this ISSP guide.

     ---

  2. Install the ISSP HA pattern:

     ```
     sles# zypper install --type pattern SGI-ISSP-High-Availability
     ```

  3. Create a SLES update repository according to the directions in the ISSP release note.

Enable Multicasting on the Ethernet Switch

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

Configure the Nodes for an HA Environment

This section discusses the following:

- "RHEL: Configure the Nodes for an HA Environment" on page 39
- "SLES: Configure the Nodes for an HA Environment" on page 42

RHEL: Configure the Nodes for an HA Environment

For RHEL, do the following:

1. Do the following on `node1`:

   a. Choose the HA node names and networks that will be used for the primary and secondary heartbeat networks.

   For example, suppose you want to use the following:

   - `192.168.1.0` as the primary heartbeat network, with HA node names of `node1-ha` and `node2-ha`
   - `192.168.10.0` as the secondary heartbeat network, with alternate HA node names of `node1-ha-alt` and `node2-ha-alt` *(optional but recommended if appropriate)*

   You would then have the following in the `/etc/hosts` file:

```
10.0.1.1    node1
10.0.1.2    node2
192.168.1.1  node1-ha
192.168.1.2  node2-ha
192.168.10.1  node1-ha-alt
192.168.10.2  node2-ha-alt
```

   b. Create the cluster:

```
node1# ccs -f /etc/cluster/cluster.conf --createcluster mycluster
node1# ccs -f /etc/cluster/cluster.conf --addnode node1-ha
node1# ccs -f /etc/cluster/cluster.conf --addnode node2-ha
```

   c. Set the cluster heartbeat timeout:

```
node1# ccs -f /etc/cluster/cluster.conf --settotem token=120000
```

   See "Set the Core Membership Timeout Value Appropriately" on page 28.

   d. *(Optional)* Add an alternate HA heartbeat network, if appropriate:

```
node1# ccs -f /etc/cluster/cluster.conf --addalt node1-ha node1-ha-alt
node1# ccs -f /etc/cluster/cluster.conf --addalt node2-ha node2-ha-alt
```

   e. Set the debug logging level:

```
node1# ccs -f /etc/cluster/cluster.conf --setlogging debug="on"
```

f. Specify how cluster manager (CMAN) should send its fencing requests to Pacemaker:

```
node1# ccs -f /etc/cluster/cluster.conf --addfencedev pcmk agent=fence_pcmk
node1# ccs -f /etc/cluster/cluster.conf --addmethod pcmk-redirect node1-ha
node1# ccs -f /etc/cluster/cluster.conf --addmethod pcmk-redirect node2-ha
node1# ccs -f /etc/cluster/cluster.conf --addfenceinst pcmk node1-ha pcmk-redirect port=node1-ha
node1# ccs -f /etc/cluster/cluster.conf --addfenceinst pcmk node2-ha pcmk-redirect port=node2-ha
```

Note: Do this regardless of whether or not fencing is enabled within Pacemaker.

2. Copy `/etc/cluster/cluster.conf` to `node2`.
3. On both nodes, disable the default startup behavior that requires quorum:

Note: CMAN assumes the cluster should not start until the node has quorum.

a. On `node1`:

```
node1# echo "CMAN_QUORUM_TIMEOUT=0" >> /etc/sysconfig/cman
```

b. On `node2`:

```
node2# echo "CMAN_QUORUM_TIMEOUT=0" >> /etc/sysconfig/cman
```

4. Ensure that the `acpid` and `NetworkManager` services are stopped on both nodes:

a. On `node1`:

```
node1# chkconfig acpid off
node1# chkconfig NetworkManager off

node1# service acpid stop
node1# service NetworkManager stop
```

b. On `node2`:

```
node2# chkconfig acpid off
node2# chkconfig NetworkManager off

node2# service acpid stop
node2# service NetworkManager stop
```

5. On both nodes, verify that the user `hacluster` and group `haclient` have been added to the node. For example, showing the default UID/GID:

   a. On `node1`:

   ```
   node1# grep hacluster /etc/passwd
   hacluster:x:189:189:cluster user:/home/hacluster:/sbin/nologin
   node1# grep haclient /etc/group
   haclient:x:189:
   ```

   b. On `node2`:

   ```
   node2# grep hacluster /etc/passwd
   hacluster:x:189:189:cluster user:/home/hacluster:/sbin/nologin
   node2# grep haclient /etc/group
   haclient:x:189:
   ```

6. On both nodes, start the `cman` and `pacemaker` services:

   a. On `node1`:

   ```
   node1# service cman start
   node1# service pacemaker start
   ```

   b. On `node2`:

   ```
   node2# service cman start
   node2# service pacemaker start
   ```

Note: If a reboot occurs during the configuration process, you must manually restart the `cman` and `pacemaker` services. After configuration and testing are complete, you will enable automatic starting via `chkconfig` (in "RHEL: Enable `cman` and `pacemaker` to be Started Automatically" on page 188).

For more information, see:

http://clusterlabs.org/quickstart.html

SLES: Configure the Nodes for an HA Environment

Note: This procedure uses explicit node IDs, but you can choose to automatically generate node IDs if you prefer.

You must follow the detailed instructions in the SUSE *High Availability Guide* section about manual cluster setup using YaST in order to initialize the cluster and configure `node1` (section 3.5):

https://www.suse.com/documentation/sle_ha/book_sleha/data/sec_ha_installation_setup_manual.html

Using the above instructions, do the following for `node1`:

1. Define the following or the primary network channel:
   - **Bind Network Address**: set to the primary network that will support the cluster heartbeat (for example, the CXFS private network), which is different from the IP address to be failed over
   - **Multicast Address**: set to a multicast address
   - **Multicast Port**: set to a multicast port

   Note: You must ensure that at least the multicast address or multicast port is unique and is not shared between the primary and redundant channels, or shared with any other HA clusters on the same local network. As a best practice for clarity, SGI recommends that both the multicast port and multicast address are unique.

   Example primary network:

   ```
   Bind Network Address: 192.168.1.0
   Multicast Address:    226.94.1.1
   Multicast Port:       5405
   ```

2. Enable **Redundant Channel** (to the right of the first network channel configuration window)
3. Define the following for the redundant channel:
   - **Bind Network Address**: set to the redundant network that will support the cluster heartbeat and is different from the primary network and from IP address to be failed over

Caution: The primary and redundant channels must not share the same bind network address.

- **Multicast Address**: set to a unique multicast address
- **Multicast Port**: set to a unique multicast port

Example redundant network:

```
Bind Network Address: 192.168.10.0
Multicast Address:    226.94.1.2
Multicast Port:       5407
```

4. Set **rrp mode** to **active** (recommended).
5. Explicitly set the HA node ID, such as `1` for `node1`. Each node must have a unique HA node ID. (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.

6. Click **Next** to go to the next screen (**Security**).
7. Set security on.
8. Select **Generate Auth Key File**. It will be created in `/etc/corosync/authkey`; this process can take several minutes to complete.
9. Keep clicking **Next** on the subsequent screens without configuring anything until you exit the GUI.

Note: You will not complete all of the steps in the wizard. The following steps take the place of the remainder of the GUI.

10. Modify the HA authorization and configuration files:

a. Set the cluster heartbeat timeout by setting the `totem token` value in the `/etc/corosync/corosync.conf` file. The `consensus` value must either be missing (meaning it will default to 1.2 times the `token` value) or it must be set to a value at least 1.2 times the `token` value. For example:

```
...
totem {
 ...
 token:  60000
 consensus:  72000
        ...
}
```

See "Set the Core Membership Timeout Value Appropriately" on page 28 and the `corosync.conf`(5) man page.

b. Turn on debug messages for all subsystems other than the `totem`. Add or update the `logging` stanza of the `/etc/corosync/corosync.conf` file:

```
logging{
...
        debug:on
...
 logger_subsys {
                subsys: TOTEM
                debug:  off
        }
 }
```

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

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

d. Copy the `authkey` and `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
```

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

```
nodeid:  2
```

11. Enable the `logd` service to be started automatically at boot time and then start them immediately:

    • On `node1`:

    ```
    node1# chkconfig logd on
    node1# service logd start
    ```

    • On `node2`:

    ```
    node2# chkconfig logd on
    node2# service logd start
    ```

12. Start the `openais` service:

    • On `node1`:

    ```
    node1# service openais start
    ```

    • On `node2`:

    ```
    node2# service openais start
    ```

    Note: If a reboot occurs during the configuration process, you must manually restart the `openais` service. After configuration and testing are complete, you will enable automatic starting via `chkconfig` (in "Enable Cluster Services to be Started Automatically" on page 187).

Test the Base HA Cluster

Do the following to test the base HA cluster:

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

```
node1# crm status inactive
```

The output should show that the cluster consists of two nodes (in this case, `node1` and `node2`, and that there are no resources. For example (truncated):

```
node1# crm status inactive
...
2 Nodes configured, 2 expected votes
0 Resources configured.

Online: [ node1 node2 ]
```

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 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
   ```

Create the ISSP HA Service

After you have created and tested the base HA cluster, configure just one of the following ISSP HA services:

- Chapter 5, "Create the CXFS NFS Edge-Serving HA Service" on page 47
- Chapter 6, "Create the DMF HA Service" on page 75
- Chapter 7, "Create the COPAN MAID OpenVault Client HA Service for Mover Nodes" on page 165

After the HA service is created, you will follow the steps in Chapter 8, "Create the Fencing Capability and Put Into Production Mode" on page 185.

Note: If you are implementing multiple HA services, you should complete the entire process for one HA service before adding another service.

Chapter 5

Create the CXFS NFS Edge-Serving HA Service

This chapter provides an overview of the following:

- "Understand the Requirements for the Edge-Serving HA Service" on page 48
- "Configure and Test the Service Components Before Applying an HA Environment" on page 49
- "Ensure that the Base HA Cluster is Operational" on page 51
- "Stop Services Before Applying an HA Environment" on page 52
- "Ensure that the NFS Lock Services are Started *(RHEL only)*" on page 53
- "Copy the `/etc/exports` Entries" on page 53
- "Map of Resources for the Edge-Serving HA Service" on page 53
- "Create the Clone" on page 55
- "Add the `cxfs-client` Resource" on page 56
- "Add the `cxfs-client-nfsserver` Resource" on page 57
- "Test the Clone" on page 59
- "Create the IP Address Alias Group" on page 62
- "Test Each IP Address Alias Group" on page 67
- "Confirm the Completed Status" on page 71
- "Requirements for a Second Edge-Serving HA Cluster" on page 72
- "Put the HA Service into Production Mode" on page 73

Note: Details about the resources are provided in individual resource chapters.

Understand the Requirements for the Edge-Serving HA Service

This section discusses the following:

- "Edge-Serving Requirements" on page 48
- "IP Address Alias Requirements in an Edge-Serving HA Cluster" on page 49

Edge-Serving Requirements

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

- NFS version 3 (NFS v3) or NFS version 4 (NFS v4).
- An HA cluster of two CXFS client-only nodes. The nodes must run the CXFS edge-serving software.
- Ensure that the edge-serving node is not an NFS client. (The NFS client service on a CXFS NFS edge-serving node does not support monitored locking.)
- There must be a file (located on shared storage) that will be used for keeping 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 edge-serving nodes within a given HA cluster must share this file
  - If there are multiple HA clusters for edge-serving nodes within one CXFS cluster, all of the cluster systems must share this file

  The `statefile` 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 the NFS lock state. (In a non-HA cluster, this would be the `/var/lib/nfs/` directory.) The directory must be shared as follows:
  - All edge-serving nodes within a given HA cluster must share this directory
  - If there are multiple HA clusters for multiple CXFS NFS edge-serving HA services within one CXFS cluster, each must have a separate state directory

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

- *(RHEL only)* On all RHEL nodes, do the following:

  1. Set the following in the `/etc/sysconfig/nfs` file:

     ```
     START_SMNOTIFY="no"
     ```

  2. Ensure that NFS lock services are started at boot time:

     – On RHEL `edge1`:

     ```
     edge1# chkconfig nfslock on
     ```

     – On RHEL `edge2`:

     ```
     node2# chkconfig nfslock on
     ```

Note: A second edge-serving HA cluster within a given CXFS cluster requires that some values be unique to each HA cluster value, but some values be identical across the HA clusters, as discussed below in "Requirements for a Second Edge-Serving HA Cluster" on page 72.

IP Address Alias Requirements in an Edge-Serving HA Cluster

CXFS NFS edge-serving HA service requires at least one `IPaddr2` resource for each IPalias group (in an active-active configuration,there are at least two groups) in each HA cluster.

Configure and Test the Service Components Before Applying an HA Environment

Configure and test the service components **before applying an HA environment**, as described in the following:

- "Configure and Test the CXFS Client Before Applying the HA Environment" on page 50
- "Configure and Test the NFS Edge Service Before Applying an HA Environment" on page 50

Configure and Test the CXFS Client Before Applying the HA Environment

Before applying an HA environment, configure CXFS according to the instructions in *CXFS 7 Administrator Guide for SGI InfiniteStorage* and *CXFS 7 Client-Only Guide for SGI InfiniteStorage*.

Test CXFS by doing the following:

1. Ensure that the node is a member of the CXFS cluster and that the CXFS client service is running;

   ```
   edge1# service cxfs_client status
   ```

2. Ensure that filesystems are mounted. For example:

   ```
   edge1# df
   ```

Configure and Test the NFS Edge Service Before Applying an HA Environment

Before applying an HA environment, 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=`*uniquenumber* export option in order to prevent stale file handles after failover.

Test the CXFS NFS edge-serving before applying an HA environment by doing the following:

1. Ensure that the NFS service is running:

   - RHEL:

     ```
     rhel# service nfs status
     ```

     If it is not running, start it:

     ```
     rhel# service nfs start
     ```

   - SLES:

     ```
     sles# service nfsserver status
     ```

     If it is not running, start it:

     ```
     sles# service nfsserver start
     ```

2. Export the filesystem from `edge1`:

   a. Configure `/etc/exports`.

   b. Export the CXFS filesystems:

   ```
   edge1# exportfs -a
   ```

   c. Run the following command on `edge1` to verify that the filesystems are exported:

```
edge1# exportfs -v
/nfsexportedfilesystem   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=xxx)
```

3. NFS-mount and test the filesystem:

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

   - NFS v3:

     ```
     otherhost# mount -t nfs -o vers=3 edge1:/nfsexportedfilesystem /mnt/test
     ```

   - NFS v4:

     ```
     otherhost# mount -t nfs -o vers=4 edge1:/nfsexportedfilesystem /mnt/test
     ```

   b. 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 steps 2 and3, exporting and then mounting the filesystem from `edge2`.

Ensure that the Base HA Cluster is Operational

Ensure that the base HA cluster is operational, as described in "Test the Base HA Cluster" on page 45.

Stop Services Before Applying an HA Environment

Do the following on both nodes to ensure that the services will be controlled by the CXFS NFS edge-serving HA service:

Note: Do not disable the `cxfs` and `cxfs_cluster` services.

- RHEL system:

  RHEL `edge1`:

  ```
  edge1# chkconfig cxfs_client off
  edge1# chkconfig nfs off

  edge1# service cxfs_client stop
  edge1# service nfs stop
  ```

  RHEL `edge2`:

  ```
  edge2# chkconfig cxfs_client off
  edge2# chkconfig nfs off

  edge2# service cxfs_client stop
  edge2# service nfs stop
  ```

- SLES system:

  SLES `edge1`:

  ```
  edge1# chkconfig cxfs_client off
  edge1# chkconfig nfsserver off

  edge1# service cxfs_client stop
  edge1# service nfsserver stop
  ```

  SLES `edge2`:

  ```
  edge2# chkconfig cxfs_client off
  edge2# chkconfig nfsserver off

  edge2# service cxfs_client stop
  ```

```
edge2# service nfsserver stop
```

Ensure that the NFS Lock Services are Started *(RHEL only)*

On both RHEL nodes, ensure that NFS lock services are started at boot time:

- On RHEL `edge1`:

  ```
  edge1# chkconfig nfslock on
  ```

- On RHEL `edge2`:

  ```
  edge2# chkconfig nfslock on
  ```

Copy the `/etc/exports` Entries

Copy the `/etc/exports` entries that you would like to make highly available from `edge1` to the `/etc/exports` file on `edge2`.

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

Map of Resources for the Edge-Serving HA Service

Figure 5-1 on page 54 shows a map of an example configuration process for CXFS NFS edge-serving using two active/active HA clusters. Cluster `A` conists of two nodes (`edge1` and `edge2`); cluster `B` consists of two different nodes (`edge3` and `edge4`). Both HA clusters reside within a single CXFS cluster. This map also describes the start/stop order for resources.

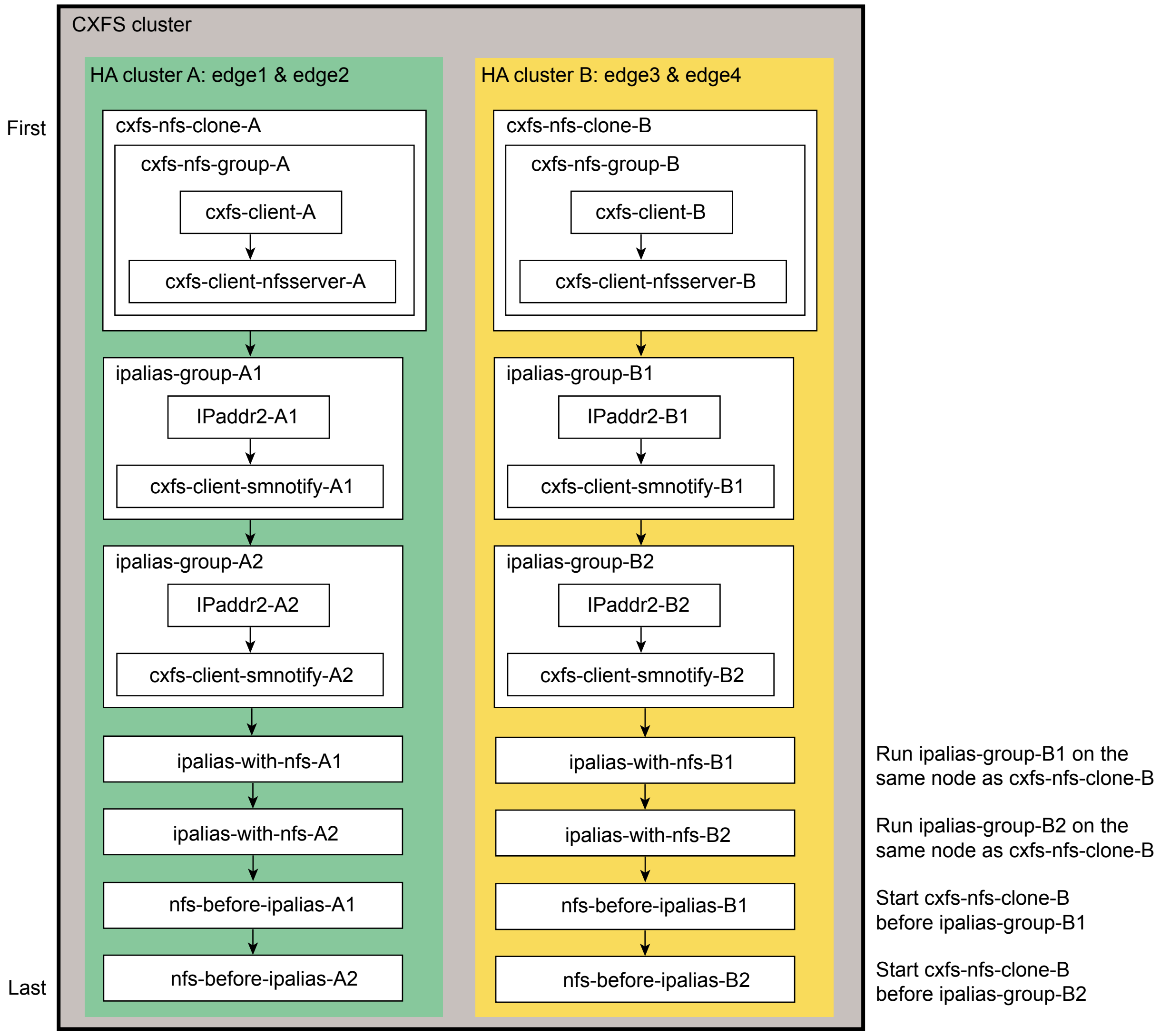

Figure 5-1 Map of Resources for two CXFS NFS Edge-Serving HA Services

Create the Clone

Use the templates in `/usr/share/doc/sgi-ha/templates` as building blocks. The instructions in this chapter assume that you use the instance names provided in the templates, such as `IPaddr2-A1` instance name for the `IPaddr2` resource type for cluster `A`, except as noted (see "Map of Resources for the Edge-Serving HA Service" on page 53 and "Conventions for Resource Instance IDs" on page 19).

Do the following:

1. Copy the contents of the `/usr/share/doc/sgi-ha/templates/cxfs-nfs-clone` template into a new partial configuration file (referred to as *workfile*).

   A `clone` resource named `CXFS-NFS-CLONE-A` for HA cluster `A` implements a CXFS NFS edge-serving HA service. It consists of a group (such as `CXFS-NFS-GROUP-A`) constructed of two resources (such as `CXFS-CLIENT-A` and `EDGENFS-A`).

   The template is located in:

   ```
   /usr/share/doc/sgi-ha/templates/cxfs-nfs-clone
   ```

   See "Use the Templates as Building Blocks" on page 18.

   Use the following:

   ```
   group CXFS-NFS-GROUP-X  CXFS-CLIENT-X  EDGENFS-X
   clone CXFS-NFS-CLONE-X  CXFS-NFS-GROUP-X \
      meta clone-max="2" target-role="Stopped" interleave="true"
   ```

Variable	Description
CXFS-NFS-GROUP-X	Name of this `group` instance, such as `CXFS-NFS-GROUP-A`
CXFS-CLIENT-X	Name of the `cxfs-client` resource instance, such as `CXFS-CLIENT-A`
EDGENFS-A	Name of the `cxfs-client-nfsserver` resource instance, such as `EDGENFS-A` for the first HA cluster

CXFS-NFS-CLONE-X	Name of this `clone` instance, such as `CXFS-NFS-CLONE-A`

For example:

```
group CXFS-NFS-GROUP-A CXFS-CLIENT-A EDGENFS-A
clone CXFS-NFS-CLONE-A CXFS-NFS-GROUP-A \
   meta clone-max="2" target-role="Stopped" interleave="true"
```

Add the `cxfs-client` Resource

Do the following:

1. Copy the `primitive` text from the `/usr/share/doc/sgi-ha/templates/cxfs-client` template into *workfile* and replace the site-specific variables as directed.

   A `cxfs-client` resource controls the CXFS client.

   The template is located in:

   `/usr/share/doc/sgi-ha/templates/cxfs-client`

   See "Use the Templates as Building Blocks" on page 18.

   Use the following:

```
primitive CXFS-CLIENT-A ocf:sgi:cxfs-client \
   op monitor interval="0" timeout="30s" \
   op monitor interval="120s" timeout="30s" on-fail="restart" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="600s" on-fail="fence" \
   params volnames="VOLUME-LIST"
```

Variable	Description
CXFS-CLIENT-A	Name of this resource instance, such as `CXFS-CLIENT-A`
VOLUME-LIST	Comma-separated list of the CXFS filesystems to be served via NFS, such as `cxfsvol1,cxfsvol2`

   For example, for cluster A:

```
primitive CXFS-CLIENT-A ocf:sgi:cxfs-client \
   op monitor interval="0" timeout="30s" \
   op monitor interval="120s" timeout="30s" on-fail="restart" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="600s" on-fail="fence" \
   params volnames="cxfsvol1,cxfsvol2"
```

Add the `cxfs-client-nfsserver` Resource

Do the following:

- Copy the `primitive` text from the `/usr/share/doc/sgi-ha/templates/cxfs-client-nfsserver` template into *workfile* and replace the site-specific variables as directed.

  A `cxfs-client-nfsserver` resource controls the NFS server running on a CXFS client. It is used in a CXFS NFS edge-serving HA service. It is part of a `clone` resource that runs on both nodes in the cluster.

  The template is located in:

  `/usr/share/doc/sgi-ha/templates/cxfs-client-nfsserver`

  See "Use the Templates as Building Blocks" on page 18.

  Use the following:

```
primitive EDGENFS-X ocf:sgi:cxfs-client-nfsserver \
   op monitor interval="0" timeout="60s" \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="300s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="600s" on-fail="restart" \
   params nfs_init_script="NFS-INIT-PATH" statedir="STATEDIR" \
          statefile="STATEFILE" volnames="VOLUME-LIST" \
          nfslock_init_script="NFSLOCK-INIT-PATH"
```

Variable	Description
EDGENFS-A	Name of this resource instance. For simplicity, give a unique name to every resource ID at your site; if you have a CXFS cluster containing two edge-serving clusters, you could use `EDGENFS-A` and `EDGENFS-B`.
NFS-INIT-PATH	Path to the NFS initialization script:

	– RHEL: `/etc/init.d/nfs` – SLES: `/etc/init.d/nfsserver`
NFSLOCK-INIT-PATH	Path to the `nfslock` initialization script: – RHEL: `/etc/init.d/nfslock` – SLES: (not used)
STATEDIR	Directory in the NFS filesystem that will be used to store NFS file-lock state for this HA cluster (equivalent to `/var/lib/nfs/` in a nonclustered configuration), such as: `/mnt/cxfsvol1/statd/nfs1-nfs2` **Note:** This value must be must be unique to this HA cluster.
STATEFILE	File in the NFS filesystem that will store NFS file-lock state for all of the NFS edge-serving HA clusters (equivalent to `/var/lib/nfs/state` in a nonclustered configuration), such as: `/mnt/cxfsvol1/statd/state` **Note:** This value must be identical to the *STATEFILE* value specified for `cxfs-client-smnotify`. All HA clusters within a CXFS cluster must have an identical *STATEFILE* value.
VOLUME-LIST	Comma-separated list of volume names containing CXFS filesystems that are to be served via NFS, such as: `cxfsvol1`

For example, for the first of two edge-serving HA clusters (cluster A) within a single CXFS cluster:

```
primitive EDGENFS-A ocf:sgi:cxfs-client-nfsserver \
   op monitor interval="0" timeout="60s" \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="300s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="600s" on-fail="restart" \
   params nfs_init_script="/etc/init.d/nfsserver" \
          statedir="/mnt/cxfsvol1/statd/nfs1-nfs2" \
          statefile="/mnt/cxfsvol1/statd/state" volnames="cxfsvol1"
```

- Verify that the `timeout` values are appropriate for your site.
- Verify that there are no comments in *workfile*.
- Save *workfile*.
- Update the database:

  ```
  edge1# crm configure load update workfile
  ```

Note: As a best practice, you should also run the following command to verify changes you make to the CIB:

```
edge1# crm_verify -LV
```

For simplicity, this step is not included in the following procedures but is recommended. For more information, see "Use the `crm_verify` Command to Verify Configuration" on page 24.

Test the Clone

Do the following to test the clone:

1. Start the `clone`. For example:

   ```
   edge1# crm resource start CXFS-NFS-CLONE-A
   ```

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

   a. View the status of the cluster on `edge1`. For example (truncated):

   ```
   edge1# crm status inactive
   ...
   2 Nodes configured, 2 expected votes
   4 Resources configured.

   Online: [ edge1 edge2 ]
   ```

```
Clone Set: CXFS-NFS-CLONE-A [CXFS-NFS-GROUP-A]
    Started: [ edge1 edge2 ]
```

b. Verify that the `cxfs_client` process is running on `edge1`:

```
edge1# 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 `edge2`.

c. View the status of the NFS daemons on `edge1`.

- RHEL:

  NFS v3 and NFS v4 (output truncated):

  ```
  edge1# service nfs status
  rpc.svcgssd is stopped
  rpc.mountd (pid 666 665 663 ... 649 647) is running...
  nfsd (pid 800 799 798 ...  672 671) is running...
  rpc.rquotad (pid 642) is running...
  ```

  Note: The `pid` numbers vary with each restart. For NFS v4, the `idmapd` services is also started (but is not reported in the output).

- SLES:

  NFS v3:

  ```
  edge1# service nfsserver status
  Checking for kernel based NFS server: mountd running
    statd running
    nfsd running
  ```

  NFS v4:

  ```
  edge1# service nfsserver status
  Checking for kernel based NFS server: idmapd running
    mountd running
    statd running
    nfsd running
  ```

Note: Although the `mountd` and `statd` daemons only apply to SLES NFS v3, they are started on SLES NFS v4 as well.

---

d. View the status of the NFS daemons as above but on `edge2`.

3. Set `edge2` to `standby` state to ensure that the resources remain on `edge1`:

```
edge1# crm node standby edge2
```

4. Confirm that `edge2` is offline and that the resources are off:

a. View the status of the cluster on `edge1`, which should show that `edge2` is in `standby` state:

```
edge1# crm status inactive
...
2 Nodes configured, 2 expected votes
4 Resources configured.

Node edge2: standby
Online: [ edge1 ]

  Clone Set: CXFS-NFS-CLONE-A [CXFS-NFS-GROUP-A]
      Started: [ edge1 ]
      Stopped: [ CXFS-NFS-GROUP-A:1 ]
```

b. Verify that the `cxfs_client` process is not running on `edge2` by executing the `ps`(1) command on `edge2` (there should be no output):

```
edge2# ps -ef | grep cxfs_client
edge2#
```

c. *(SLES only)* View the status of the NFS daemons on `edge2`, which should show for SLES that `statd` is `dead` and `nfsd` is `unused`:

- SLES NFS v3:

```
edge2# service nfsserver status
Checking for kernel based NFS server: mountd     unused
 statd                                           dead
 nfsd                                            unused
```

- SLES NFS v4:

```
edge2# service nfsserver status
Checking for kernel based NFS server: idmapd     running
 mountd                                          unused
 statd                                           dead
 nfsd                                            unused
```

Note: Although the `mountd` and `statd` daemons only apply only to SLES NFS v3, they are started on SLES NFS v4 as well.

5. Return `edge2` to online status:

```
edge1# crm node online edge2
```

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

Create the IP Address Alias Group

Do the following:

1. Create a `group` resource in another *workfile* for the first set of `IPaddr2` and `cxfs-client-smnotify` resources.

   A `group` resource named `IPALIAS-GROUP` controls the `IPaddr2` and `cxfs-client-smnotify` resources. It is used in a CXFS NFS edge-serving HA service. Each HA cluster will have a pair of `IPALIAS-GROUP` resources (such as `IPALIAS-GROUP-A1` and `IPALIAS-GROUP-A2` for HA cluster A), each with their own set of `colocation` and `order` constraints.

   Use the following:

```
group IPALIAS-GROUP-XX IPaddr2-XX CXFS-CLIENT-SMNOTIFY-XX \
   meta target-role="Stopped"
colocation IPALIAS-WITH-NFS-XX inf: IPALIAS-GROUP-XX CXFS-NFS-CLONE-X
order NFS-BEFORE-IPALIAS-XX inf: CXFS-NFS-CLONE-X IPALIAS-GROUP-XX
```

Variable	Description
IPALIAS-GROUP-XX	Name of this resource instance, such as `IPALIAS-GROUP-A1`

	for the first group in HA cluster `A`
IPaddr2-XX	Name of the `IPaddr2` resource instance, such as `IPaddr2-A1` for the first address in HA cluster `A`
CXFS-CLIENT-SMNOTIFY-XX	Name of the `cxfs-client-smnotify` resource instance, such as `CXFS-CLIENT-SMNOTIFY-A1`
IPALIAS-WITH-NFS-XX	Name of this `colocation` constraint, such as `IPALIAS-WITH-NFS-A1`
CXFS-NFS-CLONE-X	Name of the NFS `clone`, such as `CXFS-NFS-CLONE-A`
NFS-BEFORE-IPALIAS-XX	Name of this `order` constraint, such as `NFS-BEFORE-IPALIAS-A1`

For example, for the first group for cluster `A`:

```
group IPALIAS-GROUP-A1 IPaddr2-A1 CXFS-CLIENT-SMNOTIFY-A1 \
   meta target-role="Stopped"
colocation IPALIAS-WITH-NFS-1 inf: IPALIAS-GROUP-A1 CXFS-NFS-CLONE-A
order NFS-BEFORE-IPALIAS-A1 inf: CXFS-NFS-CLONE-A IPALIAS-GROUP-A1
```

The `colocation` and `order` constraints ensure that `IPALIAS-GROUP-A1` will only run on a server that is also running a `CXFS-NFS-CLONE` instance, and the `clone` instance will be started first.

2. Copy the `primitive` text from the `/usr/share/doc/sgi-ha/templates/IPaddr2` template into *workfile* and replace the site-specific variables as directed. Use a unique `primitive` ID, such as `IPaddr2-A1` for the first IP alias for HA cluster `A`.

   An `IPaddr2` resource controls the addition/deletion of an IP address alias on a network interface.

   Use the following:

```
primitive IPaddr2-XX ocf:heartbeat:IPaddr2 \
   op monitor interval="0" timeout="30s" \
   op start interval="0" timeout="90s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="100s" on-fail="fence" \
   params ip="IP-ADDRESS" \
   meta resource-stickiness="1" migration-threshold="1"
```

Variable	Description
IPaddr2-XX	Name of this resource instance, which must match the named used to define it in the associated `group` resource (for example, `IPaddr2-A1` for the first IP address in HA cluster A).
IP-ADDRESS	IP address of the virtual channel, such as `128.162.244.240`

For example, for the first IP address for HA cluster A:

```
primitive IPaddr2-A1 ocf:heartbeat:IPaddr2 \
   op monitor interval="0" timeout="30s" \
   op start interval="0" timeout="90s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="100s" on-fail="fence" \
   params ip="128.162.244.240" \
   meta resource-stickiness="1" migration-threshold="1"
```

3. Copy the `primitive` text from the `/usr/share/doc/sgi-ha/templates/cxfs-client-smnotify` template into *workfile* and replace the site-specific variables as directed. Use a unique `primitive` ID, such as `CXFS-CLIENT-SMNOTIFY-A1` for HA cluster A.

   A `cxfs-client-smnotify` resource controls NFS client notification of NFS server failovers and restarts. It is used in a CXFS NFS edge-serving HA service as part of a `group` resource that includes an `IPaddr2` resource.

   The templated is located in:

   `/usr/share/doc/sgi-ha/templates/cxfs-client-smnotify`

   Use the following:

```
primitive CXFS-CLIENT-SMNOTIFY-XX ocf:sgi:cxfs-client-smnotify \
   op monitor interval="0" timeout="60s" \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="30s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="30s" on-fail="fence" \
   params ipalias="IPALIAS-ADDRESS" statedir="STATEDIR" statefile="STATEFILE" \
          gracedir="GRACEDIR" hostname="IPALIAS-HOST" \
          seconds="120" volnames="VOLUME-LIST" \
   meta resource-stickiness="1" migration-threshold="1"
```

Variable	Description
CXFS-CLIENT-SMNOTIFY-XX	Name of this resource instance, such as `CXFS-CLIENT-SMNOTIFY-A1`
IPALIAS-ADDRESS	IP address of the 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 `128.162.244.244`
STATEDIR	Directory in the NFS filesystem that will be used to store NFS file-lock state for this NFS edge-serving HA cluster. **Note:** This value must be identical to the *STATEDIR* value specified for `cxfs-client-nfsserver`, and must be unique to this HA cluster.
STATEFILE	File in the NFS filesystem that will store NFS file-lock state for all of the NFS edge-serving HA clusters within one CXFS cluster (equivalent to `/var/lib/nfs/state` in a nonclustered configuration), such as: `/mnt/cxfsvol1/statd/state` **Note:** This value must be identical to the *STATEFILE* value specified for `cxfs-client-nfsserver`. All HA clusters within a CXFS cluster must have an identical *STATEFILE* value.
GRACEDIR	Directory on the NFS file-lock-state filesystem that specifies the directory containing the grace-period state (the grace period specified by the `seconds` attribute is the time during which the CXFS tokens

owned by a given CXFS client will be maintained by that client to permit time for failover), such as:

`/mnt/cxfsvol1/grace`

Note: All HA clusters within a CXFS cluster must have an identical *GRACEDIR* value.

IPALIAS-HOST	Hostname of *IPALIAS-ADDRESS*, which must match what is in `/etc/hosts` or be resolvable with DNS, such as `hostalias1` or `hostalias2`
VOLUME-LIST	Comma-separated list of all volumes that will be served via *IPALIAS-HOST*, such as: `cxfsvol1`

For example:

```
primitive CXFS-CLIENT-SMNOTIFY-A1 ocf:sgi:cxfs-client-smnotify \
   op monitor interval="0" timeout="60s" \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="30s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="30s" on-fail="fence" \
   params ipalias="128.162.244.244" statedir="/mnt/cxfsvol1/statd/nfs1-nfs2" \
          statefile="/mnt/cxfsvol1/statd/state" \
          gracedir="/mnt/cxfsvol1/grace" hostname="myhost" \
          seconds="120" volnames="cxfsvol1" \
   meta resource-stickiness="1" migration-threshold="1"
```

4. Save *workfile*.

5. Update the database:

```
edge1# crm configure load update workfile
```

6. Create a second `group` resource in another *workfile* for the second set of `IPaddr2` and `cxfs-client-smnotify` resources. For example, for the second group:

```
group IPALIAS-GROUP-A2 IPaddr2-A2 CXFS-CLIENT-SMNOTIFY-A2 \
        meta target-role="Stopped"

colocation IPALIAS-WITH-NFS-A2 inf: IPALIAS-GROUP-A2 CXFS-NFS-CLONE
order NFS-BEFORE-IPALIAS-A2 inf: CXFS-NFS-CLONE IPALIAS-GROUP-A2
```

7. Copy the `primitive` text from the `/usr/share/doc/sgi-ha/templates/IPaddr2` template into *workfile* and replace the site-specific variables as directed above. Use a unique `primitive` ID, such as `IPaddr2-A2`.

8. Copy the `primitive` text from the `/usr/share/doc/sgi-ha/templates/cxfs-client-smnotify` template into *workfile* and replace the site-specific variables as directed above. Use a unique `primitive` ID, such as `CXFS-CLIENT-SMNOTIFY-A2`.

9. Verify that the `timeout` values are appropriate for your site.

10. Verify that there are no comments in *workfile*.

11. Save *workfile*.

12. Update the database:

```
edge1# crm configure load update workfile
```

Test Each IP Address Alias Group

To test each IP address alias `group`, do the following:

1. Start the `group`. For example, to start `IPALIAS-GROUP-A1`:

```
edge1# crm resource start IPALIAS-GROUP-A1
```

2. Test the IP address alias resource within the `group`:

   a. Verify that the IP address is configured correctly on `edge1`:

```
edge1# 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 `edge2` does not accept the IP address packets. For example, run the following command on `edge2` (the output should be `0`):

```
edge2# 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 `edge1`:

```
nfsclient# ssh root@128.162.244.240
Last login: Mon Jul 14 10:34:58 2008 from mynode.mycompany.com
edge1# uname -n
edge1
```

d. Move the resource `group` containing the `IPaddr2` resource from `edge1` to `edge2`:

```
edge1# crm resource move IPALIAS-GROUP-A1 edge2
```

e. Verify the status:

```
edge1# crm status inactive
...
2 Nodes configured, 2 expected votes
8 Resources configured.

Online: [ edge1 edge2 ]

  Clone Set: CXFS-NFS-CLONE-A [CXFS-NFS-GROUP-A]
      Started: [ edge1 edge2 ]
  Resource Group: IPALIAS-GROUP-A1
      IPaddr2-A1    (ocf::heartbeat:IPaddr2):       Started edge2
      CXFS-CLIENT-SMNOTIFY-1      (ocf::sgi:cxfs-client-smnotify):        Started edge2
```

f. Verify that the IP address is configured correctly on `edge2`:

```
edge2# 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
```

g. Verify that `edge1` does not accept the IP address packets by running the following command on `edge1` (the output should be `0`):

```
edge1# ip -o addr show | grep -c 128.162.244.240
0
```

h. 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 `edge2`:

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

edge2# uname -n
edge2
```

i. Move the resource `group` containing the `IPaddr2` resource back to `edge1`:

```
edge1# crm resource move IPALIAS-GROUP-A1 edge1
```

j. Verify the status:

```
edge1# crm status inactive
...
2 Nodes configured, 2 expected votes
8 Resources configured.

Online: [ edge1 edge2 ]

  Clone Set: CXFS-NFS-CLONE-A [CXFS-NFS-GROUP-A]
      Started: [ edge1 edge2 ]
  Resource Group: IPALIAS-GROUP-A1
      IPaddr2-A1    (ocf::heartbeat:IPaddr2):       Started edge1
      CXFS-CLIENT-SMNOTIFY-A1   (ocf::sgi:cxfs-client-smnotify):    Started edge1
```

k. Test again as in steps a–c above.

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

```
edge1# crm resource unmove IPALIAS-GROUP-A1
```

3. Repeat steps 1 and 2 for the other `group`, such as `IPALIAS-GROUP-A2`.

4. Test the NSM notification resource within the `group`. For example, on a Linux NFS client:

   a. On a system that is outside the HA cluster (for example, a system named `nfsclient`), mount the filesystem via the IP address alias `hostname` values specified in the `cxfs-client-smnotify` resources (such as `hostalias1` and `hostalias2`, which are not the physical hostnames). For example:

```
nfsclient# mount hostalias1://mnt/nfsexportedfilesystem /hostalias1
nfsclient# mount hostalias2://mnt/nfsexportedfilesystem /hostalias2
```

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

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

c. Acquire locks:

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

d. *(NFSv3 only)* Check in the shared `sm-notify` `statedir` directory on the NFS server for resources `edge1` and `edge2` 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, ensure that the appropriate NFS file contains 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 node that uses the IP address specified by the `ipalias` value, such as `edge1`, by hostname and not by the IP address in order for this to work.) The file location varies by OS:

   • RHEL: `/var/lib/nfs/statd/sm`

   • SLES: `/var/lib/nfs/sm`

f. Put `edge1` into `standby` state:

```
edge1# crm node standby edge1
```

g. Verify that both of the IP address aliases are now on `edge2`:

```
edge2# ip addr
```

h. *(NFS v3 only)* Verify that the log files (see "Examine Log Files" on page 24) on the NFS client (`nfsclient`) contain a message about reclaiming locks for the hostname for every `ipalias` value 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` parameter. NSM reboot notification will be sent to clients for all IP address aliases in the cluster, so you will see messages for all IP address aliases that have been mounted by the client.) For example:

```
Jul 30 13:40:46 nfsclient kernel: NLM: done reclaiming locks for host edge2
Jul 30 13:40:49 nfsclient kernel: NLM: done reclaiming locks for host edge1
```

i. Make `edge1` active again:

```
edge1# crm node online edge1
```

5. Test the other `group`.

Confirm the Completed Status

Use the `status` command to confirm the resulting HA cluster:

```
edge1# crm status inactive
...
2 Nodes configured, 2 expected votes
10 Resources configured.

Online: [ edge1 edge2 ]

  Clone Set: CXFS-NFS-CLONE-A [CXFS-NFS-GROUP-A]
      Started: [ edge1 edge2 ]
  Resource Group: IPALIAS-GROUP-A1
      IPaddr2-A1     (ocf::heartbeat:IPaddr2):       Started edge1
      CXFS-CLIENT-SMNOTIFY-1     (ocf::sgi:cxfs-client-smnotify):     Started edge1
  Resource Group: IPALIAS-GROUP-A2
      IPaddr2-A2     (ocf::heartbeat:IPaddr2):       Started edge2
      CXFS-CLIENT-SMNOTIFY-2     (ocf::sgi:cxfs-client-smnotify):     Started edge2
  STONITH-edge1 (stonith:external/ipmi):        Started edge2
  STONITH-edge2 (stonith:external/ipmi):        Started edge1
```

Note: It does not matter whether `IPALIAS-GROUP-A1` runs on `edge1` or `edge2`. The important thing is that during normal operation (before failover), `IPALIAS-GROUP-A1` and `IPALIAS-GROUP-A2` run on different nodes

---

Requirements for a Second Edge-Serving HA Cluster

For a second edge-serving HA cluster, repeat the above procedures for the second HA cluster (cluster B) for `edge3` and `edge4`, using unique resources names.

Note the following requirements:

- **Identical** among all HA clusters within a single CXFS cluster:
  - *GRACEDIR* value for the `cxfs-client-smnotify` resource
  - *STATEFILE* value for the `cxfs-client-smnotify` and `cxfs-client-nfsserver` resources
- **Unique** to each HA cluster within a single CXFS cluster:
  - *STATEDIR* value for the `cxfs-client-nfsserver`, and `cxfs-client-smnotify` resources

The following figure illustrates which values are unique and which are shared.

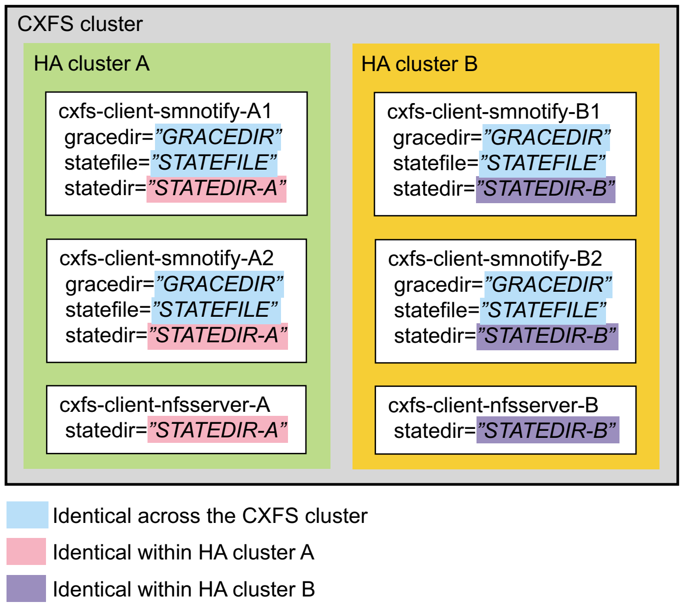

Figure 5-2 Two NFS Edge-Serving HA Clusters

Put the HA Service into Production Mode

See Chapter 8, "Create the Fencing Capability and Put Into Production Mode" on page 185 to complete the process.

Chapter 6

Create the DMF HA Service

This chapter provides an overview of the following:

- "Understand the Requirements for the DMF HA Service" on page 75
- "Configure and Test the Service Components Before Applying an HA Environment" on page 85
- "Ensure that the Base HA Cluster is Operational" on page 95
- "Stop Services Related to DMF Before Applying an HA Environment" on page 95
- "Add the Resources in the Correct Order" on page 97
- "Confirm the Completed Status" on page 164
- "Put the DMF HA Service into Production Mode" on page 164

Understand the Requirements for the DMF HA Service

Before applying an HA environment, you must understand the requirements of the individual components:

- "CXFS Requirements" on page 76
- "Local XVM Requirements" on page 78
- "Local Filesystem Requirements" on page 79
- "IP Address Alias Requirements" on page 79
- "TMF Requirements" on page 79
- "OpenVault Requirements" on page 80
- "COPAN MAID Requirements for a DMF HA Service *(Optional)*" on page 81
- "DMF Requirements" on page 81
- "Samba Requirements *(Optional)*" on page 84
- "DMF Copytool Requirements" on page 84

- "DMF Manager Requirements *(Optional)*" on page 84
- "DMF Client SOAP Service Requirements *(Optional)*" on page 85

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 76
- "CXFS Relocation Support" on page 77
- "Applications that Depend Upon CXFS Filesystems" on page 77
- "CXFS and System Reset" on page 77
- "CXFS Start/Stop Issues" on page 77
- "CXFS Volumes and Filesystems Managed by DMF" on page 78
- "`cxfs` and `cxfs_cluster` Services Not Controlled by the HA Environment" on page 78

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. 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.)

Unless otherwise directed by this guide, you should 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 software, that application must also be managed by HA software. 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. In an HA environment, the HA process must control the reset functionality for CXFS. You must use STONITH for system reset and specify the following fail policy in the CXFS configuration (set via the CXFS Manager interface or the `cxfs_admin` command):

```
fence,shutdown
```

For more information, see:

- "`fence_ipmilan` Template" on page 276
- "`ipmi` Template" on page 287
- *CXFS 7 Administrator Guide for SGI InfiniteStorage* chapters about CXFS Manager and `cxfs_admin` For the `cxfs_admin`(8) man page and help text

CXFS Start/Stop Issues

You must start the CXFS cluster `cxfs_cluster` service and CXFS filesystem `cxfs` service before starting the appropriate underlying HA control services (`cman` and pacemaker for RHEL nodes, openais for SLES nodes). 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, as directed later in this procedure (Chapter 8, "Create the Fencing Capability and Put Into Production Mode" on page 185).

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

CXFS Volumes and Filesystems Managed by DMF

The CXFS volumes specified for the cxfs resource must not include any volumes that represent filesystems managed by DMF.

cxfs and cxfs_cluster Services Not Controlled by the HA Environment

Do not disable the cxfs and cxfs_cluster services (that is, do not set the service to off via chkconfig). Unlike other services, these services **will not** be controlled by HA software in an HA cluster.

Local XVM Requirements

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

All local XVM physical volumes (*physvols*) that are managed by HA software 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:

- High-availability 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.

Local Filesystem Requirements

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

The filesystem must not be listed in `/etc/fstab`.

The `Filesystem` resources used with Samba require `fstype="none"`, which differs from the template below, and `options="bind"`.

IP Address Alias Requirements

DMF HA service (in either a CXFS or a local XVM environment) requires at least one `IPaddr2` resource for the DMF and OpenVault server. You must allocate an IP address alias on the subnet used for DMF and OpenVault communication. The address must be a virtual address managed by an `IPaddr2` resource within the same resource group as the `openvault` resource. You must also add an associated virtual hostname to your local DNS and 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.

You may need other aliases, depending on your configuration, 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.

TMF Requirements

To use TMF as the mounting service, all tape devices should be configured as `DOWN` in the TMF configuration file on all nodes. The loaders should be configured as `UP`.

During HA operation, the HA software will control this service.

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

OpenVault Requirements

To use OpenVault as the 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*.
- Provide a directory for OpenVault's use within an HA filesystem in the DMF resource group. This is known as the *serverdir directory*. The directory will hold OpenVault's database and logs. The directory can be either of the following:
  - Within the root of an HA filesystem dedicated to OpenVault use
  - Within another HA filesystem, such as the filesystem specified by the `HOME_DIR` parameter in the DMF configuration file (`/etc/dmf/dmf.conf`)

  In non-HA environments, the OpenVault server's files reside in `/var/opt/openvault/server`. During the conversion to an HA environment, OpenVault will move its databases and logs into the specified directory within an HA 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*.
- During HA operation, the HA software will control the `openvault`service.

COPAN MAID Requirements for a DMF HA Service *(Optional)*

Using COPAN MAID shelves in an active/passive DMF HA cluster consisting of potential DMF servers also requires the following:

- OpenVault must be configured to manage the RAID sets
- In an implementation using local XVM, 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
- 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 6-1 on page 99.

DMF Requirements

This section discusses the following:

- "Potential DMF Server Requirements" on page 82
- "Ordering of Resources" on page 83
- "Virtual Hostname Requirement" on page 83
- "Parallel Data-Mover Option Requirements" on page 83
- "Control of the `dmf` Service" on page 83
- "Use of the Wildcard in OpenVault Configuration" on page 83

Potential DMF Server Requirements

Each potential DMF server must:

- Be a member of the DMF HA cluster
- Run the required product and HA software.
- Have the `ident` service started and configured to restart on reboot, as documented in the *DMF 6 Administrator Guide*.
- Have connectivity to the same set of libraries and drives as every other potential DMF server. (If one node has access to only a subset of the drives when the DMF server is failed over to that node, at that point DMF would not be able to access data on volumes left mounted in inaccessible drives.)
- 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 `Filesystem` resource in that resource group.

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

  The DMF filesystems to be managed are as follows:

  – The managed user filesystems (do not include these in the `volnames` attribute list for the `cxfs` resource)

  – 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 move 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*.

Ordering of Resources

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). See Figure 6-1 on page 99.

Virtual Hostname Requirement

There must be a virtual hostname for use by DMF. See "IP Address Alias Requirements" on page 79.

Parallel Data-Mover Option Requirements

If you are using the DMF Parallel Data-Mover Option, you must define a node object in the DMF configuration file for each potential DMF server. Set the INTERFACE parameter in the node object for each potential DMF server to the same virtual hostname used for SERVER_NAME in the base object.

Control of the dmf Service

During HA operation, the HA software will control the dmf service.

Use of the Wildcard in OpenVault Configuration

The procedure shown in this chapter 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*.

Samba Requirements *(Optional)*

The `/etc/samba` and `/var/lib/samba` directories must be on shared storage and must use `bind` mounts.

The `Filesystem` resources for `/etc/samba` and `/var/lib/samba` require `fstype="none"` (which differs from the `/usr/share/doc/sgi-ha/templates/Filesystem` template), and `options="bind"`. See "`Filesystem` Example for the Samba Directory `bind`-Mounted on `/etc/samba`" on page 282.

DMF Copytool Requirements

Using the DMF copytool for Lustre (`lhsmtool_dmf`) in an HA environment requires the following:

- You must use `dmcopytool-1.0.16` or later for proper execution of the `dmcopytool` resource agent.
- There must be a Lustre filesystem mount point dedicated for use by the DMF copytool. This mount point must exist on both nodes.
- There must be a Lustre filesystem archive mount point defined in the DMF configuration file as a `filesystem` object containing a `MIGRATION_LEVEL` parameter set to `archive`. This mount point must exist on both nodes.
- There must be a directory in a filesystem managed by DMF that is used to archive or restore Lustre files. This filesystem must be mounted and the directory must exist prior to executing the `lhsmtool_dmf`(8) command.

DMF Manager Requirements *(Optional)*

During HA operation, the HA software will control the `dmfman` service. See "Stop Services Related to DMF Before Applying an HA Environment" on page 95.

DMF Client SOAP Service Requirements *(Optional)*

During HA operation, the HA software must control the `dmfsoap` service. See "Stop Services Related to DMF Before Applying an HA Environment" on page 95.

Configure and Test the Service Components Before Applying an HA Environment

Configure and test the service components **before applying an HA environment**, as described in the following:

- Filesystem, one of the following:
  - "Configure and Test CXFS Before Applying an HA Environment" on page 86
  - "Configure and Test Local XVM Before Applying an HA Environment" on page 86
- Mounting service, one of the following:
  - "Configure and Test TMF Before Applying an HA Environment" on page 87
  - "Configure and Test OpenVault Before Applying an HA Environment" on page 87, and *(optional)* "Configure and Test the COPAN MAID OpenVault Client Before Applying a DMF HA Environment" on page 88
- "Configure and Test DMF Before Applying an HA Environment" on page 89
- "Configure and Test the NFS Service for DMF Use Before Applying an HA Environment" on page 89 *(Optional)*
- "Configure and Test Samba Before Applying an HA Environment" on page 91 *(Optional)*
- "Configure and Test the Winbind Service Before Applying an HA Environment" on page 91 *(Optional)*
- "Configure and Test Lustre Before Applying the DMF Copytool an HA Environment" on page 92 *(Optional)*
- "Test DMF Manager Before Applying an HA Environment" on page 95 *(Optional)*
- "Test the DMF Client SOAP Service Before Applying an HA Environment" on page 95 *(Optional)*

Configure and Test CXFS Before Applying an HA Environment

Do the following:

1. Before applying an HA environment, configure CXFS on `node1` (which must be a CXFS server-capable administration node), according to the instructions in the following:
   - "CXFS Requirements" on page 76
   - *CXFS 7 Administrator 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.

Configure and Test Local XVM Before Applying an HA Environment

Before applying an HA environment, use the instructions in the *XVM Volume Manager Administrator Guide* to do the following on `node1` for each of the local XVM filesystems that you want to later 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 service before applying HA, ensure that you can create and delete files in each of the mounted filesystems.

Configure and Test TMF Before Applying an HA Environment

Before applying an HA environment, configure TMF on `node1` according to the instructions in the *TMF 6 Administrator 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 software 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 service before applying HA, 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
   ```

Configure and Test OpenVault Before Applying an HA Environment

Before applying an HA environment, 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*. 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 an HA environment is performed.

To test the OpenVault service before applying HA, 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 using tapes 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   true   false     inuse     loaded    true     DMF105

node1# ov_unmount -A dmf -V DMF105 -d drive0
Unmounted DMF105
node1# exit
```

Repeat the sequence for `drive1`.

Note: The `tsmt` step is only useful when using tapes.

Configure and Test the COPAN MAID OpenVault Client Before Applying a DMF HA Environment

Before applying an HA environment, configure the following OpenVault components for each COPAN MAID shelf by executing the `ov_shelf`(8) command on `node1` (or `mover1`), 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.

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

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

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

Configure and Test DMF Before Applying an HA Environment

Before applying an HA environment, configure DMF according to the instructions in the *DMF 6 Administrator Guide*.

To test the DMF service before applying HA, 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
   ```

Configure and Test the NFS Service for DMF Use Before Applying an HA Environment

Before applying an HA environment, set up the NFS exports in the `/etc/exports` file on both CXFS nodes as you would normally. The `/etc/exports` file should be identical on both nodes.

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

To test the NFS service before applying HA, do the following:

1. Ensure that the node is a member of the CXFS cluster and has filesystems mounted. For example:

   ```
   node1# /usr/cluster/bin/clconf_info
   node1# df
   ```

2. Ensure that the NFS service is running on the CXFS metadata server:

   - RHEL:

     ```
     rhel# service nfs status
     ```

     If it is not running, start it:

     ```
     rhel# service nfs start
     ```

   - SLES:

     ```
     sles# service nfsserver status
     ```

     If it is not running, start it:

     ```
     sles# service nfsserver start
     ```

3. Export the filesystem from `node1`:

   a. Configure `/etc/exports`.

   b. Export the CXFS filesystems:

      ```
      node1# exportfs -a
      ```

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

```
node1# exportfs -v
/nfsexportedfilesystem   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=xxx)
```

4. NFS-mount and test the filesystem from `node1`:

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

      - NFS v3:

        ```
        otherhost# mount -t nfs -o vers=3 node1:/nfsexportedfilesystem /mnt/test
        ```

      - NFS v4:

        ```
        otherhost# mount -t nfs -o vers=4 node1:/nfsexportedfilesystem /mnt/test
        ```

   b. 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
```

5. Repeat steps 3 and 4, exporting and then mounting the filesystem from `node2`.

Configure and Test Samba Before Applying an HA Environment

Before applying an HA environment, set up the Samba service on `node1` as you would normally (before applying HA), but place the Samba configuration files and directories on shared storage.

To test the Samba service before applying HA, 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)

Configure and Test the Winbind Service Before Applying an HA Environment

Before applying the HA environment, configure Winbind and do the following to test it:

1. Verify the configuration:

   a. List the users of the domain:

      ```
      node1# wbinfo -u
      ```

   b. List the groups:

      ```
      node1# wbinfo -g
      ```

2. Verify the `nsswitch` module:

   a. Verify the password:

      ```
      node1# sudo getent passwd
      ```

   b. Verify the groups:

      ```
      node1# sudo getent group
      ```

Configure and Test Lustre Before Applying the DMF Copytool an HA Environment

Before applying an HA environment, you must configure Lustre and test it to ensure that the basic Lustre HSM commands operate normally. For example, perform the following sanity test:

1. Archive and release files using the tier-1 storage (`t1`), which will be the XFS or CXFS filesystem managed by DMF, periodically checking the state:

```
node1# dd if=/dev/urandom of=/lustre/test_t1 bs=1M count=1
node1# md5sum /lustre/test_t1          (record this value for later comparison)

node1# lfs hsm_archive -Dt1 /lustre/test_t1
node1# lfs hsm_state /lustre/test_t1   (after a few seconds, this should be "exists archived")

node1# lfs hsm_release /lustre/test_t1
node1# lfs hsm_state /lustre/test_t1   (after a few seconds, this should be "released exists archived")

node1# md5sum /lustre/test_t1          (implicitly recalls the file, value should match above value)
node1# lfs hsm_state /lustre/test_t1   (should be "exists archived)

node1# lfs hsm_remove /lustre/test_t1
node1# lfs hsm_state /lustre/test_t1   (no archive state)

node1# rm /lustre/test_t1
```

For example:

```
node1# dd if=/dev/urandom of=/lustre/test_t1 bs=1M count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB) copied, 0.114834 s, 9.1 MB/s
node1# md5sum /lustre/test_t1
701e3c19baea92096004ff94c6695d46  /lustre/test_t1

node1# lfs hsm_archive -Dt1 /lustre/test_t1
node1# lfs hsm_state /lustre/test_t1
/lustre/test_t1: (0x00000009) exists archived, archive_id:1

node1# lfs hsm_release /lustre/test_t1
node1# lfs hsm_state /lustre/test_t1
/lustre/test_t1: (0x0000000d) released exists archived, archive_id:1

node1# md5sum /lustre/test_t1
701e3c19baea92096004ff94c6695d46  /lustre/test_t1
node1# lfs hsm_state /lustre/test_t1
/lustre/test_t1: (0x00000009) exists archived, archive_id:1

node1# lfs hsm_remove /lustre/test_t1
node1# lfs hsm_state /lustre/test_t1
/lustre/test_t1: (0x00000000), archive_id:1

node1# rm /lustre/test_t1
rm: remove regular file '/lustre/test_t1'? y
node1# lfs hsm_state /lustre/test_t1
can't get hsm state for /lustre/test_t1: No such file or directory
```

2. Archive and release a file using the tier-2 storage (`t2`), which will be archived directly to the storage on disk/tape:

Note: This test will take longer to complete than the tier–1 case (perhaps several minutes).

```
node1# dd if=/dev/urandom of=/lustre/test_t2 bs=1M count=1
node1# md5sum /lustre/test_t2            (record this value for later comparison)

node1# lfs hsm_archive -Dt2 /lustre/test_t2
```

```
node1# dmdidle                             (force the data out to tape)
node1# lfs hsm_state /lustre/test_t2       (after a few minutes, this should be "exists archived")

node1# lfs hsm_release /lustre/test_t2
node1# lfs hsm_state /lustre/test_t2       (after a few minutes, this should be "released exists archived")

node1# md5sum /lustre/test_t2              (implicitly recalls the file, value should match above value)
node1# lfs hsm_state /lustre/test_t2       (should be "exists archived)

node1# lfs hsm_remove /lustre/test_t2
node1# lfs hsm_state /lustre/test_t2       (no archive state)

node1# rm /lustre/test_t2
```

For example:

```
node1# dd if=/dev/urandom of=/lustre/test_t2 bs=1M count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB) copied, 0.114834 s, 9.1 MB/s
node1# md5sum /lustre/test_t2
2cd70bb4e9ec10de116c9d1937f58d22  /lustre/test_t2

node1# lfs hsm_archive -Dt2 /lustre/test_t2
node1# dmdidle
node1# lfs hsm_state /lustre/test_t2
/lustre/test_t2: (0x00000009) exists archived, archive_id:1

node1# lfs hsm_release /lustre/test_t2
node1# lfs hsm_state /lustre/test_t2
/lustre/test_t2: (0x0000000d) released exists archived, archive_id:1

node1# md5sum /lustre/test_t2
2cd70bb4e9ec10de116c9d1937f58d22  /lustre/test_t2
node1# lfs hsm_state /lustre/test_t2
/lustre/test_t2: (0x00000009) exists archived, archive_id:1

node1# lfs hsm_remove /lustre/test_t2
node1# lfs hsm_state /lustre/test_t2
/lustre/test_t2: (0x00000000), archive_id:1

node1# rm /lustre/test_t2
```

```
rm: remove regular file ‘/lustre/test_t2’? **y**
node1# **lfs hsm_state /lustre/test_t2**
can’t get hsm state for /lustre/test_t2: No such file or directory
```

Test DMF Manager Before Applying an HA Environment

Before applying an HA environment, verify that DMF Manager is operational by pointing your browser to the following address:

`https://`*YOUR_DMF_SERVER*`:11109`

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

Test the DMF Client SOAP Service Before Applying an HA Environment

Before applying an HA environment, verify that the standard DMF client SOAP service is operational pointing your browser to the following address:

`https://`*YOUR_DMF_SERVER*`:11110/server.php`

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

Ensure that the Base HA Cluster is Operational

Ensure that the base HA cluster is operational, as described in "Test the Base HA Cluster" on page 45.

Stop Services Related to DMF Before Applying an HA Environment

Do the following on both nodes to ensure that the services will be controlled by the DMF HA service:

- RHEL:

  RHEL node1:

  ```
  node1# **chkconfig dmf off**
  node1# **chkconfig dmfman off**
  node1# **chkconfig dmfsoap off**
  ```

```
node1# chkconfig nfs off
node1# chkconfig openvault off

node1# service dmf stop
node1# service dmfman stop
node1# service dmfsoap stop
node1# service nfs stop
node1# service openvault stop
```

RHEL node2:

```
node2# chkconfig dmf off
node2# chkconfig dmfman off
node2# chkconfig dmfsoap off
node2# chkconfig nfs off
node2# chkconfig openvault off

node2# service dmf stop
node2# service dmfman stop
node2# service dmfsoap stop
node2# service nfs stop
node2# service openvault stop
```

- SLES:

  SLES node1:

```
node1# chkconfig dmf off
node1# chkconfig dmfman off
node1# chkconfig dmfsoap off
node1# chkconfig nfsserver off
node1# chkconfig openvault off
node1# chkconfig tmf off (optional)

node1# service dmf stop
node1# service dmfman stop
node1# service dmfsoap stop
node1# service nfsserver stop
node1# service openvault stop
node1# service tmf stop  (optional)
```

SLES `node2`:

```
node2# chkconfig dmf off
node2# chkconfig dmfman off
node2# chkconfig dmfsoap off
node2# chkconfig nfsserver off
node2# chkconfig openvault off
node2# chkconfig tmf off (optional)

node2# service dmf stop
node2# service dmfman stop
node2# service dmfsoap stop
node2# service nfsserver stop
node2# service openvault stop
node2# service tmf stop  (optional)
```

Note: Do not disable the `cxfs` and `cxfs_cluster` services. Unlike other services, these services **will not** be controlled by HA software in an HA cluster.

Add the Resources in the Correct Order

This section discusses the following:

- "Summary of the DMF HA Resources" on page 98
- "Add the `dmf-group` and Clustered/Local Filesystems Resources" on page 101
- "Add the `IPaddr2` Resource (DMF HA)" on page 107
- "Add the Mounting Service Resources" on page 110
- "Add the `dmf` Resource" on page 135
- " Add the `nfsserver` Resource *(Optional)*" on page 142
- "Add the Samba Resources: `smb`, `nmb`, and `winbind` *(Optional)*" on page 146
- "Add the `dmcopytool` Resources *(Optional)*" on page 154
- "Add the `dmfman` Resource *(Optional)*" on page 159
- "Add the `dmfsoap` Resource *(Optional)*" on page 162

Summary of the DMF HA Resources

For the DMF HA implementation, you will create a single resource group (using an example ID name of `DMF-GROUP`) that will contain all of the other resources, adding one resource at a time to the group and then testing it.

Note: The ID names (such as `DMF-GROUP`) are site-configurable. For simplicity, this guide uses the defaults found in the template files.

As a starting point, use the templates in `/usr/share/doc/sgi-ha/templates` as building blocks. Add the resources in the order shown in Figure 6-1 on page 99, skipping products that do not apply to your site.

Figure 6-1 shows the overall DMF HA implementation in a two-node active/passive HA cluster (`node1` and `node2`), using the suggested default IDs found in the templates in `/usr/share/doc/sgi-ha/templates`. This map also describes the start/stop order for resources. Table 6-1 lists the resource-specific sections that provide the details.

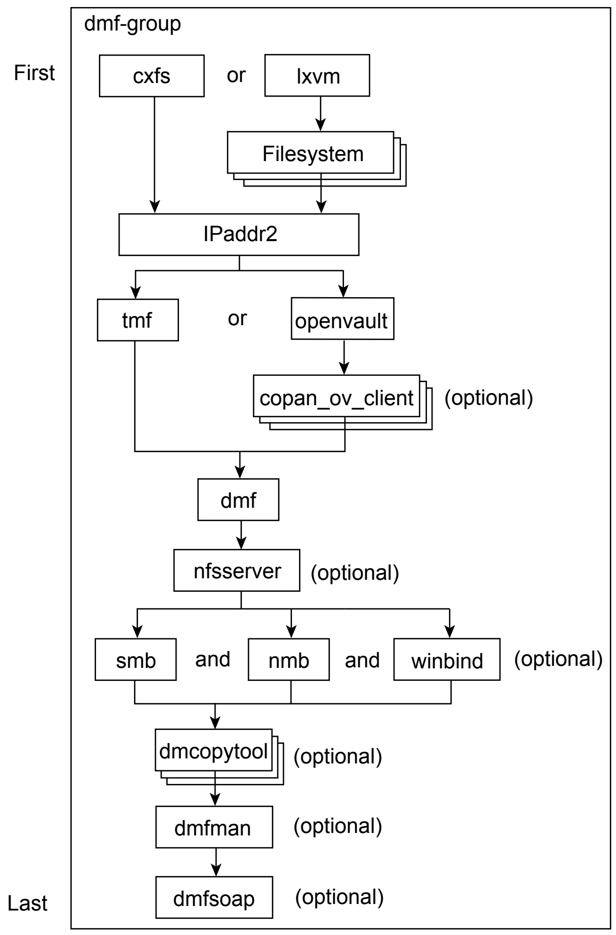

Figure 6-1 Map of Resources for the DMF HA Service

Table 6-1 Order of Resources Used in the DMF HA Service

Order	Purpose	Resource Type	Example ID	Instructions
First	Group name and Filesystem	`cxfs`	`DMF-GROUP` and `CXFS`	"Add the `dmf-group` and `CXFS` Resources" on page 102
			— or —	
		`lxvm` and `Filesystem`	`DMF-GROUP` and `LXVM` *plus* `FILESYSTEM-`*n* *(one or more)*	"Add the `dmf-group` and `lxvm` and `Filesystem` Resources" on page 105
	IP alias	`IPaddr2`	`IP`	"Add the `IPaddr2` Resource (DMF HA)" on page 107
	Mounting service	`tmf`	`TMF`	"Add the `tmf` Resource" on page 112
			— or —	
		`openvault`	`OV` *plus*	"Add the `openvault` Resource" on page 118
		`copan_ov_client`	`C0`*n* *(one or more)*	"Add the `copan_ov_client` Resource *(Optional)*" on page 127
	DMF	`dmf`	`DMF`	"Add the `dmf` Resource" on page 135
	NFS *(optional)*	`nfsserver`	`NFS`	" Add the `nfsserver` Resource *(Optional)*" on page 142
	Samba *(optional)*	`smb` `nmb` `winbind`	`SMB` `NMB` `WINBIND`	"Add the Samba Resources: `smb`, `nmb`, and `winbind` *(Optional)*" on page 146
	DMF copytool *(optional)*	`dmcopytool`	`DMCOPYTOOL`	"Add the `dmcopytool` Resources *(Optional)*" on page 154
	DMF Manager *(optional)*	`dmfman`	`DMFMAN`	"Add the `dmfman` Resource *(Optional)*" on page 159
Last	DMF SOAP *(optional)*	`dmfsoap`	`DMFSOAP`	"Add the `dmfsoap` Resource *(Optional)*" on page 162

Add the `dmf-group` and Clustered/Local Filesystems Resources

The DMF HA implementation uses a single `group` resource that contains all of the other required resources, so that HA processes can coherently control all of the resources, starting and stopping them in the required order. (The name of the group can be any name you choose.)

Figure 6-2 shows the step to add the group for the DMF HA service and the choice between a CXFS environment or a local XVM environment.

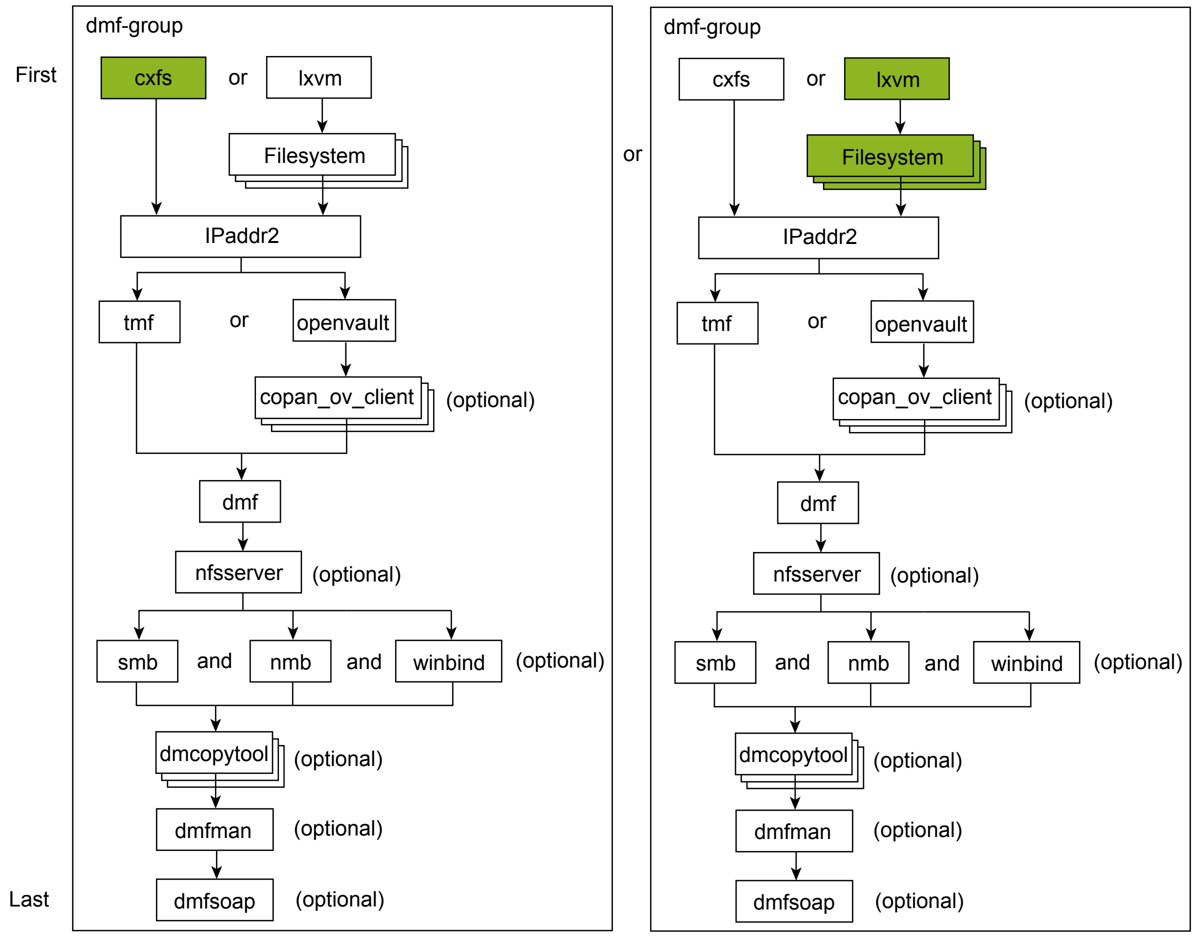

Figure 6-2 Adding the Resources for the DMF Group and the Clustered / Local Filesystems

You will create a group either for the CXFS environment using cluster XVM or for individual XFS filesystems using the local XVM environment:

- "Add the `dmf-group` and `CXFS` Resources" on page 102
- "Add the `dmf-group` and `lxvm` and `Filesystem` Resources" on page 105

Add the `dmf-group` and `CXFS` Resources

The purpose of this resource is to ensure that the defined filesystems run on the DMF server in a CXFS environment. This resource controls the location of the CXFS metadata server for CXFS filesystems that must be run on the DMF server, such as the DMF administrative filesystems, filesystems holding OpenVault server configuration, and filesystems for NFS/Samba configuration. This resource must not include any filesystems managed by DMF.

See the *DMF 6 Administrator Guide* for filesystem requirements, such as mount options.

Do the following:

1. Create a new partial configuration file (*workfile*) that contains the following:

   ```
   group DMF-GROUP CXFS
   ```

2. Copy the `primitive` text from the `cxfs` template into *workfile* and replace the site-specific variables as directed. The template is located in:

   ```
   /usr/share/doc/sgi-ha/templates/cxfs
   ```

   See "Use the Templates as Building Blocks" on page 18.

   A `cxfs` resource controls of the CXFS metadata location. It is used in a DMF HA service.

   Use the following:

   ```
   primitive CXFS ocf:sgi:cxfs \
      op monitor interval="120s" timeout="180s" on-fail="restart" \
      op monitor interval="0" timeout="180s" \
      op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
      op stop interval="0" timeout="60s" on-fail="fence" \
      params volnames="VOLUME-LIST" \
      meta resource-stickiness="1" migration-threshold="1"
   ```

Variable	**Description**
CXFS	Name of this resource instance, such as `CXFS`
VOLUME-LIST	Comma-separated list of volumes under the `/dev/cxvm` directory, such as: `cxfsvol1,cxfsvol2` **Note:** Do not include any links or volumes that are filesystems managed by DMF.

For example:

```
primitive CXFS ocf:sgi:cxfs \
   op monitor interval="120s" timeout="180s" on-fail="restart" \
   op monitor interval="0" timeout="180s" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="60s" on-fail="fence" \
   params volnames="cxfsvol1,cxfsvol2" \
   meta resource-stickiness="1" migration-threshold="1"
```

3. Verify that the `timeout` values are appropriate for your site.
4. Verify that there are no comments in *workfile*.
5. Save *workfile*.
6. Update the database:

```
node1# crm configure load update workfile
```

Note: As a best practice, you should also run the following command to verify changes you make to the CIB:

```
node1# crm_verify -LV
```

For testing purposes, ignore any `stonith` errors at this point.

For simplicity, this step is not included in the following procedures but is recommended. For more information, see "Use the `crm_verify` Command to Verify Configuration" on page 24.

7. Test the resource:

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

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

   ```
   node1# df -lh
   ```

   ii. 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"
   ```

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

   ```
   node1# crm resource move DMF-GROUP node2
   ```

c. Verify the status:

   ```
   node1# crm status inactive
   ```

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

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

   Note: Only those volumes in `/dev/cxvm` that are **not** managed by DMF will now be served on `node2` instead of `node1`.

   ---

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

   ```
   node1# crm resource move DMF-GROUP node1
   ```

f. Verify the status:

   ```
   node1# crm status inactive
   ```

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

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

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

```
node1# crm resource unmove DMF-GROUP
```

Add the `dmf-group` and `lxvm` and `Filesystem` Resources

Note: See the *DMF 6 Administrator Guide* for filesystem requirements, such as mount options.

The DMF HA service in a local XVM environment requires the `lxvm` and `Filesystem` resources:

Do the following:

1. Make sure that none of the filesystems to be controlled are mounted on either node.
2. Make sure that none of the filesystems to be controlled are present in `/etc/fstab` on either node.
3. Make sure that the local XVM volumes are visible and online on `node1`.
4. Create a new partial configuration file (*workfile*) that contains the following:

   ```
   group DMF-GROUP LXVM
   ```

5. Copy the `primitive` text from the `lxvm` template into *workfile* and replace the site-specific variables as directed in the template comments or in "`lxvm` Template" on page 290. The template is located in:

   ```
   /usr/share/doc/sgi-ha/templates/lxvm
   ```

   See "Use the Templates as Building Blocks" on page 18.

6. Verify that the `timeout` values are appropriate for your site.
7. Verify that there are no comments in *workfile*.
8. Save *workfile*.
9. Update the database:

   ```
   node1# crm configure load update workfile
   ```

Note: As a best practice, you should also run the following command to verify changes you make to the CIB:

```
node1# crm_verify -LV
```

For simplicity, this step is not included in the following procedures but is recommended. For more information, see "Use the `crm_verify` Command to Verify Configuration" on page 24.

---

10. Test the resource:

    a. Move the resource group containing the `lxvm` resource from `node1` to `node2`:

    ```
    node1# crm resource move DMF-GROUP node2
    ```

    b. Verify the status:

    ```
    node1# crm status inactive
    ```

    Note: If the timeout is too short for a `start` operation, the `crm status inactive` and `crm_verify -LV` output and the log files will have an entry that refers to the action being “`Timed Out`” (see "Examine Log Files" on page 24). For example (line breaks shown here for readability):

    ```
    node1# crm status inactive | 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
    ```

    ---

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

    ```
    node1# crm resource move DMF-GROUP node1
    ```

    d. Verify the status:

    ```
    node1# crm status inactive
    ```

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

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

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

```
node1# crm resource unmove DMF-GROUP
```

Add the `IPaddr2` Resource (DMF HA)

Figure 6-3 shows the step to add the `IPaddr2` resource for the IP alias.

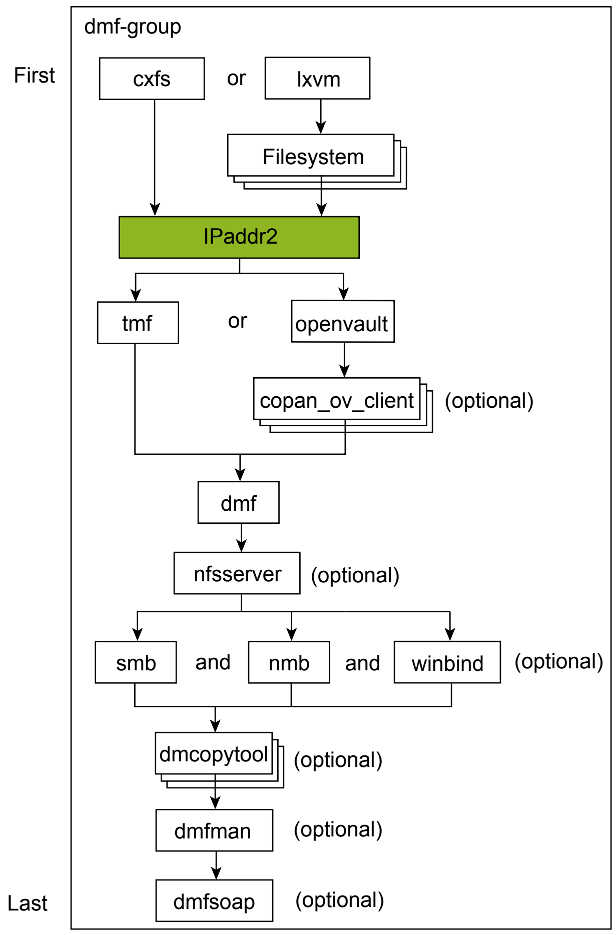

Figure 6-3 Adding the Resource for the IP Alias

Do the following to add the IP address alias (`IPaddr2`) resource:

1. Create another *workfile* that contains the following, where *Previously_Added_Resources* are all of the resources added to this point in the procedure. For example, for the DMF HA example used in this guide:

   ```
   group DMF-GROUP Previously_Added_Resources IP
   ```

   For more information about the group definition, see "`dmf-group` Template" on page 268.

2. Copy the `primitive` in the `IPaddr2` template into *workfile* and replace the site-specific variables as directed. The template is located in:

   ```
   /usr/share/doc/sgi-ha/templates/IPaddr2
   ```

   See "Use the Templates as Building Blocks" on page 18.

   An `IPaddr2` resource controls the addition/deletion of an IP address alias on a network interface. It is used either a CXFS environment or a local XVM environment.

   Use the following:

   ```
   primitive IP ocf:heartbeat:IPaddr2 \
      op monitor interval="0" timeout="30s" \
      op start interval="0" timeout="90s" on-fail="restart" requires="fencing" \
      op stop interval="0" timeout="100s" on-fail="fence" \
      params ip="IP-ADDRESS" \
      meta resource-stickiness="1" migration-threshold="1"
   ```

Variable	**Description**
IP	Name of this resource instance, which must match the named used to define it in the associated `group` resource, such as example `IP` used DMF HA example in this guide
IP-ADDRESS	IP address of the virtual channel, such as `128.162.244.240`

   For example:

   ```
   primitive IP ocf:heartbeat:IPaddr2 \
      op monitor interval="0" timeout="30s" \
      op start interval="0" timeout="90s" on-fail="restart" requires="fencing" \
   ```

```
   op stop interval="0" timeout="100s" on-fail="fence" \
   params ip="128.162.244.240" \
   meta resource-stickiness="1" migration-threshold="1"
```

The `monitor` operation probes to see if the resource is already running. Normally, this is the only `monitor` operation required for an `IPaddr2` resource.

3. Verify that the `timeout` values are appropriate for your site.
4. Verify that there are no comments in *workfile*.
5. Save *workfile*.
6. Update the database with the new resource:

   ```
   node1# crm configure load update workfile
   ```

7. Test the `IPaddr2` resource:

   a. Verify that the IP address 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
```

   b. 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#
      ```

   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`:

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

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

      ```
      node1# crm resource move DMF-GROUP node2
      ```

e. Verify the status:

```
node1# crm status inactive
```

f. 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
```

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

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

h. 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
```

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

```
node1# crm resource move DMF-GROUP node1
```

j. Verify the status:

```
node1# crm status inactive
```

k. Test again as in steps a-c above.

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

```
node1# crm resource unmove DMF-GROUP
```

Add the Mounting Service Resources

Figure 6-4 shows the step to add the resource for the mounting service, which is either the TMF mounting service or the OpenVault mounting service using optional COPAN OpenVault client resources.

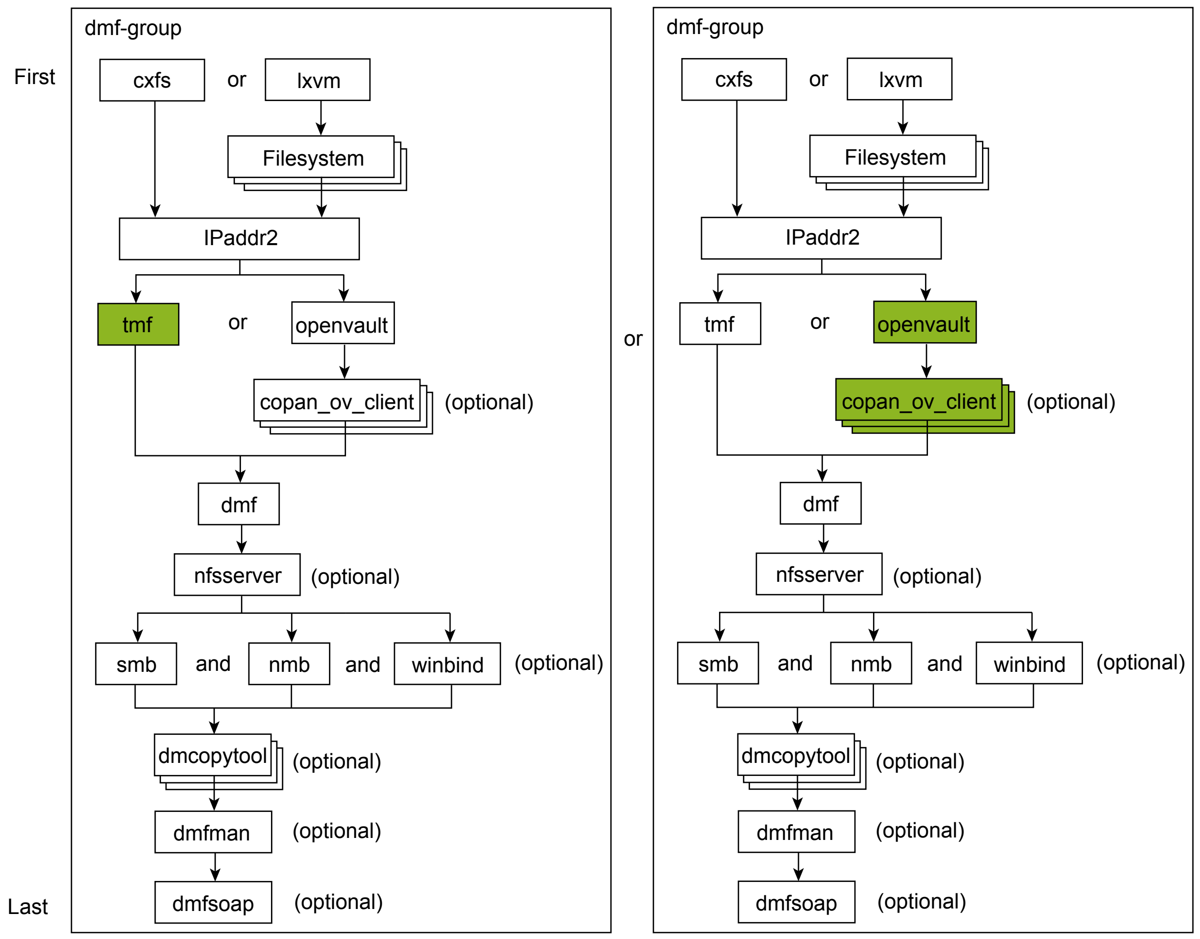

Figure 6-4 Adding the Resource for the Mounting Service

This section discusses the following:

- "Add the `tmf` Resource" on page 112
- "Add the `openvault` Resource" on page 118
- "Create the OpenVault Components on the Passive Node (COPAN MAID Only)" on page 125

- "Add the `copan_ov_client` Resource *(Optional)*" on page 127
- "Add Any OpenVault Clients that are Not DMF Servers *(Optional)*" on page 132
- "Test the `openvault` Resource" on page 133

Add the `tmf` Resource

To configure TMF for an HA environment, 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
   ```

3. 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.

4. Create another *workfile* that contains the following:

   `group DMF-GROUP` *Previously_Added_Resources* `TMF`

   For more information about the group definition, see "`dmf-group` Template" on page 268.

5. Copy the `primitive` in the `tmf` template into *workfile* and replace the site-specific variables as directed. The template is located in:

   ```
   /usr/share/doc/sgi-ha/templates/tmf
   ```

   See "Use the Templates as Building Blocks" on page 18.

   A `tmf` resource controls the start/stop of TMF. It may be used in a DMF HA service.

   Use the following:

   ```
   primitive TMF ocf:sgi:tmf \
      op monitor interval="120s" timeout="60s" on-fail="restart" \
      op monitor interval="0" timeout="60s" \
      op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   ```

```
op stop interval="0" timeout="600s" on-fail="fence" \
params devgrpnames="DEVGRPNAME-LIST" mindevsup="MINDEVSUP-LIST" \
       devtimeout="DEVTIMEOUT-LIST" loader_names="LOADERNAME-LIST" \
       loader_hosts="HOST-LIST" loader_users="USER-LIST" \
       loader_passwords="PASSWORD-LIST" admin_emails="EMAIL-ADDRESS-LIST" \
meta resource-stickiness="1" migration-threshold="1"
```

Variable	Description
TMF	Name of this resource instance, such as `TMF`.
DEVGRPNAME-LIST	Comma-separated list of TMF device groups defined in the `tmf.config` file that are to be managed by HA software, such as: `ibm3592,t10ka`
MINDEVSUP-LIST	Comma-separated list of the minimum 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-LIST	Comma-separated list of device 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`

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 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 value of 90+18+10=118.

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 value in *DEVTIMEOUT-LIST*. For example, 2*118=236. You should usually use the same `timeout` value for `start` and `stop`.

LOADERNAME-LIST

Comma-separated list of loader names configured in *DEVGRPNAME-LIST* `tmf.config` that correspond to the device groups listed in *DEVGRPNAME-LIST*, such as:

```
ibm3494,l700a
```

HOST-LIST

Comma-separated list of hosts through which the corresponding loaders listed in *LOADERNAME-LIST* are controlled, such as:

```
ibm3494cps,stkacsls
```

USER-LIST	Comma-separated list of user names that are used to log in to the corresponding hosts listed in *HOST-LIST*, such as: `root,acssa`
PASSWORD-LIST	Comma-separated list of passwords corresponding to the user names listed in *USER-LIST*, such as: `passwd1,passwd2`
EMAIL-ADDRESS-LIST	(Optional) Comma-separated list of administrator email addresses corresponding to the device groups listed in *DEVGRPNAME-LIST*, 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.

For example:

```
primitive TMF ocf:sgi:tmf \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="600s" on-fail="fence" \
   params devgrpnames="ibm3592,t10ka" mindevsup="1,0" devtimeout="120,240" \
          loader_names="ibm3494,l700a" loader_hosts="ibm3494cps,stkacsls" \
          loader_users="root,acssa" loader_passwords="passwd1,passwd2" \
          admin_emails="root,admin1" \
   meta resource-stickiness="1" migration-threshold="1"
```

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

6. Verify that the `timeout` values are appropriate for your site.

7. Verify that there are no comments in *workfile*.

8. Save *workfile*.

9. Update the database with the new resource:

   ```
   node1# crm configure load update workfile
   ```

10. Test the new `tmf` resource:

    a. Use `tmmls` to show the loader status.

    b. Use `tmstat` to show the drive status. Verify that all of the tape drives in all HA device groups are in `assn` or `idle` status on `node1`.

    c. Move the resource group containing the `TMF` resource to `node2`:

       ```
       node1# crm resource move DMF-GROUP node2
       ```

    d. Verify the status:

       ```
       node1# crm status inactive
       ```

    e. 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`

    f. Verify that the `timeout` values for the `start`, `stop`, and `monitor` operations are appropriate. Use the following guidelines:

       i. On `node2`, look in the log files (see "Examine Log Files" on page 24) for the time when the resource start operation started and ended. Also capture the start and end times of the `monitor` operation.

       ii. On `node1`, look in the log files to find the start and stop times for the stop operation.

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

       iv. Based on the above, choose values that you estimate will be acceptable timeouts that are sufficiently long so that you do not risk unnecessary failovers. In general, start with a longer timeout and shorten as required.

Following are examples of finding the `start`, `stop`, and `monitor` operation durations on a SLES system (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
```

g. Modify values in the `primitive` definition as needed.

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

```
node1# crm resource move DMF-GROUP node1
```

i. Verify the status:

```
node1# crm status inactive
```

j. 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`

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

```
node1# crm resource unmove DMF-GROUP
```

Add the `openvault` Resource

Do the following to configure OpenVault for an HA environment:

1. Ensure that all of the resources within the resource group are moved back to `node1` (if not already there).
2. Create another *workfile* that contains the following:

   ```
   group DMF-GROUP Previously_Added_Resources OV
   ```

   For more information about the group definition, see "`dmf-group` Template" on page 268.
3. Copy the `primitive` in the `openvault` template into *workfile* and replace the site-specific variables as directed. The template is located in:

   ```
   /usr/share/doc/sgi-ha/templates/openvault
   ```

   See "Use the Templates as Building Blocks" on page 18.

   An `openvault` resource controls the start/stop of all OpenVault processes. It is used in a DMF HA service.

   Use the following:

   ```
   primitive OV ocf:sgi:openvault \
      op monitor interval="120s" timeout="60s" on-fail="restart" \
      op monitor interval="0" timeout="60s" \
      op start interval="0" timeout="300s" on-fail="restart" requires="fencing" \
      op stop interval="0" timeout="90s" on-fail="fence" \
      params virtualhost="VIRTUALHOSTVALUE" serverdir="SERVERDIR" \
      meta resource-stickiness="1" migration-threshold="1" is-managed="false"
   ```

Variable	Description
OV	Name of this resource instance, such as `OV`.
VIRTUALHOSTVALUE	Hostname where the OpenVault server will be listening (which must also have its own `IPaddr2`

resource instance, see "`IPaddr2` Template" on page 283).

SERVERDIR

The directory that will eventually contain the OpenVault server configuration. *SERVERDIR* could be a directory that will be dedicated for OpenVault use (such as `/dmf/openvault`) or it could be an HA filesystem in the same resource group that has sufficient space (such as `/dmf/home`) to contain the subdirectory (such as `/dmf/home/openvault`) and its contents. The filesystem must be either:

- A directory that will become a mountable CXFS filesystem managed by a `cxfs` resource
- A directory that will become a mountable XFS filesystem managed by a `Filesystem` resource in the same resource group as the `openvault` resource

Note: As part of the conversion to an HA environment, OpenVault will create this directory and move its database and logs into the directory; OpenVault will fail if the directory already exists.

For example, if you define *SERVERDIR* as `/dmf/home/openvault`, the parent directory `/dmf/home` directory can exist but the entire path `/dmf/home/openvault` **must not** exist.

For example:

```
primitive OV ocf:sgi:openvault \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="300s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="90s" on-fail="fence" \
   params virtualhost="myvirtualhost" serverdir="/dmf/home/openvault" \
   meta resource-stickiness="1" migration-threshold="1" is-managed="false"
```

Note: For the initial configuration process, the setting for the `is-managed` attribute must be `false` as shown above. The step in section "Make the `openvault` Resource HA-Managed" on page 127 will reset this attribute to `true` so that the resource will run under HA control.

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

4. Verify that the `timeout` values are appropriate for your site.
5. Verify that there are no comments in *workfile*.
6. Save *workfile*.
7. Update the database with the new resource:

   ```
   node1# crm configure load update workfile
   ```

8. Run `ov_admin` on `node1`, in order to convert the OpenVault server into an HA configuration and move the OpenVault database to the directory specified for the `serverdir` parameter specified above in step 3:

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

   Verify that the server hostname matches the `virtualhost` value you specified in step 3 above. If the server name matches, then the OpenVault server is properly configured for HA.

9. Exit `ov_admin`.

Note: From here forward, whenever `ov_admin` asks for the server hostname, you must use *virtualhostvalue*.

---

10. After allowing a few moments for the OpenVault server conversion to complete, verify that there are no unusable libraries or drives on `node1` by using the `ov_stat`(8) command with the `-ldxy` options (the `-y` option restricts the output to **unusable** LCPs and DCPs). If everything is operating normally at this point, there should be **no** output. For example:

```
node1# ov_stat -ldxy
node1#
```

However, if there are DCPs listed, it may be that the DCPs have not yet connected to the new OpenVault server configured above. (This is particularly likely if you have parallel data-mover nodes, which may take more time to reconnect.) Wait a few moments and then issue the `ov_stat -ldxy` command again until there is no output.

11. 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 select the step to activate a client machine:

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

       Answer the questions as following, optionally supplying a security key:

       ```
       Which Client Machine do you want to activate? [] node2
       What security key would you like the Client Machine node2 to use? [none] mykey
       Will DCPs and/or LCPs also be configured to run on "node2"? [Yes] yes
       ```

       Note: If you originally entered the server name when activating the `dmf` application instances on `node1` (rather than the wildcard `*` character, which allows the `dmf` application to be used from any host, as suggested in the *DMF 6 Administrator Guide*), you must also create a privileged and an unprivileged `dmf` application instance for `node2`.

    b. On `node2`, use `ov_admin` to enable the node to issue administrative commands by using *virtualhostvalue* (the default) entering *mykey* if you specified the key in step 11a:

       ```
       node2# ov_admin
       ...
       Name where the OpenVault server is listening? [virtualhostvalue]
       What port number is the OpenVault server on virtualhostvalue using? [695]
       What security key would you like the admin commands to use? [none] mykey
       ```

12. Define DCPs and LCPs on the passive node (`node2`).

Note: If your site contains COPAN native MAID shelves, you will create their OpenVault components later in "Create the OpenVault Components on the Passive Node (COPAN MAID Only)" on page 125. Therefore, you can skip this step if your site contains only COPAN native MAID shelves (and no physical tape library or COPAN VTL).

---

a. Configure drives by selecting the following:

```
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 the following:

```
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`.
- When prompted for the drive name at a given element address, use the `ov_drive`(8) command to the determine the drive name that corresponds to serial number provided in the prompt. For example `80300189-D00`:

```
For the drive at location "drive 0 VENDOR='' PRODUCT='' SERIAL='80900229-D00'
    Enter a drive name for the element address "256":
```

In the above, the serial number is 80900229-D00, which corresponds to C16d00 in the following ov_drive(8) output (truncated):

```
node2# ov_drive
Drives:
Drive           Drive Group     Disabled     Vendor          Product         Serial Number
sl48_1          lto3            false        HP              Ultrium 3-SCSI  80900229-D0
```

All DCPs and LCPs have now been configured and started on node2.

c. On node1:

i. Verify that the DCPs are running successfully by using the -dx options to ov_stat. For example, the following output shows that the DCPs are usable and ready for the OpenVault server on the active HA node (node1) and running in disconnected mode (unusable and inactive) on the passive node (node2):

```
node1# ov_stat -dx
DCPName        DriveName Usable SoftState    Disabled MsgLevel
sl48_1a@node1 sl48_1    true   disconnected false    debug
sl48_1a@node2 sl48_1    true   disconnected false    debug
sl48_1b@node1 sl48_1    true   ready        false    debug
sl48_1b@node2 sl48_1    true   disconnected false    debug
```

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 that the LCPs are usable and ready for the OpenVault server on the active HA node (node1) and running in disconnected mode (unusable and inactive) on the passive node (node2):

```
node1# ov_stat -lx
LCPName     LibraryName Usable SoftState    Disabled MsgLevel
sl48a@node1 sl48        true   disconnected false    debug
sl48a@node2 sl48        true   disconnected false    debug
sl48b@node1 sl48        true   ready        false    debug
```

```
sl48b@node2 sl48        true   disconnected false    debug
```

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.

d. Exit `ov_admin`.

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

```
node2# ov_stop
```

14. Run `ov_admin` on each parallel data-mover node:

a. Enter *virtualhostvalue* and the port number (and security key *mykey*, if needed):

```
mover# ov_admin
...
Name where the OpenVault server is listening?  [servername] virtualhostvalue
What port number is the OpenVault server on virtualhostvalue using? [695]
What security key would you like the admin commands to use? [none] mykey
```

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

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

d. Exit `ov_admin`.

e. Restart the OpenVault client components (DCPs and any LCPs) to connect to the OpenVault server using the virtual server name:

```
mover# service openvault stop
mover# 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 C00 is owned by owner_nodename
```

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

```
node1# ov_stop
```

Create the OpenVault Components on the Passive Node (COPAN MAID Only)

Note: This step only applies if you have COPAN MAID connected to the active DMF server and passive DMF server. If you do not, skip to "Make the `openvault` Resource HA-Managed" on page 127.

When you configured the services according to the information in *COPAN MAID for DMF Quick Start Guide* before applying HA, you executed an `ov_shelf`(8) command for each shelf in order to create the required OpenVault components (see "Configure and Test the COPAN MAID OpenVault Client Before Applying a DMF HA Environment" on page 88).

In this step, you will create corresponding OpenVault components for the passive 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`
- Passive node: `node2`

Note: For more information about the shelf ID, 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.

Make the `openvault` Resource HA-Managed

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

```
node1# crm resource manage OV
```

The conversion is now complete (the `is-managed` attribute for the `openvault` primitive is now set to `true`).

Add the `copan_ov_client` Resource *(Optional)*

1. Create another *workfile* that contains the following, where *Cxx* is the COPAN MAID shelf ID:

```
group DMF-GROUP Previously_Added_Resources  Cxx
```

For example, using `C00` for cabinet 0 shelf 0:

```
group DMF-GROUP Previously_Added_Resources C00
```

For more information about the group definition, see "`dmf-group` Template" on page 268.

2. Copy the `primitive` in the `copan_ov_client` template into *workfile* and replace the site-specific variables for the appropriate section as directed. The template is located in:

   `/usr/share/doc/sgi-ha/templates/copan_ov_client`

   See "Use the Templates as Building Blocks" on page 18.

   A `copan_ov_client` resource defines a COPAN MAID shelf to be used by an OpenVault client. It is used in a COPAN MAID OpenVault client HA service or optionally in a DMF HA service. Each shelf requires its own `primitive` instance.

   Use the following:

   ```
   primitive SHELF ocf:sgi:copan_ov_client \
      op monitor interval="120s" timeout="60s" on-fail="restart" \
      op monitor interval="0" timeout="60s" \
      op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
      op stop interval="0" timeout="120s" on-fail="fence" \
      params shelf_name="SHELF" \
      meta resource-stickiness="1" migration-threshold="1"
   ```

   Note: The `copan_ov_client` template contains two sections, one for a COPAN MAID OpenVault client HA service and another for a DMF HA service – you must fill in the values in the section for the service you want to configure and delete the other section entirely.

Variable	**Description**
SHELF	Name of this resource instance, normally the three-character COPAN shelf ID, using the naming convention described in the *COPAN MAID for DMF Quick Start Guide*, such as `C00` for cabinet 0, shelf 0 (the bottom shelf)
SHELF-MOVER1-LOCATION	Name of the `location` constraint for the primary parallel data-mover node, such as `C00-MOVER1-LOCATION`

MOVER1	Name of the primary parallel data-mover node, such as `MOVER1`
SHELF-MOVER2-LOCATION	Name of the `location` constraint for the secondary parallel data-mover node, such as `C00-MOVER2-LOCATION`
MOVER2	Name of the secondary parallel data-mover node, such as `MOVER2`
SHELF-CXFS-COLOCATION	Name of the `colocation` constraint, such as `C00-CXFS-COLOCATION`
CXFS-CLIENT-CLONE	Name of the `clone`, such as `CXFS-CLIENT-CLONE` (see "`cxfs-client-clone` Template" on page 249)
SHELF-CXFS-ORDER	Name of the `order` constraint, such as `C00-CXFS-ORDER`

Note: The `copan_ov_client` template contains two sections, one for a COPAN MAID OpenVault client HA service and another for a DMF HA service – you must fill in the values in the section for the service that you want to configure and delete the other section entirely.

For example, for a **DMF HA service**:

```
primitive C16 ocf:sgi:copan_ov_client \
   op monitor interval="0" timeout="60s" \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   params shelf_name="C16" \
   meta resource-stickiness="1" migration-threshold="1"
```

Note: The `primitive` instance for the COPAN MAID OpenVault client HA service requires `location`, `colocation`, and `order` constraints and a high value for `resource-stickiness` indicates that it will be unlikely to move to the other node, whereas the `primitive` instance for the DMF HA service requires no constraints and has a `resource-stickiness` value that matches the rest of the resources in the `DMF-GROUP` resource (see "`dmf-group` Template" on page 268).

You should configure an instance of this resource for each shelf that will be owned by the active DMF server. (Shelves that are owned by a parallel data-mover node are not included in the DMF HA service.)

3. Verify that the `timeout` values are appropriate for your site.
4. Delete the section of the template that does not apply to your configuration.
5. Verify that there are no comments in *workfile*.
6. Save *workfile*.
7. Update the database with the new resource:

   ```
   node1# crm configure load update workfile
   ```

8. Test the new resource. For example, using shelf ID `C00`:

   a. Verify that shelf `C00` becomes available after a few minutes on `node1`:

```
node1# ov_stat -L C00 -D 'C00.*'
Library Name   Broken   Disabled   State     LCP State
C00            false    false      ready     ready


DriveName Group      Usable Access Disabled  SoftState HardState Occupied PCL
C00d00    dg_c00     true   true   false     ready     unloaded   false
C00d01    dg_c00     true   true   false     ready     unloaded   false
C00d02    dg_c00     true   true   false     ready     unloaded   false
C00d03    dg_c00     true   true   false     ready     unloaded   false
C00d04    dg_c00     true   true   false     ready     unloaded   false
C00d05    dg_c00     true   true   false     ready     unloaded   false
C00d06    dg_c00     true   true   false     ready     unloaded   false
```

b. Move the resource group containing the `copan_ov_client` resource from `node1` to `node2`:

```
node1# crm resource move DMF-GROUP node2
```

c. Verify the status:

```
node1# crm status inactive
```

d. Verify that shelf `C00` becomes available after a few minutes on `node2`:

```
node2# ov_stat -L C00 -D 'C00.*'
Library Name         Broken   Disabled   State     LCP State
C00                  false    false      ready     ready

DriveName Group      Usable Access Disabled  SoftState HardState Occupied PCL
C00d00    dg_c00     true   true   false     ready     unloaded  false
C00d01    dg_c00     true   true   false     ready     unloaded  false
C00d02    dg_c00     true   true   false     ready     unloaded  false
C00d03    dg_c00     true   true   false     ready     unloaded  false
C00d04    dg_c00     true   true   false     ready     unloaded  false
C00d05    dg_c00     true   true   false     ready     unloaded  false
C00d06    dg_c00     true   true   false     ready     unloaded  false
```

e. Move the resource group containing the `copan_ov_client` resource back to `node1`:

```
node1# crm resource move DMF-GROUP node1
```

f. Verify the status:

```
node1# crm status inactive
```

g. Verify that shelf `C00` becomes available after a few minutes on `node1`:

```
node1# ov_stat -L C00 -D 'C00.*'
Library Name         Broken   Disabled   State     LCP State
C00                  false    false      ready     ready


DriveName Group      Usable Access Disabled  SoftState HardState Occupied PCL
C00d00    dg_c00     true   true   false     ready     unloaded  false
C00d01    dg_c00     true   true   false     ready     unloaded  false
C00d02    dg_c00     true   true   false     ready     unloaded  false
C00d03    dg_c00     true   true   false     ready     unloaded  false
```

```
C00d04    dg_c00     true   true   false     ready     unloaded  false
C00d05    dg_c00     true   true   false     ready     unloaded  false
C00d06    dg_c00     true   true   false     ready     unloaded  false
```

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

```
node1# crm resource unmove DMF-GROUP
```

Add Any OpenVault Clients that are Not DMF Servers *(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 the server hostname, specify the `virtualhost` value from the workfile in step 3 above.

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:

1. 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
   ```

2. On the OpenVault client node, use `ov_admin` to enable the node to issue administrative commands by entering the `virtualhost` value in *workfile*, the port number, and security key as needed:

```
node2# ov_admin
...
Name where the OpenVault server is listening? [virtualhostvalue]
What port number is the OpenVault server on virtualhostvalue using? [695]
What security key is used for admin commands on the HA OpenVault servers? [none]
```

Test the openvault Resource

Test the new openvault resource:

1. Verify that the OpenVault libraries and drives become available after a few minutes on node1. Using the -ldxy options to the ov_stat command, the only output you should see is that of the inactive LCPs and DCPs that apply to the passive server. For example:

```
node1# ov_stat -ldxy
LCPName     LibraryName Usable SoftState Disabled MsgLevel
sl48a@node2 sl48        false  inactive  false    debug
sl48b@node2 sl48        false  inactive  false    debug

DCPName       DriveName Usable SoftState Disabled MsgLevel
sl48_1a@node2 sl48_1    false  inactive  false    debug
sl48_1b@node2 sl48_1    false  inactive  false    debug
```

   In the above output, the LCPs and DCPs for node2 are expected to be inactive because node2 is the passive server. Therefore, the above output indicates that all is well.

   If you prefer to see all of the LCPs and DCPs regardless of state, do not include the -y option. For example:

```
node1# ov_stat -ldxy
LCPName     LibraryName Usable SoftState    Disabled MsgLevel
sl48a@node1 sl48        true   ready        false    debug
sl48a@node2 sl48        false  inactive     false    debug
sl48b@node1 sl48        true   disconnected false    debug
sl48b@node2 sl48        false  inactive     false    debug

DCPName       DriveName Usable SoftState    Disabled MsgLevel
sl48_1a@node1 sl48_1    true   ready        false    debug
sl48_1a@node2 sl48_1    false  inactive     false    debug
sl48_1b@node1 sl48_1    true   disconnected false    debug
sl48_1b@node2 sl48_1    false  inactive     false    debug
```

   Note: If multiple instances of an LCP/DCP exist on the active server, exactly one will be listed with a SoftState of ready and the others will be listed as disconnected. The particular instance listed as ready may change over time and is not important.

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

   ```
   node1# crm resource move DMF-GROUP node2
   ```

3. Verify the status:

   ```
   node1# crm status inactive
   ```

4. Verify that all of the drives become available after a few moments on `node2`. Using the `-ldxy` options to the `ov_stat` command, the only output you should see is that of the inactive LCPs and DCPs that apply to the passive server (now `node1`). For example:

```
node2# ov_stat -ldxy
LCPName      LibraryName Usable SoftState Disabled MsgLevel
sl48a@node1 sl48        false  inactive  false    debug
sl48b@node1 sl48        false  inactive  false    debug

DCPName       DriveName Usable SoftState Disabled MsgLevel
sl48_1a@node1 sl48_1    false  inactive  false    debug
sl48_1b@node1 sl48_1    false  inactive  false    debug
```

   In the above output, all is well because the only output displayed are the inactive LCPs and DCPs that are connected to what is now the passive DMF server (`node1`).

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

   ```
   node1# crm resource move DMF-GROUP node1
   ```

6. Verify the status:

   ```
   node1# crm status inactive
   ```

7. Verify that all of the drives become available after a few moments on `node1`. Using the `-ldxy` options to the `ov_stat` command, the only output you should see is that of the inactive LCPs and DCPs that apply to the passive server (now back to `node2`). For example:

```
node1# ov_stat -ldxy
LCPName      LibraryName Usable SoftState Disabled MsgLevel
sl48a@node2 sl48        false  inactive  false    debug
sl48b@node2 sl48        false  inactive  false    debug

DCPName       DriveName Usable SoftState Disabled MsgLevel
sl48_1a@node2 sl48_1    false  inactive  false    debug
sl48_1b@node2 sl48_1    false  inactive  false    debug
```

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

```
node1# crm resource unmove DMF-GROUP
```

Add the dmf Resource

Figure 6-5 shows the step to add the resource for DMF.

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*.

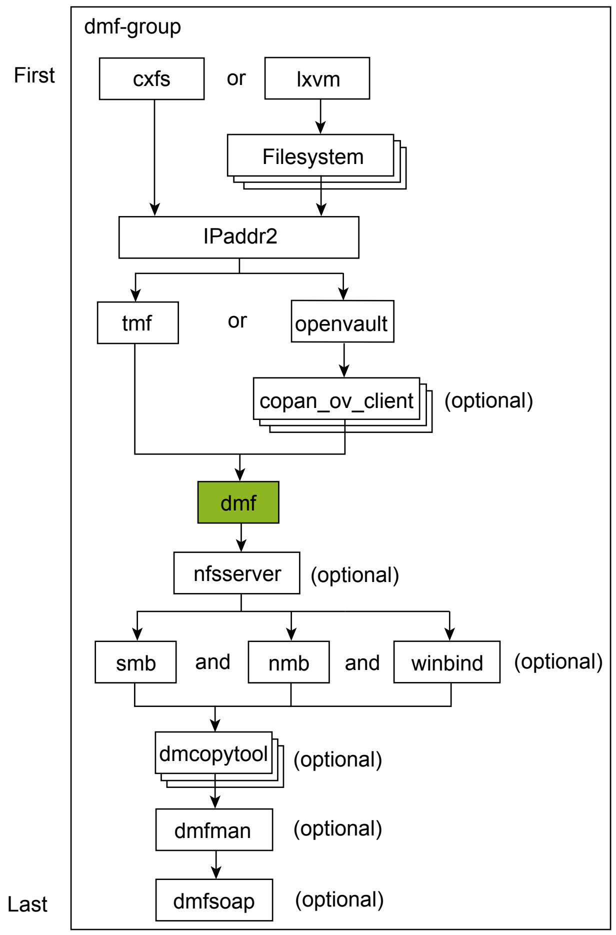

Figure 6-5 Adding the Resource for DMF

Do the following:

1. Make the filesystem backup inventory accessible from all DMF servers in the HA cluster.

   The backup of 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 the directory specified by the DMF `HOME_DIR` configuration parameter, such as `/dmf/home/site_specific`:

- On `node1` (which currently contains 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`.

- (*OpenVault only*) 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.
- (*DMF Parallel Data-Mover Option only*) Create `node` objects for each potential DMF server and set the `INTERFACE` parameter in those `node` objects to the same virtual hostname used for `SERVER_NAME` in the `base` object.

  Note: If you are not using the DMF Parallel Data-Mover Option, then no `node` objects are needed.

For more information, see the `dmf.conf`(5) man page and the *DMF 6 Administrator Guide.*

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 specified by the `HOME_DIR` parameter in the DMF configuration file. 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 the first field of the lines containing “`dmf`” with the OpenVault virtual hostname. For example, if the `virtualhost` value in the *workfile* is `myvirtualhost`, the result would be:

```
myvirtualhost   dmf   *  CAPI none
myvirtualhost   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. Create another *workfile* that contains the following:

   `group DMF-GROUP` *Previously_Added_Resources* `DMF`

   For more information about the group definition, see "`dmf-group` Template" on page 268.

6. Copy the `primitive` in the `dmf` template into *workfile* and replace the site-specific variables as directed. The template is located in:

   `/usr/share/doc/sgi-ha/templates/dmf`

   See "Use the Templates as Building Blocks" on page 18.

   A `dmf` resource starts/stops DMF. It is used in a DMF HA service.

   Use the following:

```
primitive DMF ocf:sgi:dmf \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   params monitor_level="0" \
   meta resource-stickiness="1" migration-threshold="1"
```

Variable	Description
DMF	Name of this resource instance, such as `DMF`

Note: In a CXFS environment, ensure that the `timeout` values are appropriate for your site; you must account for the time required to relocate the CXFS metadata server for the managed filesystems.

For example:

```
primitive DMF ocf:sgi:dmf \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   params monitor_level="0" \
   meta resource-stickiness="1" migration-threshold="1"
```

7. Verify that the `timeout` values are appropriate for your site.
8. Verify that there are no comments in *workfile*.
9. Save *workfile*.
10. Update the database with the new resource:

    ```
    node1# crm configure load update workfile
    ```

11. Test the resource:

    a. 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
    ```

    b. Move the resource group containing the `dmf` resource to `node2`:

    ```
    node1# crm resource move DMF-GROUP node2
    ```

    c. Verify the status:

    ```
    node1# crm status inactive
    ```

    d. 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
```

e. Move the resource group containing the `dmf` resource back to `node1`:

```
node1# crm resource move DMF-GROUP node1
```

f. Verify the status:

```
node1# crm status inactive
```

g. 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
```

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

```
node1# crm resource unmove DMF-GROUP
```

Add the `nfsserver` Resource *(Optional)*

Figure 6-6 shows the step to add the optional resource for the NFS server.

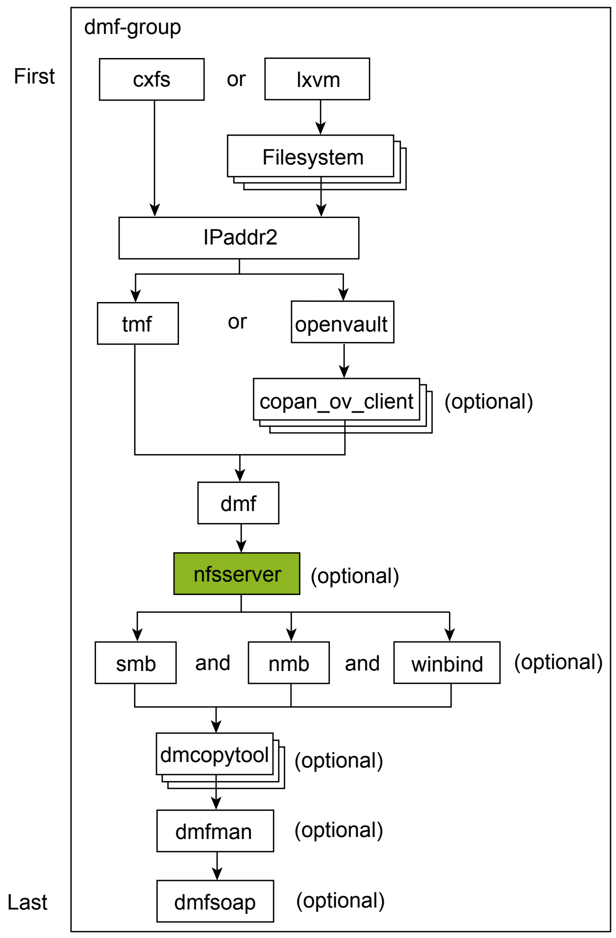

Figure 6-6 Adding the Resource for the NFS Server

To configure NFS for an HA environment, 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=`*uniquenumber* export option in order to prevent stale file handles after failover.

2. *(RHEL only)* On both RHEL nodes, do the following:

   a. Set the following in the `/etc/sysconfig/nfs` file:

   ```
   START_SMNOTIFY="no"
   ```

   b. Ensure that NFS lock services are started at boot time:

   - On RHEL `node1`:

     ```
     node1# chkconfig nfslock on
     ```

   - On RHEL `node2`:

     ```
     node2# chkconfig nfslock on
     ```

3. Create another *workfile* that contains the following:

   ```
   group DMF-GROUP Previously_Added_Resources NFS
   ```

   For more information about the group definition, see "`dmf-group` Template" on page 268.

4. Copy the `primitive` in the `nfsserver` template into *workfile* and replace the site-specific variables as directed. The template is located in:

   ```
   /usr/share/doc/sgi-ha/templates/nfsserver
   ```

   See "Use the Templates as Building Blocks" on page 18.

   An `nfsserver` resource controls the start/stop of NFS. It may be used in a DMF HA service.

   Use the following:

   ```
   primitive NFS ocf:heartbeat:nfsserver \
      op monitor interval="120s" timeout="60s" on-fail="restart" \
      op monitor interval="0" timeout="60s" \
      op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
      op stop interval="0" timeout="120s" on-fail="fence" \
      params nfs_shared_infodir="STATEDIR" nfs_ip="IP ADDRESS-ALIAS" \
      meta resource-stickiness="1" migration-threshold="1"
   ```

Variable	Description
NFS	Name of this resource instance, such as `NFS`.
STATEDIR	Directory in the NFS filesystem that will be used to store NFS file-lock state for this HA cluster, such as the `/mnt/cxfsvol1/.nfs` subdirectory in the exported filesystem.
IP ADDRESS-ALIAS	IP address 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 this `nfsserver` resource), for example `128.162.244.244`. This IP address alias will be the same IP address specified for *IP-ADDRESS* in one of the `IPaddr2` resources (see "`IPaddr2` Template" on page 283).

For example:

```
primitive NFS ocf:heartbeat:nfsserver \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   params nfs_shared_infodir="/mnt/cxfsvol1/.nfs" nfs_ip="128.162.232.79" \
   meta resource-stickiness="1" migration-threshold="1"
```

5. Verify that the `timeout` values are appropriate for your site.
6. Verify that there are no comments in *workfile*.
7. Save *workfile*.
8. Update the database with the new resource:

   ```
   node1# crm configure load update workfile
   ```

9. Test the new resource:

   a. 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)
```

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

```
otherhost# mount node1:/nfsexportedfilesystem /mnt/test
```

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

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

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

```
node1# crm resource move DMF-GROUP node2
```

e. Verify the status:

```
node1# crm status inactive
```

f. 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)
```

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

```
otherhost# echo "more test data" > /mnt/test/testFile1B
otherhost# cat /mnt/test/testFile1B
more test data
```

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

```
node1# crm resource move DMF-GROUP node1
```

i. Verify the status:

```
node1# crm status inactive
```

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

```
node1# crm resource unmove DMF-GROUP
```

Add the Samba Resources: `smb`, `nmb`, and `winbind` *(Optional)*

Figure 6-7 shows the step to add the resources associated with Samba, which are optional.

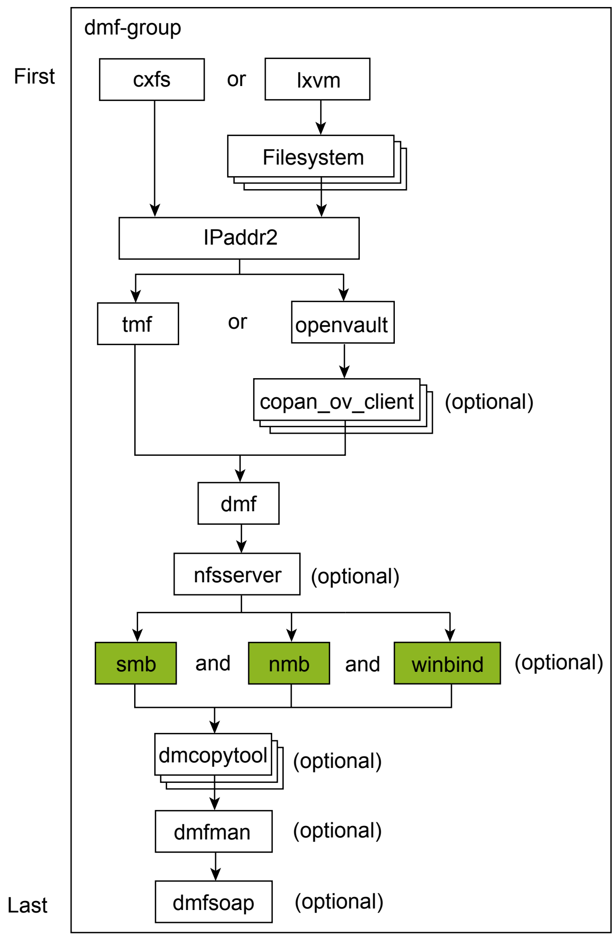

Figure 6-7 Adding the Resources for Samba

To configure Samba for an HA environment, do the following:

1. If you are using `winbind`, ensure that the HA nodes do not use pluggable authentication modules (PAM) or name service switch (NSS) that could have dependencies on `winbind`. That is, there must not be dependencies on `winbind` in the files in the `/etc/pam.d` directory or the files listed in the `/etc/nsswitch.conf` file.

Note: Not all configurations with `winbind` will work in an HA cluster.

2. Copy the Samba directories to a shared location. For example:

   ```
   node1# cp -r /etc/samba /mnt/data/.ha/etc_samba
   node1# cp -r /var/lib/samba /mnt/data/.ha/var_lib_samba
   ```

3. Create a `Filesystem` resource for each of the new Samba directories:

   a. Create another *workfile* that contains the following, where *PREVIOUSLY_ADDED_RESOURCES* are all of the resources added to this point in the procedure and *FILESYSTEM* is one of the filesystems to be controlled by HA processes (use a unique ID for each `Filesystem` instance):

      ```
      group DMF-GROUP PREVIOUSLY_ADDED_RESOURCES  FILESYSTEM
      ```

   b. Copy the `primitive` from the `Filesystem` template into *workfile* and replace the site-specific variables as directed in the template comments or in "`Filesystem` Template" on page 280. The template is located in:

      ```
      /usr/share/doc/sgi-ha/templates/Filesystem
      ```

      See "Use the Templates as Building Blocks" on page 18.

      Note: The `Filesystem` resource for Samba requires `fstype="none"`, which differs from the template, and `options="bind"`.

   c. Verify that the `timeout` values are appropriate for your site.

   d. Verify that there are no comments in *workfile*.

   e. Save *workfile*.

   f. Update the database with the new resource:

      ```
      node1# crm configure load update workfile
      ```

   g. Test the `Filesystem` resource:

      i. 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.

---

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

```
node1# df -hl
```

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

```
node1# crm resource move DMF-GROUP node2
```

iv. Verify the status:

```
node1# crm status inactive
```

v. 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.

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

```
node1# crm resource move DMF-GROUP node1
```

vii. Verify the status:

```
node1# crm status inactive
```

viii. 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.

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

```
node1# crm resource unmove DMF-GROUP
```

h. Repeat steps a–g for each filesystem to be managed by HA in a local XVM environment. Use a unique instance ID for each filesystem.

For example:

- For the `/mnt/data/.ha/etc_samba` directory, you could use a `primitive` ID of `etc_samba`, a `device` value of `/mnt/data/.ha/etc_samba`, and a `directory` value of `/etc/samba`. In this case, the `group` definition would be:

  group DMF-GROUP *Previously_Added_Resources* etc_samba

- For the `/mnt/data/.ha/var_lib_samba` directory, you could use a `primitive` ID of `var_lib_samba`, a `device` value of `/mnt/data/.ha/var_lib_samba`, and a `directory` value of `/var/lib/samba`. In this case, the `group` definition would be:

  group DMF-GROUP *Previously_Added_Resources* var_lib_samba

  After you have added all of the filesystems, the `DMF-GROUP` definition could appear something like the following:

```
group DMF-GROUP LXVM dmfusr1 dmfusr2 home spool move_fs journals \
                tmp move_fs cache store
```

4. Create another *workfile* that contains the following:

   group DMF-GROUP *Previously_Added_Resources* SMB NMB

   If your Samba implementation uses an authentication type that requires the `winbind` daemon, use the following:

   group DMF-GROUP *Previously_Added_Resources* SMB NMB WINBIND

5. Copy the `primitive` in the `smb` and `nmb` templates and optionally the `winbind` template into *workfile* and replace the site-specific variables as directed:

   • `smb`:

     An `smb` resource controls the `smb` service for Samba. It may be used in a DMF HA service. The template is located in:

     `/usr/share/doc/sgi-ha/templates/smb`

     See "Use the Templates as Building Blocks" on page 18.

     Use the following:

```
primitive SMB lsb:smb \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="60s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="60s" on-fail="fence" \
   meta resource-stickiness="1" migration-threshold="1"
```

Variable	**Description**
SMB	Name of this resource instance, such as `SMB`.

     For example:

```
primitive SMB lsb:smb \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="60s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="60s" on-fail="fence" \
   meta resource-stickiness="1" migration-threshold="1"
```

   • `nmb`:

     An `nmb` resource controls the `nmbd` service for Samba. It may be used in a DMF HA service. The template is located in:

     `/usr/share/doc/sgi-ha/templates/nmb`

     See "Use the Templates as Building Blocks" on page 18.

     Use the following:

```
primitive NMB lsb:nmb \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="60s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="60s" on-fail="fence" \
   meta resource-stickiness="1" migration-threshold="1"
```

Variable	**Description**
NMB	Name of this resource instance, such as `NMB`.

For example:

```
primitive NMB lsb:nmb \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="60s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="60s" on-fail="fence" \
   meta resource-stickiness="1" migration-threshold="1"
```

- `winbind`:

  This template is located in:

  `/usr/share/doc/sgi-ha/templates/winbind`

  See "Use the Templates as Building Blocks" on page 18.

  If your Samba implementation uses an authentication type that requires the `winbind` daemon, you can use a `winbind` resource to control it. It may be used in a DMF HA service.

  Use the following:

```
primitive WINBIND lsb:winbind \
   op monitor interval="0" timeout="60s" \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="60s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="60s" on-fail="fence" \
   meta resource-stickiness="1" migration-threshold="1"
```

Variable	Description
WINBIND	Name of this resource instance, such as WINBIND

For example:

```
primitive WINBIND lsb:winbind \
   op monitor interval="0" timeout="60s" \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="60s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="60s" on-fail="fence" \
   meta resource-stickiness="1" migration-threshold="1"
```

.

6. Verify that the `timeout` values are appropriate for your site.
7. Verify that there are no comments in *workfile*.
8. Save *workfile*.
9. Update the database with the new resources:

   ```
   node1# crm configure load update workfile
   ```

10. Test the new resources:

    a. Ensure that the resource group containing the `smb` and `nmb` resources and optionally the `winbind` resource is on `node1`:

    ```
    node1# crm resource move DMF-GROUP node1
    ```

    b. Verify the status:

    ```
    node1# crm status inactive
    ```

    c. 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.

d. Move the resource group containing the smb and nmb resources from node1 to node2:

```
node1# crm resource move DMF-GROUP node2
```

e. Verify the status:

```
node1# crm status inactive
```

f. 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
```

g. Move the resource group containing the smb and nmb resources back to node1:

```
node1# crm resource move DMF-GROUP node1
```

h. Verify the status:

```
node1# crm status inactive
```

i. Remove the implicit location constraints imposed by the administrative move command executed above:

```
node1# crm resource unmove DMF-GROUP
```

Add the dmcopytool Resources *(Optional)*

Figure 6-8 shows the step to add an optional dmcopytool resource for the DMF copytool for Lustre.

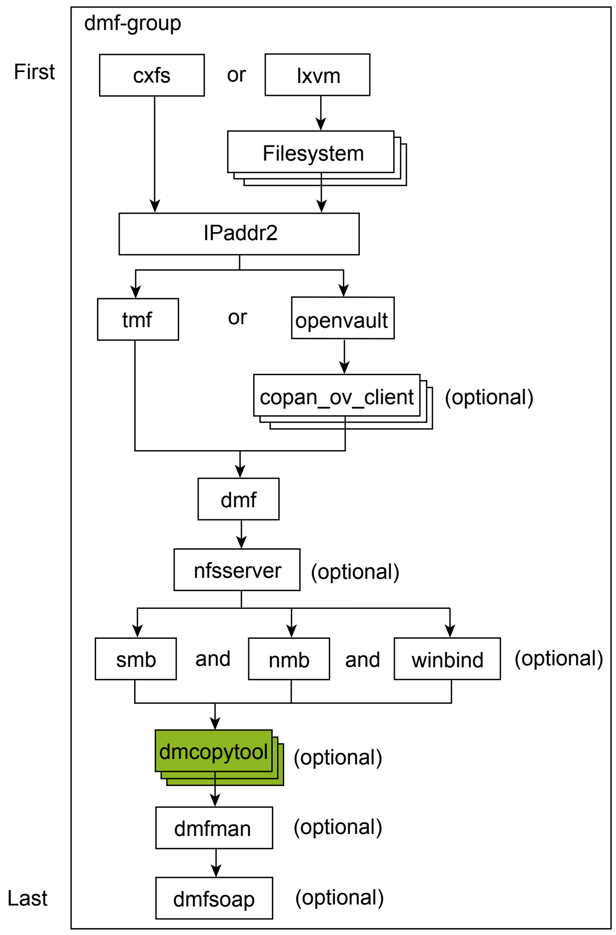

Figure 6-8 Adding the Resource for the DMF Copytool

To configure a DMF copytool resource for an HA environment, do the following:

1. Create another *workfile* that contains the following:

   `group DMF-GROUP` *Previously_Addeded_Resources* `DMCOPYTOOL`

2. Copy the `primitive` in the `dmcopytool` template into *workfile* and replace the site-specific variables as directed. The template is located in:

   `/usr/share/doc/sgi-ha/templates/dmcopytool`

   See "Use the Templates as Building Blocks" on page 18.

   A `dmcopytool` resource controls the SGI DMF copytool (`lhsmtool_dmf`), which is a version of the Lustre copytool (`lhsmtool_posix`) that has been tuned to work efficiently with the DMF archive system. Its primary function is to copy files between the Lustre filesystem and the DMF archive system. It may optionally be used in a DMF HA service.

   Use the following:

```
primitive DMCOPYTOOL ocf:sgi:dmcopytool \
  op monitor interval="0" timeout="60s" \
  op monitor interval="120s" timeout="60s" on-fail="restart" \
  op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
  op stop interval="0" timeout="600s" on-fail="fence" \
  params archive="NUM" hsmroot="PATH" lustredevice="DEVICE" \
   dmctmount="DMCT-MOUNT" dmfmount="DMF-MOUNT" \
  meta resource-stickiness="1" migration-threshold="1"
```

Variable	**Description**
DMCOPYTOOL	Name of this resource instance, such as `DMCOPYTOOL`
NUM	The archive number registered with the Lustre HSM coordinator for this instance of the copytool. Each Lustre filesystem using the DMF copytool should be given its own unique archive number.
PATH	Path of a directory within a filesystem managed by DMF that is used for archiving or restoring Lustre files. This directory must already exist.
DEVICE	The Lustre filesystem device to be mounted for use by the DMF copytool.
DMCT-MOUNT	The Lustre filesystem mount point that will be used by the DMF copytool.

DMF-MOUNT	The Lustre filesystem mount point that will be used by DMF. This mount point must match the name of a `filesystem` object defined in the DMF configuration file with a `MIGRATION_LEVEL` parameter set to `archive`.

For example:

```
primitive DMCOPYTOOL ocf:sgi:dmcopytool \
  op monitor interval="0" timeout="60s" \
  op monitor interval="120s" timeout="60s" on-fail="restart" \
  op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
  op stop interval="0" timeout="600s" on-fail="fence" \
  params archive="2" hsmroot="/usr3" lustredevice="128.162.245.79@tcp0:/lustrefs" \
  dmctmount="/lfs_dmct" dmfmount="/lfs_dmf" \
  meta resource-stickiness="1" migration-threshold="1"
```

3. Verify that the `timeout` values are appropriate for your site.
4. Add any other parameters as required by your site (see below).
5. Verify that there are no comments in *workfile*.
6. Save *workfile*.
7. Update the database with the new resource:

   node1# **crm configure load update *workfile***

8. Test the new resource:

   a. Point your browser at `https://`*virtualIPaddress*`:11110/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*.

   b. Move the resource group containing the `dmfsoap` resource from `node1` to `node2`:

      node1# **crm resource move DMF-GROUP node2**

   c. Verify the status:

      node1# **crm status inactive**

   d. Repeat step 8a to verify that the DMF copytool is still available.

e. Move the resource group containing the `dmcopytool` resource back to `node1`:

```
node1# crm resource move DMF-GROUP node1
```

f. Verify the status:

```
node1# crm status inactive
```

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

```
node1# crm resource unmove DMF-GROUP
```

Additional parameters, which correspond to options on the `lhsmtool_dmf`(8) man page:

- `mountoptions` specifies a comma-separated list of mount options for the Lustre filesystem used by the DMF copytool.
- `loglevel` specifies the logging filter level. The level value can be any of the following, shown in ascending verbosity:

  `off`
  `fatal`
  `error`
  `warning`
  `normal`
  `information` (default)
  `debug`

- `interval` specifies the time, in seconds, between progress reports sent to the Lustre HSM coordinator. The default is `30`. (For most DMF environments, you should use the default.)
- `bandwidth` specifies the overall I/O bandwidth limit. By default, bandwidth is specified in mebibytes per second. A value of `0` disables the bandwidth limit. The default is `0`. This value is ignored for `t2` (tier 2) storage.
- `chunksize` specifies the default transfer buffer size. By default, size is specified in mebibytes For best performance, this value should match the default stripe size for Lustre files. The default is `1`. This value is ignored for Tier 2 copying. This value is ignored for `t2` storage.

Add the `dmfman` Resource *(Optional)*

Figure 6-9 shows the step to add the optional `dmfman` resource for DMF Manager.

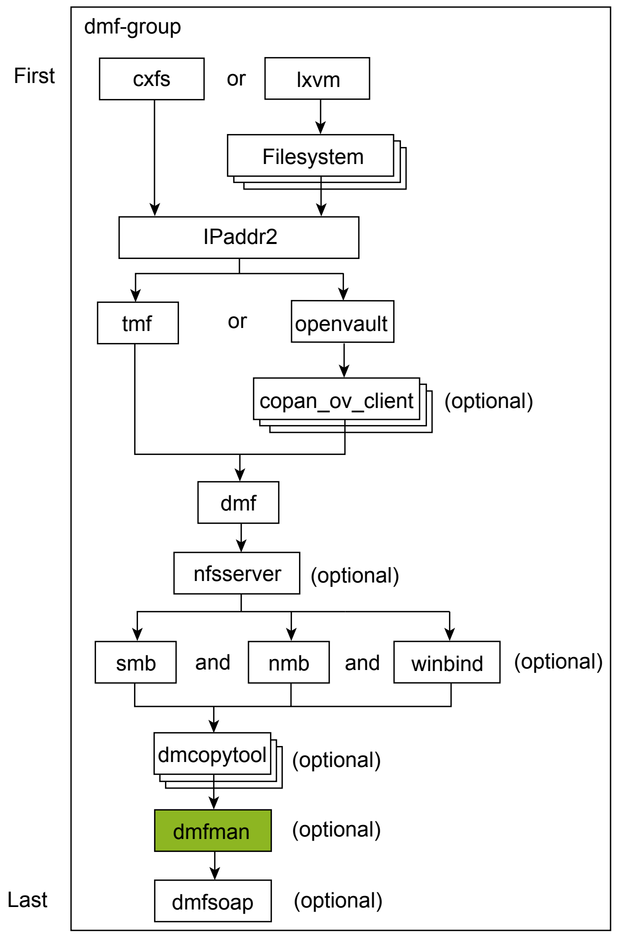

Figure 6-9 Adding the Resource for DMF Manager

To configure DMF Manager for an HA environment, do the following:

1. Run the `dmfman_setup_ha` script to create the required links and directories in a commonly accessible filesystem (such as the directory specified by the `HOME_DIR`

parameter in the DMF configuration file) that will allow DMF statistics archives to be accessible across the HA cluster:

- On `node1`:

  ```
  node1# /usr/lib/dmf/dmfman_setup_ha -d HOME_DIR
  ```

- On `node2`:

  ```
  node2# /usr/lib/dmf/dmfman_setup_ha -d HOME_DIR
  ```

For example, if the `HOME_DIR` parameter is set to `/dmf/home` in `/etc/dmf/dmf.conf`, you would enter the following:

- On `node1`:

  ```
  node1# /usr/lib/dmf/dmfman_setup_ha -d /dmf/home
  ```

- On `node2`:

  ```
  node2# /usr/lib/dmf/dmfman_setup_ha -d /dmf/home
  ```

2. Create another *workfile* that contains the following:

   ```
   group DMF-GROUP Previously_Added_Resources DMFMAN
   ```

   For more information about the group definition, see "`dmf-group` Template" on page 268.

3. Copy the `primitive` in the `dmfman` template to *workfile* and replace the site-specific variables as directed. The template is located in:

   ```
   /usr/share/doc/sgi-ha/templates/dmfman
   ```

   See "Use the Templates as Building Blocks" on page 18.

   A `dmfman` resource controls DMF Manager. It may be used in a DMF HA service.

   Use the following:

   ```
   primitive DMFMAN ocf:sgi:dmfman \
      op monitor interval="0" timeout="60s" \
      op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
      op stop interval="0" timeout="120s" on-fail="fence" \
      meta resource-stickiness="1" migration-threshold="100"
   ```

Variable	Description
DMFMAN	Name of this resource instance, such as `DMFMAN`

For example:

```
primitive DMFMAN ocf:sgi:dmfman \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   meta resource-stickiness="1" migration-threshold="100"
```

4. Verify that the `timeout` values are appropriate for your site.
5. Verify that there are no comments in *workfile*.
6. Save *workfile*.
7. Update the database with the new resource:

   ```
   node1# crm configure load update workfile
   ```

8. Test the new resource:

   a. Point your browser at `https://`*virtualIPaddress*`:11109` 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*.

   b. Move the resource group containing the `dmfman` resource from `node1` to `node2`:

   ```
   node1# crm resource move DMF-GROUP node2
   ```

   c. Verify the status:

   ```
   node1# crm status inactive
   ```

   d. Repeat step 8a to verify that DMF Manager is still available.

   e. Move the resource group containing the `dmfman` resource back to `node1`:

   ```
   node1# crm resource move DMF-GROUP node1
   ```

   f. Verify the status:

   ```
   node1# crm status inactive
   ```

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

```
node1# crm resource unmove DMF-GROUP
```

Add the `dmfsoap` Resource *(Optional)*

Figure 6-10 shows the step to add the optional `dmfsoap` resource for the DMF SOAP client.

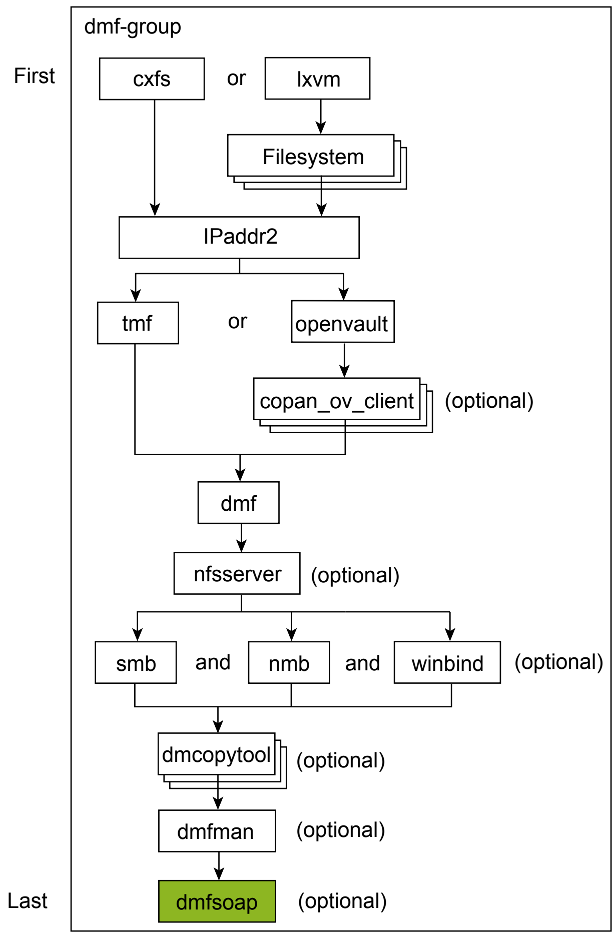

Figure 6-10 Adding the Resource for DMF Manager

To configure the DMF client SOAP service for an HA environment, do the following:

1. Create another *workfile* that contains the following:

   ```
   group DMF-GROUP Previously_Addeded_Resources DMFSOAP
   ```

2. Copy the `primitive` in the `dmfsoap` template into *workfile* and replace the site-specific variables as directed. The template is located in:

   ```
   /usr/share/doc/sgi-ha/templates/dmfsoap
   ```

   See "Use the Templates as Building Blocks" on page 18.

   A `dmfsoap` resource controls the DMF client Simple Object Access Protocol (SOAP) service. It may optionally be used in a DMF HA service.

   Use the following:

   ```
   primitive DMFSOAP ocf:sgi:dmfsoap \
      op monitor interval="0" timeout="60s" \
      op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
      op stop interval="0" timeout="120s" on-fail="fence" \
      meta resource-stickiness="1" migration-threshold="100"
   ```

Variable	**Description**
DMFSOAP	Name of this resource instance, such as `DMFSOAP`

   For example:

   ```
   primitive DMFSOAP ocf:sgi:dmfsoap \
      op monitor interval="0" timeout="60s" \
      op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
      op stop interval="0" timeout="120s" on-fail="fence" \
      meta resource-stickiness="1" migration-threshold="100"
   ```

3. Verify that the `timeout` values are appropriate for your site.

4. Verify that there are no comments in *workfile*.

5. Save *workfile*.

6. Update the database with the new resource:

   ```
   node1# crm configure load update workfile
   ```

7. Test the new resource:

a. Point your browser at `https://`*virtualIPaddress*`:11110/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*.

b. Move the resource group containing the `dmfsoap` resource from `node1` to `node2`:

```
node1# crm resource move DMF-GROUP node2
```

c. Verify the status:

```
node1# crm status inactive
```

d. Repeat step 7a to verify that the DMF client SOAP service is still available.

e. Move the resource group containing the `dmfsoap` resource back to `node1`:

```
node1# crm resource move DMF-GROUP node1
```

f. Verify the status:

```
node1# crm status inactive
```

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

```
node1# crm resource unmove DMF-GROUP
```

Confirm the Completed Status

Use the `status` command to confirm the resulting HA cluster:

```
node1# crm status inactive
```

Put the DMF HA Service into Production Mode

See Chapter 8, "Create the Fencing Capability and Put Into Production Mode" on page 185 to complete the process.

Chapter 7

Create the COPAN MAID OpenVault Client HA Service for Mover Nodes

This chapter provides an overview of the following:

- "Understand the Requirements for the COPAN MAID OpenVault Client HA Service" on page 165
- "Configure and Test the Service Components Before Applying an COPAN MAID HA Environment" on page 167
- "Ensure that the Base HA Cluster is Operational" on page 168
- "Stop the `cxfs_client` and `openvault` Services Before Applying an HA Environment" on page 168
- "Disable the Parallel Data-Mover Nodes and the Services" on page 169
- "Map of Resources for the COPAN MAID OpenVault HA Service" on page 170
- "Create the OpenVault Components on the Failover Node" on page 170
- "Create and Test the HA Service" on page 172
- "Confirm the Completed Status" on page 183
- "Put the HA Service into Production Mode" on page 184

Understand the Requirements for the COPAN MAID OpenVault Client HA Service

Before applying an HA environment, you must understand the requirements of the individual components:

COPAN MAID Requirements for an OpenVault Client HA Service

This section discusses the following:

- "COPAN MAID Requirements in Any HA Cluster" on page 166

- "COPAN MAID Requirements in a Mover-Node HA Cluster" on page 166

COPAN MAID Requirements 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 Requirements in a Mover-Node HA Cluster

In addition to the requirements listed in "COPAN MAID Requirements in Any HA Cluster" on page 166, 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 170.

Configure and Test the Service Components Before Applying an COPAN MAID HA Environment

Configure and test the service components **before applying an HA environment**, as described in the following:

- "Configure and Test the CXFS Client Before Applying a COPAN MAID HA Environment" on page 167
- "Test the COPAN MAID OpenVault Client Before Applying a COPAN MAID HA Environment" on page 167

Configure and Test the CXFS Client Before Applying a COPAN MAID HA Environment

Before applying an HA environment, configure CXFS according to the instructions in *CXFS 7 Administrator Guide for SGI InfiniteStorage* and *CXFS 7 Client-Only Guide for SGI InfiniteStorage*. To test, do the following:

1. Ensure that the node is a member of the CXFS cluster. For example:

   ```
   node1# /usr/cluster/bin/cxfs_info
   ```

2. Ensure that the CXFS client service is running:

   ```
   node1# service cxfs-client status
   ```

   If it is not running, start it:

   ```
   node1# service cxfs-client start
   ```

3. Ensure that filesystems are mounted. For example:

   ```
   node1# df
   ```

Test the COPAN MAID OpenVault Client Before Applying a COPAN MAID HA Environment

Configure the following OpenVault components for each COPAN MAID shelf by executing the `ov_shelf`(8) command on `node1` (or `mover1`), 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.

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

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

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

Ensure that the Base HA Cluster is Operational

Ensure that the base HA cluster is operational, as described in "Test the Base HA Cluster" on page 45.

Stop the `cxfs_client` and `openvault` Services Before Applying an HA Environment

Do the following on both parallel data-mover nodes to ensure that the non-HA services will be controlled by the COPAN MAID OpenVault Client HA service:

- On `mover1`:

  ```
  mover1# chkconfig cxfs_client off
  mover1# chkconfig openvault off

  mover1# service cxfs_client stop
  mover1# service openvault stop
  ```

- On `mover2`:

  ```
  mover2# chkconfig cxfs_client off
  mover2# chkconfig openvault off

  mover2# service cxfs_client stop
  mover2# service openvault stop
  ```

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 mover1 mover2
   ```

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 `mover1`:

     ```
     mover1# ps -ef | egrep 'dmatrc|dmatwc' | grep -v grep
     mover1#
     ```

   - On `mover2`:

     ```
     mover2# ps -ef | egrep 'dmatrc|dmatwc' | grep -v grep
     mover2#
     ```

   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 the DMF server, determine which CXFS filesystems are mounted:

   ```
   dmfserver# ls /dev/cxvm
   ```

   Save the output from this command for use later when you define the *VOLUME_LIST* for the `volnames` parameter in the `cxfs-client` resource in "Create and Test the HA Service" on page 172.

Map of Resources for the COPAN MAID OpenVault HA Service

Figure 7-1 shows a map of an example configuration process for the OpenVault client service for COPAN MAID shelves in an active/active HA cluster that consists of two parallel data-mover nodes named `mover1` and `mover2`. (`mover1` is the same node as `node1` referred to in Chapter 4, "Create the Base HA Cluster" on page 35.) The map uses the suggested default IDs found in the templates in `/usr/share/doc/sgi-ha/templates`.

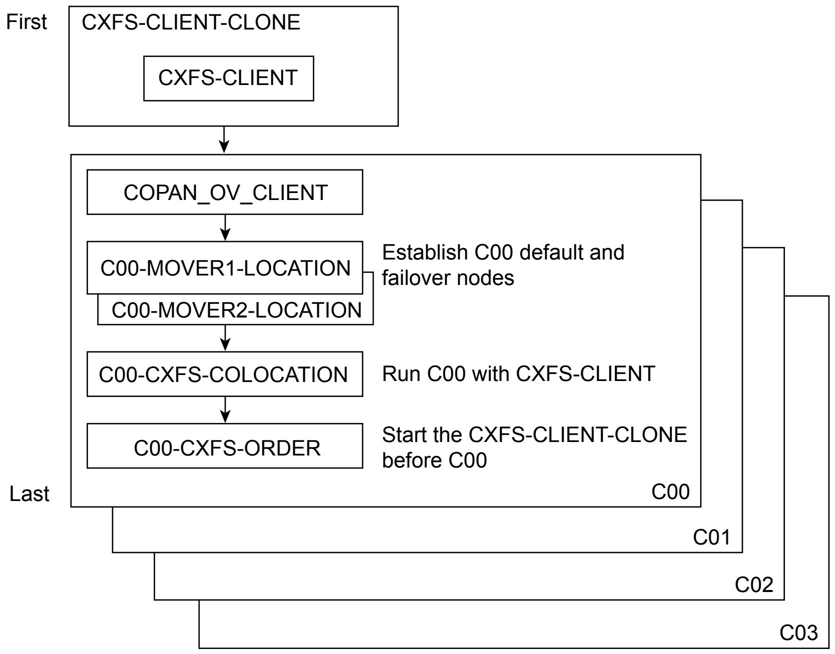

Figure 7-1 Map of Resources for the COPAN MAID OpenVault HA Service

Create the OpenVault Components on the Failover Node

When you configured the non-HA 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 "Configure and Test the COPAN MAID OpenVault Client Before Applying a DMF HA Environment" on page 88).

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: `mover1`
- Failover node: `mover2`

Note: For more information about the shelf identifier, see *COPAN MAID for DMF Quick Start Guide*.

Do the following:

1. On `mover1`:

   a. Export the shelf, hostname, and OCF root environment variables for use by the `copan_ov_client` script:

   ```
   mover1# export OCF_RESKEY_shelf_name=C00
   mover1# export OCF_RESKEY_give_host=mover2
   mover1# export OCF_ROOT=/usr/lib/ocf
   ```

   b. Transfer ownership of the shelf from `mover1` to `mover2`:

   ```
   mover1# /usr/lib/ocf/resource.d/sgi/copan_ov_client give
   ```

2. On `mover2`:

   a. Verify that `mover2` now owns the shelf's XVM volumes (`C00A` through `C00Z`, although not necessarily listed in alphabetical order):

   ```
   mover2# xvm -d local probe | grep C00
   phys/copan_C00M
   phys/copan_C00B
   phys/copan_C00G
   ...
   ```

   b. Create the OpenVault components for `mover2`:

   ```
   mover2# ov_shelf create C00
   ```

Messages such as the following may be seen at this point, but do not indicate an error:

```
Cartridge "C00B00" in the openvault database starts with the string "C00" but its owning library is unknown
Cartridge "C00B01" in the openvault database starts with the string "C00" but its owning library is unknown
Cartridge "C00B02" in the openvault database starts with the string "C00" but its owning library is unknown
```

For more information, see *COPAN MAID for DMF Quick Start Guide*.

c. Stop the newly created LCP and DCPs for the shelf:

```
mover2# ov_stop C00*
```

d. Export the shelf, hostname, and OCF root environment variables for use by the `copan_ov_client` script:

```
mover2# export OCF_RESKEY_shelf_name=C00
mover2# export OCF_RESKEY_give_host=mover1
mover2# export OCF_ROOT=/usr/lib/ocf
```

e. Transfer ownership of the shelf from `mover2` back to `mover1`:

```
mover2# /usr/lib/ocf/resource.d/sgi/copan_ov_client give
```

3. On `mover1`, verify that `mover1` once again owns the shelf's XVM volumes:

```
mover1# 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, `mover1` should be the default node for half of the shelves and `mover2` should be the default node for the remaining shelves.

Create and Test the HA Service

This section discusses the following:

- "Add the Clone" on page 173

- "Add the `cxfs-client` Resource" on page 174
- "Test the Clone" on page 176
- "Add the `copan_ov_client` Resources" on page 178
- "Create the STONITH Capability" on page 181
- "Test the `copan_ov_client` Resource" on page 181

Add the Clone

Figure 7-2 shows the steps to add the resources for the clone and the CXFS client.

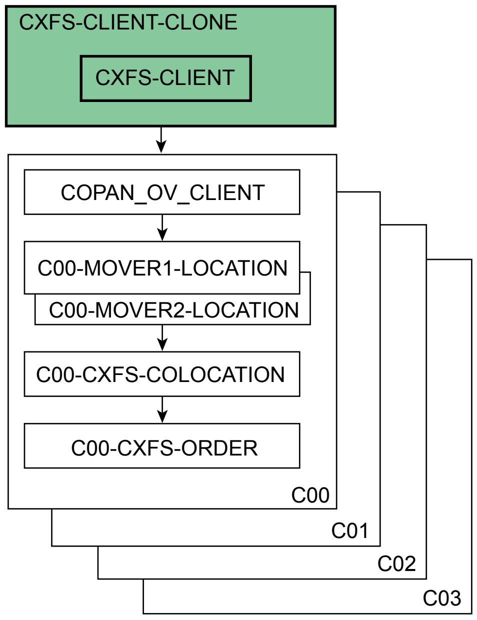

Figure 7-2 Adding the Resources for the Clone and the CXFS Client

As a starting point, use the templates in `/usr/share/doc/sgi-ha/templates` as building blocks.

The instructions in this chapter assume that you use the instance names provided in the templates, such as `CXFS-CLIENT-CLONE`, except as noted (see "Map of Resources for the COPAN MAID OpenVault HA Service" on page 170 and "Conventions for Resource Instance IDs" on page 19).

Do the following:

1. Copy the contents of the `/usr/share/doc/sgi-ha/templates/cxfs-client-clone` template into a new file partial configuration file (referred to as *workfile*).

   A `clone` resource named `CXFS-CLIENT-CLONE` implements control of a CXFS client. It is used in a COPAN MAID OpenVault client HA service.

   The template is located in:

   ```
   /usr/share/doc/sgi-ha/templates/cxfs-client-clone
   ```

   See "Use the Templates as Building Blocks" on page 18.

   Use the following:

   ```
   clone CXFS-CLIENT-CLONE  CXFS-CLIENT \
      meta clone-max="2" target-role="Stopped" interleave="true"
   ```

Add the `cxfs-client` Resource

Do the following:

1. Copy the `primitive` text from the `/usr/share/doc/sgi-ha/templates/cxfs-client` template into *workfile* and replace the site-specific variables as directed.

   A `cxfs-client` resource controls the CXFS client. The template is located in:

   ```
   /usr/share/doc/sgi-ha/templates/cxfs-client
   ```

   See "Use the Templates as Building Blocks" on page 18.

Use the following:

```
primitive CXFS-CLIENT ocf:sgi:cxfs-client \
   op monitor interval="0" timeout="30s" \
   op monitor interval="120s" timeout="30s" on-fail="restart" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="600s" on-fail="fence" \
   params volnames="VOLUME-LIST"
```

Variable	**Description**
CXFS-CLIENT	Name of this resource instance, such as `CXFS-CLIENT`
VOLUME-LIST	Comma-separated list of volume names under the `/dev/cxvm` directory for all of the managed filesystems and the DMF administrative filesystems represented by the following parameters in the DMF configuration file:

`CACHE_DIR`
`SPOOL_DIR`
`TMP_DIR`
`MOVE_FS`
`STORE_DIRECTORY` for a DCM MSP

For example, suppose you have the following output:

```
# ls /dev/cxvm
cache        dmfusr2       move
diskmsp      home          spool
dmfusr1      journal       tmp
```

Note: In a COPAN MAID OpenVault client HA service, you should not include the `HOME_DIR` or `JOURNAL_DIR` filesystems because a parallel data-mover node typically needs no access to these filesystems.

You would likely enter the following (everything except `home` and `journal`):

```
cache,diskmsp,dmfusr1,dmfusr2,move,spool,tmp
```

For example:

```
primitive CXFS-CLIENT ocf:sgi:cxfs-client \
   op monitor interval="0" timeout="30s" \
   op monitor interval="120s" timeout="30s" on-fail="restart" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="600s" on-fail="fence" \
   params volnames="cache,diskmsp,dmfusr1,dmfusr2,move,spool,tmp"
```

2. Verify that the `timeout` values are appropriate for your site.
3. Verify that there are no comments in *workfile*.
4. Save *workfile*.
5. Update the database:

```
node1# crm configure load update workfile
```

Note: As a best practice, you should also run the following command to verify changes you make to the CIB:

```
node1# crm_verify -LV
```

For simplicity, this step is not included in the following procedures but is recommended. For more information, see "Use the `crm_verify` Command to Verify Configuration" on page 24.

Test the Clone

1. Start the `clone`. For example:

```
mover1# 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 `mover1`. For example (truncated):

```
mover1# crm status inactive
...
2 Nodes configured, 2 expected votes
2 Resources configured.

Online: [ mover1 mover2 ]

 Clone Set: CXFS-CLIENT-CLONE [CXFS_CLIENT]
     Started: [ mover1 mover2 ]
```

b. Verify that the `cxfs_client` process is running on `mover1` and `mover2`. For example:

- On `mover1`:

```
mover1# 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 `mover2`:

```
mover2# 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 `mover2` to `standby` state to ensure that the resources remain on `mover1`:

```
mover1# crm node standby mover2
```

4. Confirm that `mover2` is offline and that the resources are off:

a. View the status of the cluster on `mover1`, which should show that `mover2` is in `standby` state. For example:

```
mover1# crm status inactive
    ...
    2 Nodes configured, 2 expected votes
    2 Resources configured.

    Node mover2: standby
    Online: [ mover1 ]

    Clone Set: CXFS-CLIENT-CLONE [CXFS_CLIENT]
        Started: [ mover1 ]
        Stopped: [ cxfs_client:1 ]
```

b. Verify that the `cxfs_client` process is not running on `mover2`. For example, executing the following command on `mover2` should provide no output:

```
mover2# ps -ef | grep cxfs_client | grep -v grep
mover2#
```

5. Return `mover2` to online status by executing the following on `mover1`:

```
mover1# crm node online mover2
```

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.

Add the copan_ov_client Resources

Figure 7-3 shows the steps to add the resources for the set of COPAN MAID OpenVault client resources.

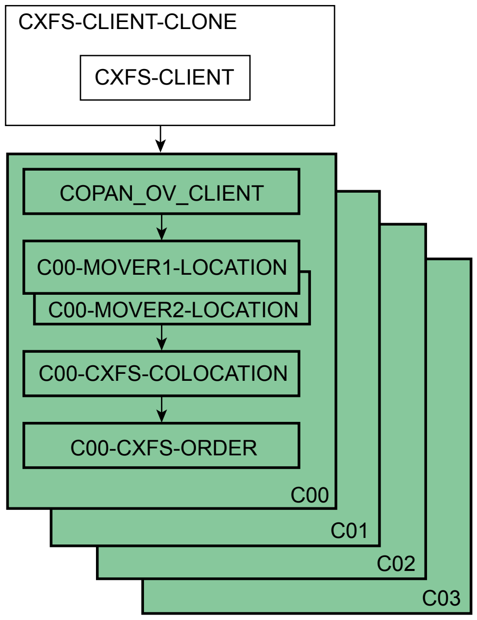

Figure 7-3 Adding the Resources for the COPAN MAID OpenVault Clients

Do the following to add a `copan_ov_client` resource:

1. Create another *workfile* file that contains the `primitive` and `location` text for the `copan_ov_client` resource, replacing the site-specific variables as directed.

   A `copan_ov_client` resource defines a COPAN MAID shelf to be used by an OpenVault client. Each shelf requires its own `primitive` instance.

   This template is located in:

   `/usr/share/doc/sgi-ha/templates/copan_ov_client`

   See "Use the Templates as Building Blocks" on page 18.

   Use the following:

   ```
   primitive SHELF ocf:sgi:copan_ov_client \
      op monitor interval="120s" timeout="60s" on-fail="restart" \
      op monitor interval="0" timeout="60s" \
      op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
      op stop interval="0" timeout="120s" on-fail="fence" \
      params shelf_name="SHELF" \
      meta resource-stickiness="250" migration-threshold="1"
   location SHELF-MOVER1-LOCATION SHELF 200: MOVER1
   location SHELF-MOVER2-LOCATION SHELF 100: MOVER2
   colocation SHELF-CXFS-COLOCATION inf: SHELF CXFS-CLIENT-CLONE
   order SHELF-CXFS-ORDER inf: CXFS-CLIENT-CLONE SHELF
   ```

   Note: The `copan_ov_client` template contains two sections, one for a COPAN MAID OpenVault client HA service and another for a DMF HA service – you must fill in the values in the section for the service you want to configure and delete the other section entirely.

Variable	**Description**
SHELF	Name of this resource instance, normally the three-character COPAN shelf ID, using the naming convention described in the *COPAN MAID for DMF Quick Start Guide*, such as `C00` for cabinet 0, shelf 0 (the bottom shelf)

SHELF-MOVER1-LOCATION	Name of the `location` constraint for the primary parallel data-mover node, such as `C00-MOVER1-LOCATION`
MOVER1	Name of the primary parallel data-mover node, such as `MOVER1`
SHELF-MOVER2-LOCATION	Name of the `location` constraint for the secondary parallel data-mover node, such as `C00-MOVER2-LOCATION`
MOVER2	Name of the secondary parallel data-mover node, such as `MOVER2`
SHELF-CXFS-COLOCATION	Name of the `colocation` constraint, such as `C00-CXFS-COLOCATION`
CXFS-CLIENT-CLONE	Name of the `clone`, such as `CXFS-CLIENT-CLONE`
SHELF-CXFS-ORDER	Name of the `order` constraint, such as `C00-CXFS-ORDER`

For example:

```
primitive C00 ocf:sgi:copan_ov_client \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   params shelf_name="C00" \
   meta resource-stickiness="250" migration-threshold="1"

location C00-MOVER1-LOCATION C00 200: MOVER1
location C00-MOVER2-LOCATION C00 100: MOVER2
colocation C00-CXFS-COLOCATION inf: C00 CXFS-CLIENT-CLONE
order C00-CXFS-ORDER inf: CXFS-CLIENT-CLONE C00
```

Note: The `primitive` instance for the COPAN MAID OpenVault client HA service requires `location`, `colocation`, and `order` constraints and a high value for `resource-stickiness` indicates that it will be unlikely to move to the other node.

---

2. Save *workfile*.
3. Update the database:

   ```
   node1# crm configure load update workfile
   ```

4. Repeat steps 1–3 for any other shelves to be managed by the HA service.

Create the STONITH Capability

Follow the instructions in Chapter 8, "Create the Fencing Capability and Put Into Production Mode" on page 185 to create the STONITH capability.

Test the `copan_ov_client` Resource

Test the `copan_ov_client` resource by manually moving it 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 C01 resource from parallel data-mover node mover1 to parallel data-mover node mover2:

1. Verify that mover2 is ready to receive the resource by examining its status output with the crm(8) command. For example:

   ```
   mover2# crm status inactive
   ...
   2 Nodes configured, 2 expected votes
   6 Resources configured.

   Online: [ mover1 mover2 ]

   Clone Set: CXFS-CLIENT-CLONE [CXFS_CLIENT]
       Started: [ mover1 mover2 ]
   C00 (ocf::sgi:copan_ov_client):     Started mover1
   C01 (ocf::sgi:copan_ov_client):     Started mover1
   Clone Set: STONITH-node1 [stonith-ipmi]
       Started: [ mover1 mover2 ]
   ```

   In the above output, note the following:

   - The C01 resource is Started on mover1
   - mover2 is Online
   - The CXFS-CLIENT and STONITH-node1 clones have a status of Started on mover2

2. On the DMF server, disable mover1 so that it will cease all COPAN MAID shelf activity:

   ```
   dmfserver# dmnode_admin -d mover1
   ```

3. Verify that there are no dmatwc or dmatrc data-mover processes running on mover1. For example, the output of the following command should be empty:

   ```
   mover1# ps -ef | egrep 'dmatrc|dmatwc' | grep -v grep
   mover1#
   ```

   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 `mover2`:

   ```
   mover2# crm resource failcount C00 delete mover2
   mover2# crm resource move C00 mover2
   ```

   This may take a few moments to complete.

5. Verify that the resource has moved to `mover2`. For example:

   ```
   mover2# crm status inactive
   ...
   2 Nodes configured, 2 expected votes
   6 Resources configured.

   Online: [ mover1 mover2 ]

   Clone Set: CXFS-CLIENT-CLONE [CXFS_CLIENT]
       Started: [ mover1 mover2 ]
   C00 (ocf::sgi:copan_ov_client):     Started mover1
   C01 (ocf::sgi:copan_ov_client):     Started mover2
   Clone Set: STONITH-node1 [stonith-ipmi]
       Started: [ mover1 mover2 ]
   ```

   In the above output, note that `C01` resource is started on `mover2`.

6. Remove the constraint that requires `C01` to be controlled by `mover2`:

   ```
   mover2# crm resource unmove C01
   ```

7. Repeat steps 3 through 6 for any other `copan_ov_client` resources requiring a move from `mover1` to `mover2`.

8. On the DMF server, reenable `mover1` so that it can resume COPAN MAID shelf activity:

   ```
   dmfserver# dmnode_admin -e mover1
   ```

Confirm the Completed Status

Use the `status` command to confirm the resulting HA cluster:

```
node1# crm status inactive
```

Put the HA Service into Production Mode

Given that the fencing capability exists and the resources have been tested, you can now use the HA cluster in production mode.

Create the Fencing Capability and Put Into Production Mode

This chapter discusses the following steps to ensure data integrity and put the tested HA cluster into production mode:

- "Create the STONITH Facility" on page 185
- "Enable Node-Level Fencing" on page 187
- "Test the STONITH Facility" on page 187
- "Remove Constraints" on page 187
- "Enable Cluster Services to be Started Automatically" on page 187

Create the STONITH Facility

Create the appropriate STONITH facility:

- "RHEL STONITH" on page 185
- "SLES STONITH" on page 186

RHEL STONITH

Do the following to create a STONITH facility for RHEL:

1. Copy the contents of the `/usr/share/doc/sgi-ha/templates/fence-ipmilan` template into a new file partial configuration file (referred to as *workfile*) and replace the site-specific variables as directed in "`fence_ipmilan` Template" on page 276.
2. Verify that the `timeout`, `userid`, `passwd`, and `interface` values are appropriate for your site.
3. Verify that there are no comments in *workfile*.
4. Save *workfile*.

5. Update the database:

   ```
   rhel# crm configure load update workfile
   ```

6. Repeat steps 1 through 5 for the second node, using different `delay` values, such as `0` and `15`.

7. To prevent the `acpid` service from inappropriately attempting a graceful shutdown, ensure that the services not currently running and will not restart (on both RHEL nodes):

   - On RHEL `node1`:

     ```
     node1# service acpid stop
     node1# chkconfig acpid off
     ```

   - On RHEL `node2`:

     ```
     node2# service acpid stop
     node2# chkconfig acpid off
     ```

SLES STONITH

Do the following to create a STONITH facility for SLES:

1. Copy the contents of the `/usr/share/doc/sgi-ha/templates/ipmi` template into a new file partial configuration file (referred to as *workfile*) and replace the site-specific variables as directed in "`ipmi` Template" on page 287.

2. Verify that the `timeout`, `userid`, `passwd`, and `interface` values are appropriate for your site.

3. Verify that there are no comments in *workfile*.

4. Save *workfile*.

5. Update the database:

   ```
   node1# crm configure load update workfile
   ```

6. Repeat steps 1 through 5 for the second node.

Enable Node-Level Fencing

Enable node-level fencing (which was previously disabled for testing purposes):

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

Test the STONITH Facility

To test the STONITH facility, use the `node fence` command on one node to fence the other node. For example, to test the failover of resources running on `node1`:

1. From `node2`, reset `node1`:

   ```
   node2# crm node fence node1
   ```

2. Wait for the failover to occur.
3. On `node2`, execute the following:

   ```
   node2# crm status inactive
   ```

   If all of the resources are not failed over, wait again and repeat the command.

Remove Constraints

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 resource
```

Enable Cluster Services to be Started Automatically

This section discusses the following:

- "RHEL: Enable `cman` and `pacemaker` to be Started Automatically" on page 188
- "SLES: Enable `openais` to be Started Automatically" on page 188

RHEL: Enable `cman` and `pacemaker` to be Started Automatically

If you are using RHEL, enable the `cman` and `pacemaker` services to be started automatically at boot time:

1. On `node1`:

   ```
   node1# chkconfig cman on
   node1# chkconfig pacemaker on
   ```

2. On `node2`:

   ```
   node2# chkconfig cman on
   node2# chkconfig pacemaker on
   ```

SLES: Enable `openais` to be Started Automatically

If you are using SLES, enable the `openais` service to be started automatically at boot time:

- On `node1`:

  ```
  node1# chkconfig openais on
  ```

- On `node2`:

  ```
  node2# chkconfig openais on
  ```

Chapter 9

Administrative Tasks and Considerations

This chapter discusses various administrative tasks and considerations for an HA cluster:

Viewing the Cluster Status

Use the following commands to view the cluster status:

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

For example, the following truncated output shows that there was an error in starting DMF on `node2`:

```
node1# crm status failcounts
...
2 Nodes configured, 2 expected votes
9 Resources configured.

Online: [ node1 node2 ]

  Resource Group: DMF-GROUP
      LXVM      (ocf::sgi:lxvm):        Started node1
      home      (ocf::heartbeat:Filesystem):    Started node1
      spool     (ocf::heartbeat:Filesystem):    Started node1
      journals  (ocf::heartbeat:Filesystem):    Started node1
      tmp       (ocf::heartbeat:Filesystem):    Started node1
      store     (ocf::heartbeat:Filesystem):    Started node1
      dmi_fs    (ocf::heartbeat:Filesystem):    Started node1
      IP        (ocf::heartbeat:IPaddr2):       Started node1
      DMF       (ocf::sgi:dmf): Started node1

Migration summary:
* Node node1:
* Node node2:
    DMF: migration-threshold=1 fail-count=1000000

Failed actions:
     DMF_start_0 (node=node2, call=34, rc=-2, status=Timed Out): unknown exec error
```

Viewing the Cluster Configuration

To view the cluster configuration, enter the following:

```
ha# crm configure show
```

If you want to view the entire contents of the cluster information base (CIB) in XML format, including dynamic status information, you can use the following `cibadmin`(8) command:

```
ha# cibadmin -Q
```

To limit the output to a specific set of information, use the following:

```
ha# cibadmin -o modifier -Q
```

The *modifier* value can be one of the following:

```
constraints
crm_config
nodes
resources
status
```

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
```

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 233). 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 Files

See "Examine Log Files" on page 24.

Clearing the Resource Failcount

To clear individual resource failcounts, either reboot the nodes or enter the following on each node for each individual resource:

```
ha# crm resource failcount resource 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 resource 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/pacemaker/pengine`.

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 documentation listed in "High Availability Overview" on page 1.

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 those parameters that require 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 `C01` resource from parallel data-mover node `mover1` to parallel data-mover node `mover2`:

1. Verify that `mover2` is ready to receive the resource by examining its status output with the `crm`(8) command. For example:

   ```
   mover2# crm status inactive
   ...
   2 Nodes configured, 2 expected votes
   6 Resources configured.

   Online: [ mover1 mover2 ]

   Clone Set: CXFS-CLIENT-CLONE [CXFS_CLIENT]
       Started: [ mover1 mover2 ]
   C00 (ocf::sgi:copan_ov_client):     Started mover1
   ```

```
C01 (ocf::sgi:copan_ov_client):     Started mover1
Clone Set: STONITH-node1 [stonith-ipmi]
    Started: [ mover1 mover2 ]
```

In the above output, note the following:

- The `C01` resource is `Started` on `mover1`
- `mover2` is `Online`
- The `CXFS-CLIENT` and `STONITH-node1` clones have a status of `Started` on `mover2`

2. On the DMF server, disable `mover1` so that it will cease all COPAN MAID shelf activity:

   ```
   dmfserver# dmnode_admin -d mover1
   ```

3. Verify that there are no `dmatwc` or `dmatrc` data-mover processes running on `mover1`. For example, the output of the following command should be empty:

   ```
   mover1# ps -ef | egrep 'dmatrc|dmatwc' | grep -v grep
   mover1#
   ```

   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 `mover2`:

   ```
   mover2# crm resource failcount C00 delete mover2
   mover2# crm resource move C00 mover2
   ```

   This may take a few moments to complete.

5. Verify that the resource has moved to `mover2`. For example:

   ```
   mover2# crm status inactive
   ...
   2 Nodes configured, 2 expected votes
   6 Resources configured.

   Online: [ mover1 mover2 ]

   Clone Set: CXFS-CLIENT-CLONE [CXFS_CLIENT]
       Started: [ mover1 mover2 ]
   ```

```
C00 (ocf::sgi:copan_ov_client):     Started mover1
C01 (ocf::sgi:copan_ov_client):     Started mover2
Clone Set: STONITH-node1 [stonith-ipmi]
    Started: [ mover1 mover2 ]
```

In the above output, note that `C01` resource is started on `mover2`.

6. Remove the constraint that requires `C01` to be controlled by `mover2`:

```
mover2# crm resource unmove C01
```

7. Repeat steps 3 through 6 for any other `copan_ov_client` resources requiring a move from `mover1` to `mover2`.

8. On the DMF server, reenable `mover1` so that it can resume COPAN MAID shelf activity:

```
dmfserver# dmnode_admin -e mover1
```

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 appropriate underlying HA control services (`cman` and `pacemaker` for RHEL, `openais` for SLES) 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 213.

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:

- "COPAN MAID OpenVault Client HA Service for Mover Nodes Rolling Upgrade" on page 198
- "CXFS NFS Edge-Serving HA Rolling Upgrade" on page 200
- "DMF HA Rolling Upgrade" on page 203

COPAN MAID OpenVault Client 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 (`mover1` and `mover1`):

1. Read the release notes for the software you intend to upgrade 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 `mover2` to `mover1` according to the directions in "Manually Moving a `copan_ov_client` Resource" on page 195.

4. Stop all services on `mover2` and ensure that they will not restart on the next boot of `mover2`:

   - RHEL:

     ```
     mover2# chkconfig cman off
     mover2# chkconfig pacemaker off

     mover2# service pacemaker stop
     ```

   - SLES:

     ```
     mover2# chkconfig openais off

     mover2# service openais stop
     ```

5. Upgrade the software on `mover2`.

6. Reboot `mover2`.

7. Ensure that the services will restart upon reboot and start services immediately:

   - RHEL:

     ```
     mover2# service cman start
     mover2# service pacemaker start

     mover2# chkconfig cman on
     mover2# chkconfig pacemaker on
     ```

   - SLES:

     ```
     mover2# service openais start

     mover2# chkconfig openais on
     ```

8. Move all `copan_ov_client` resources running on `mover1` to `mover2` according to the directions in "Manually Moving a `copan_ov_client` Resource" on page 195.

9. Repeat step 4 through step 7 above but executed for `mover1`.

10. Move the `copan_ov_client` resources than normally run on `mover1` back to that node, according to the directions in "Manually Moving a `copan_ov_client` Resource" on page 195.

11. Remove the constraints required to move the resources by executing the following command for each `copan_ov_client` resource

    ```
    mover2# crm resource unmove copan_ov_client_name
    ```

    The cluster is now back to normal operational state.

CXFS NFS Edge-Serving HA Rolling Upgrade

Note: 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. Therefore, you must use CXFS recovery and not CXFS relocation during a rolling upgrade.

Do the following for a rolling upgrade in a CXFS NFS edge-serving 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, 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 appropriate underlying HA control services from being started automatically at boot time and stop the currently running services:

   - RHEL:

     ```
     node2# chkconfig cman off
     node2# chkconfig pacemaker off

     node2# service pacemaker stop
     ```

   - SLES:

```
node2# chkconfig openais off
node2# service openais stop
```

6. Upgrade the software on `node2`.
7. Reboot `node2` and verify basic upgraded functionality.
8. Enable HA services to be started automatically at boot time and immediately start it on `node2`:

   - RHEL:

     ```
     node2# chkconfig cman on
     node2# chkconfig pacemaker on

     node2# service cman start
     node2# service pacemaker start
     ```

   - SLES:

     ```
     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 CXFS services from being started automatically at boot time:

   ```
   node2# chkconfig cxfs off
   node2# chkconfig cxfs_cluster off
   ```

4. Disable the appropriate underlying HA control services from being started automatically at boot time:

   - RHEL:

     ```
     node2# chkconfig cman off
     node2# chkconfig pacemaker off
     ```

   - SLES:

     ```
     node2# chkconfig openais off
     ```

5. Ensure that the resource groups are running on `node1`.

Note: Moving the DMF-GROUP 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 DMF-GROUP resource group via CXFS recovery by resetting the node that is running the DMF-GROUP resource.

Do one of the following, as appropriate for your site:

- Using CXFS **recovery**:

  ```
  node1# crm node fence node2
  ```

- Using CXFS **relocation**:

  Note: Stopping the appropriate underlying HA control services 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.

  – RHEL:

  ```
  node2# crm resource move DMF-GROUP node1

  node2# service pacemaker stop
  node2# service cxfs stop
  node2# service cxfs_cluster stop
  ```

  – SLES:

  ```
  node2# crm resource move DMF-GROUP node1

  node2# service openais stop
  node2# service cxfs stop
  node2# service cxfs_cluster stop
  ```

6. Verify that the services have moved to node1.

7. Upgrade the software on node2 and reboot.

8. 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
   ```

9. Verify that `node2` is fully back in the CXFS cluster with filesystems mounted.

10. Enable the appropriate underlying HA control services to be started automatically at boot time and then start them immediately on `node2`:

    • RHEL:

    ```
    node2# chkconfig cman on
    node2# chkconfig pacemaker on

    node2# service cman start
    node2# service pacemaker start
    ```

    • SLES:

    ```
    node2# chkconfig openais on

    node2# service openais start
    ```

11. Repeat steps 3 through 10 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 DMF-GROUP node1
   ```

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

   ```
   node1# crm resource unmove DMF-GROUP
   ```

   The cluster is now back to normal operational state.

Stopping the Underlying HA Control Services

Note: Stopping HA control services 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 appropriate underlying HA control services on the local node, enter the following:

- RHEL:

  ```
  ha# service pacemaker stop
  ```

- SLES:

  ```
  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 and enabled in the CIB.

For example, to reset `node1` from `node2`:

```
node2# crm node fence node1
```

Hardware Maintenance on a Cluster Node

This section discusses the following:

- "Hardware Maintenance in a COPAN MAID OpenVault Client HA Service" on page 207
- "Hardware Maintenance in a CXFS NFS Edge-Serving HA Service" on page 210
- "Hardware Maintenance in a DMF HA Service" on page 211

Hardware Maintenance in a COPAN MAID OpenVault Client HA Service

If you must perform maintenance on one node in the cluster that provides the COPAN MAID OpenVault client HA service, do the following:

1. 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.
2. On the DMF server, disable the parallel data-mover node on which you want to perform maintenance (`downnode`) so that it will cease all COPAN MAID shelf activity:

   ```
   dmfserver# dmnode_admin -d downnode
   ```

   Wait for the activity to cease on `downnode` before continuing to the next step.

3. Verify that there are no `dmatwc` or `dmatrc` data-mover processes running on `downnode`. For example, the output of the following command should be empty:

   ```
   downnode# ps -ef | egrep 'dmatrc|dmatwc' | grep -v grep
   downnode#
   ```

   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. Ensure that any `copan_ov_client` shelf resources are running on the node that **does not** require maintenance (`upnode`). For example:

   ```
   downnode# crm resource move C00 upnode
   downnode# crm resource move C01 upnode
   ```

5. Set `downnode` 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:

   ```
   upnode# crm node standby downnode
   ```

   This will shut down `CXFS_CLIENT` on `downnode` automatically.

6. Disable the appropriate underlying HA control services from being started automatically on `downnode` at boot time and stop the currently running services:

   - RHEL:

     ```
     downnode# chkconfig cman off
     downnode# chkconfig pacemaker off

     downnode# service pacemaker stop
     ```

   - SLES:

     ```
     downnode# chkconfig openais off
     downnode# service openais stop
     ```

7. Perform maintenance on `downnode`.

8. Reboot `downnode` and verify basic functionality.

9. Enable HA services to be started automatically at boot time and immediately start it on `downnode`:

   - RHEL:

     ```
     downnode# chkconfig cman on
     downnode# chkconfig pacemaker on

     downnode# service cman start
     downnode# service pacemaker start
     ```

   - SLES:

     ```
     downnode# chkconfig openais on
     downnode# service openais start
     ```

10. Make `downnode` active again:

    ```
    downnode# crm node online downnode
    ```

11. (Optional) If you want to move shelf resources back to `downnode`:

    a. On the DMF server, disable the other parallel data-mover node (`upnode`):

    ```
    dmfserver# dmnode_admin -d upnode
    ```

    Wait for the activity to cease on `upnode` before continuing to the next step.

    b. Verify that there are no `dmatwc` or `dmatrc` data-mover processes running on `upnode`, as in step 3.

    c. Move back all of the `copan_ov_client` resources (for example, `C00` and `C01`) that you want to run on `downnode`:

    ```
    upnode# crm resource move C00 downnode
    upnode# crm resource move C01 downnode
    ```

    d. On the DMF server, reenable `upnode` so that it will permit COPAN MAID shelf activity:

    ```
    dmfserver# dmnode_admin -e upnode
    ```

    e. On the DMF server, reenable `downnode` so that it will permit COPAN MAID shelf activity:

    ```
    dmfserver# dmnode_admin -e downnode
    ```

12. Remove any implicit location constraints imposed by the administrative `move` commands above. For example:

    ```
    downnode# crm resource unmove C00
    downnode# crm resource unmove C01
    ```

Hardware Maintenance in a CXFS NFS Edge-Serving HA Service

If you must perform maintenance on one node in the cluster that provides the CXFS NFS edge-serving HA service, do the following:

1. 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.
2. Ensure that the resource groups are running on the node that **does not** require maintenance (`upnode`). For example:

   ```
   downnode# crm resource move IPALIAS-GROUP-2 upnode
   ```

3. Set the node on which you intend to perform maintenance (`downnode`) 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 `downnode`:

   ```
   upnode# crm node standby downnode
   ```

   This will shut down `CXFS_CLIENT` on `downnode` automatically.
4. Disable the appropriate underlying HA control services from being started automatically on `downnode` at boot time and stop the currently running services:
   - RHEL:

     ```
     downnode# chkconfig cman off
     downnode# chkconfig pacemaker off

     downnode# service pacemaker stop
     ```

   - SLES:

     ```
     downnode# chkconfig openais off
     downnode# service openais stop
     ```

5. Perform maintenance on `downnode`.

6. Reboot downnode and verify basic functionality.
7. Enable HA services to be started automatically at boot time and immediately start it on downnode:
   - RHEL:

     ```
     downnode# chkconfig cman on
     downnode# chkconfig pacemaker on

     downnode# service cman start
     downnode# service pacemaker start
     ```

   - SLES:

     ```
     downnode# chkconfig openais on
     downnode# service openais start
     ```

8. Make downnode active again:

   ```
   downnode# crm node online downnode
   ```

9. Move back any resource groups that you want to run on downnode. For example:

   ```
   upnode# crm resource move IPALIAS-GROUP-2 downnode
   ```

10. Remove any implicit location constraints imposed by the administrative move commands above. For example:

    ```
    downnode# crm resource unmove IPALIAS-GROUP-2
    ```

Hardware Maintenance in a DMF HA Service

If you must perform maintenance on one node in the cluster that provides the DMF HA service, do the following:

1. On the node that requires maintenance (downnode), disable the underlying HA control services from being started automatically at boot time:
   - RHEL:

     ```
     downnode# chkconfig cman off
     downnode# chkconfig pacemaker off
     ```

   - SLES:

     ```
     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 inactive
   ```

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 HA service to be started automatically at boot time and start it immediately on downnode:

   - RHEL:

     ```
     downnode# chkconfig cman on
     downnode# chkconfig pacemaker on

     downnode# service cman start
     ```

```
downnode# service pacemaker start
```

- SLES:

```
downnode# chkconfig openais on
```

```
downnode# service openais start
```

10. Verify that `downnode` rejoins the HA cluster:

```
downnode# crm status inactive
```

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 COPAN MAID OpenVault Client HA Service on Mover Nodes" on page 214
- "Full Outage for CXFS NFS Edge-Serving HA Service" on page 216
- "Full Outage for DMF HA Service" on page 218

Full Outage for COPAN MAID OpenVault Client HA Service on Mover Nodes

Do the following to perform a full outage for a COPAN MAID OpenVault client HA service 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 mover1
   dmfserver# dmnode_admin -d mover2
   ```

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 an HA environment:

   - RHEL:

     ```
     ha# chkconfig cman off
     ha# chkconfig pacemaker off
     ```

   - SLES:

     ```
     ha# chkconfig openais off
     ```

4. On both nodes, stop the appropriate underlying HA control services (do this simultaneously):

   - RHEL:

     ```
     ha# service pacemaker stop
     ```

   - SLES:

     ```
     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 an HA environment:
   - RHEL:
     ```
     ha# chkconfig cman on
     ha# chkconfig pacemaker on
     ```
   - SLES:
     ```
     ha# chkconfig openais on
     ```
10. On both nodes, start the appropriate underlying HA control services (do this simultaneously):
    - RHEL:
      ```
      ha# service cman start
      ha# service pacemaker start
      ```
    - SLES:
      ```
      ha# service openais start
      ```
11. On either node, verify cluster status:
    ```
    ha# crm status inactive
    ```
12. On the DMF server, reenable both nodes for COPAN MAID activity:
    ```
    dmfserver# dmnode_admin -e mover1
    dmfserver# dmnode_admin -e mover2
    ```

Full Outage for CXFS NFS Edge-Serving HA Service

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, see Figure 5-1 on page 54). For example, using the example procedures in this guide, you would stop the IP address 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 an HA environment and CXFS from being started automatically at boot time:

   - On all HA servers:
     - RHEL:

       ```
       ha# chkconfig cman off
       ha# chkconfig pacemaker off
       ```

     - SLES:

       ```
       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-servering node, 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 appropriate underlying HA control service from being started automatically at boot time and stop the currently running service immediately:

    - RHEL:

      ```
      ha# chkconfig cman on
      ha# chkconfig pacemaker on

      ha# service cman start
      ha# service pacemaker start
      ```

    - SLES:

      ```
      ha# chkconfig openais on

      ha# service openais start
      ```

11. Verify the HA cluster status:

```
ha# crm status inactive
```

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 Service

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, see Figure 6-1 on page 99). Using the example procedures in this guide, you would stop the group:

```
ha# crm resource stop DMF-GROUP
```

3. Disable the services related to an HA environment and CXFS (if applicable) from being started automatically at boot time:

   • On all HA servers:

     – RHEL:

     ```
     ha# chkconfig cman off
     ha# chkconfig pacemaker off
     ```

     – SLES:

     ```
     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 appropriate underlying HA control services to be started automatically at boot time and start them immediately:
   - RHEL:

     ```
     ha# chkconfig cman on
     ha# chkconfig pacemaker on

     ha# service cman start
     ha# service pacemaker start
     ```

- SLES:

  ```
  ha# chkconfig openais on
  ```

  ```
  ha# service openais start
  ```

10. Verify HA cluster status:

    ```
    ha# crm status inactive
    ```

11. Start resources in the correct order (top-down). For example:

    ```
    ha# crm resource start DMF-GROUP
    ```

12. Move the resources to the correct locations:

    ```
    ha# crm resource move DMF-GROUP 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 DMF-GROUP
    ```

Chapter 10

Troubleshooting

This chapter discusses the following:

- "Diagnosing Problems" on page 221
- "Failover Testing Strategies" on page 228
- "Corrective Actions" on page 232

For more information, see the documentation listed in “About this Guide”.

Diagnosing Problems

If you notice problems, do the following:

- "Monitor the Status Output" on page 221
- "Verify the Configuration in Greater Detail" on page 222
- "Match Status Events To Error Messages" on page 222
- "Verify `chkconfig` Settings" on page 222
- "Diagnose the Problem Resource" on page 224
- "Examine Application-Specific Problems that Impact HA" on page 224
- "Test the STONITH Capability" on page 224
- "Gather Troubleshooting Data" on page 225
- "Use SGI Knowledgebase" on page 228

Monitor the Status Output

Use the `crm status inactive` 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
```

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 the log files. See "Examine Log Files" on page 24.

Verify `chkconfig` Settings

Verify the `chkconfig` settings for the following services when used in an HA cluster:

- "Off" on page 223
- "On" on page 223
- "Optionally On" on page 223

Off

The following services must be `off` if used (most services):

`acpid` (RHEL)
`cxfs_client`
`dmf`
`dmfman`
`dmfsoap`
`nfs` (RHEL)
`nfsserver` (SLES)
`openvault`
`smb`
`nmb`
`winbind`

On

The following services must be `on` if used:

- RHEL:

  ```
  cman
  cxfs
  cxfs_cluster
  nfslock
  pacemaker
  ```

- SLES:

  ```
  cxfs
  cxfs_cluster
  openais
  logd
  ```

Optionally On

The following service may optionally may be `on`:

`tmf` (if used)

Diagnose the Problem Resource

To diagnose problems at the application level, put the cluster into maintenance mode. You might have to stop the appropriate underlying HA control services on all nodes (`cman` and `pacemaker` on RHEL nodes or `openais` on SLES 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 lan  -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 lan -U admin -P admin -H 128.162.232.79 power reset
```

     Note: Some BMC hardware may require `-I lanplus`.

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 225
- "Collect System Logs" on page 226
- "Collect HA Cluster Information" on page 226
- "Collect SGI Service-Specific Information" on page 227
- "Generate a Kernel Crash Dump" on page 227

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:

- RHEL:

  ```
  rhel# /usr/sbin/system_info_gather  -Avv -n
  ```

- SLES:

  ```
  sles# /usr/sbin/system_info_gather  -Avv -n
  sles# /sbin/supportconfig
  ```

The `system_info_gather` script will print information to a time-stamped file named using the following format:

yyyymmdd_hhmmss-*hostname*-`inventory`*[options]*

For example, for output gathered on October 17, 2013, for a host named `myserver` using the `-Avv` options:

```
20131017_103317-myserver.mycompany.com-inventoryA2
```

Collect System Logs

Collect any other system log files that may contain information about `corosync/pacemaker` or the services included in the HA configuration (if not otherwise gathered by the above tools).

Collect HA Cluster Information

To collect HA cluster information, use the following commands:

- RHEL: `crm_report`(8)
- SLES: `hb_report`(8)

The commands use a similar format:

```
ha# command -f "from_time" [-t "to_time"] [destination_directory]
```

where:

- *command* is either `crm_report` (RHEL) or `hb_report` (SLES)
- *from_time* and *to_time* are in the format *YYYY-MM-DD H:M:S*. If you omit the `-t` option, the command collects data up to the present.
- *destination_directory* is the destination directory. If you omit this argument, the command will use its own naming convention for the compressed tarball:
  - RHEL default:

    `pcmk-`*date*`.tar.bz2`

    For example, the tarball will be named `pcmk-Tue-10-Dec-2013.tar.bz2` if you execute the following command on 10 December 2013:

    ```
    rhel# crm_report -f "2013-12-09 08:00"
    ```
  - SLES default:

    `hb_report-`*date*`.tar.bz2`

    For example, the tarball will be named `hb_report-Tue-10-Dec-2013.tar.bz2` if you execute the following command on 10 December 2013:

    ```
    sles# hb_report -f "2013-12-09 08:00"
    ```

For more details, see the `crm_report`(8) and `hb_report`(8) man pages.

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:

   ```
   ha# 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 status failcounts
   ```

   b. If there has been a failure, ensure that the issue that caused a resource failure has been resolved.

   c. Reset the individual 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 for the resource group 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 10-1 describes failover testing strategies.

Table 10-1 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: • RHEL: `rhel#` **`service nfs stop`** • SLES: `sles#` **`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 232
- "Recovering from a CIB Corruption" on page 233
- "Clearing the Failcounts After a Severe Error" on page 233

Recovering from an Incomplete Failover

After an incomplete failover, in which one or more of the individual resources are not started and the cluster can no longer provide high availability, you must do the following to restore functionality:

1. Put the cluster into maintenance mode:

   ```
   ha# crm configure property maintenance-mode=true
   ```

2. Determine which individual resources have failcounts:

   ```
   ha# crm resource failcount resourcePRIMITIVE show node
   ```

   Repeat for each individual resource on each node.

3. Troubleshoot the failed resource operations. Examine the log files (see "Examine Log Files" on page 24) 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.

5. Remove the failcounts found in step 2:

   ```
   ha# crm resource failcount failed_resourcePRIMITIVE delete node
   ```

   Repeat this for each individual failed resource on each node.

6. Remove error messages:

   ```
   ha# crm resource cleanup failed_resourcePRIMITIVE node
   ```

   Repeat this for each individual failed resource on each node.

7. Return the cluster to managed status, enter 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 192.

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 individual resources to be set to `INFINITY`. This means that the individual resources cannot run on a specific node again until the failcount is cleared, which requires administrative action. See "Clearing the Resource Failcount" on page 192.

Resource Reference

This appendix discusses the following:

`copan_ov_client`

This section discusses the following:

- "`copan_ov_client` Purpose" on page 237
- "`copan_ov_client` Requirements" on page 237
- "Testing Before Applying HA" on page 238
- "HA Control of the Service" on page 238
- "`copan_ov_client` Template" on page 238

`copan_ov_client` Purpose

The `copan_ov_client` resource defines a COPAN MAID shelf to be used by an OpenVault client. Each shelf requires its own `primitive` instance. It is used in a COPAN MAID OpenVault client HA service for DMF parallel data-mover nodes or optionally in a DMF HA service. See:

- Chapter 7, "Create the COPAN MAID OpenVault Client HA Service for Mover Nodes" on page 165
- Chapter 6, "Create the DMF HA Service" on page 75

The following figures depict the order in which this resource falls in the configuration process:

- Figure 6-4 on page 111
- Figure 7-3 on page 178

`copan_ov_client` Requirements

See:

- "COPAN MAID Requirements for a DMF HA Service *(Optional)*" on page 81
- "COPAN MAID Requirements for an OpenVault Client HA Service" on page 165

Testing Before Applying HA

See "Configure and Test the COPAN MAID OpenVault Client Before Applying a DMF HA Environment" on page 88.

HA Control of the Service

To use the `copan_ov_client` resource, you must stop the `openvault` service on each node before applying an HA environment:

```
node1# chkconfig openvault off
node1# service openvault off

node2# chkconfig openvault off
node2# service openvault off
```

The HA software will control this service.

`copan_ov_client` Template

Template location:

`/usr/share/doc/sgi-ha/templates/copan_ov_client`

Use the following:

- **COPAN MAID OpenVault client HA service:**

```
primitive SHELF ocf:sgi:copan_ov_client \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   params shelf_name="SHELF" \
   meta resource-stickiness="250" migration-threshold="1"
location SHELF-MOVER1-LOCATION SHELF 200: MOVER1
location SHELF-MOVER2-LOCATION SHELF 100: MOVER2
colocation SHELF-CXFS-COLOCATION inf: SHELF CXFS-CLIENT-CLONE
order SHELF-CXFS-ORDER inf: CXFS-CLIENT-CLONE SHELF
```

- **DMF HA service:**

```
primitive SHELF ocf:sgi:copan_ov_client \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   params shelf_name="SHELF" \
   meta resource-stickiness="1" migration-threshold="1"
```

Note: The `copan_ov_client` template contains two sections, one for a COPAN MAID OpenVault client HA service and another for a DMF HA service – you must fill in the values in the section for the service you want to configure and delete the other section entirely.

Variable	Description
SHELF	Name of this resource instance, normally the three-character COPAN shelf ID, using the naming convention described in the *COPAN MAID for DMF Quick Start Guide*, such as `C00` for cabinet 0, shelf 0 (the bottom shelf)
SHELF-MOVER1-LOCATION	Name of the `location` constraint for the primary parallel data-mover node, such as `C00-MOVER1-LOCATION`
MOVER1	Name of the primary parallel data-mover node, such as `MOVER1`
SHELF-MOVER2-LOCATION	Name of the `location` constraint for the secondary parallel data-mover node, such as `C00-MOVER2-LOCATION`
MOVER2	Name of the secondary parallel data-mover node, such as `MOVER2`
SHELF-CXFS-COLOCATION	Name of the `colocation` constraint, such as `C00-CXFS-COLOCATION`

CXFS-CLIENT-CLONE	Name of the `clone`, such as `CXFS-CLIENT-CLONE` (see "`cxfs-client-clone` Template" on page 249)
SHELF-CXFS-ORDER	Name of the `order` constraint, such as `C00-CXFS-ORDER`

For example, for a **COPAN MAID OpenVault client HA service**:

```
primitive C00 ocf:sgi:copan_ov_client \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   params shelf_name="C00" \
   meta resource-stickiness="250" migration-threshold="1"

location C00-MOVER1-LOCATION C00 200: MOVER1
location C00-MOVER2-LOCATION C00 100: MOVER2
colocation C00-CXFS-COLOCATION inf: C00 CXFS-CLIENT-CLONE
order C00-CXFS-ORDER inf: CXFS-CLIENT-CLONE C00
```

For example, for a **DMF HA service**:

```
primitive C16 ocf:sgi:copan_ov_client \
   op monitor interval="0" timeout="60s" \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   params shelf_name="C16" \
   meta resource-stickiness="1" migration-threshold="1"
```

Note: The `primitive` instance for the COPAN MAID OpenVault client HA service requires `location`, `colocation`, and `order` constraints and a high value for `resource-stickiness` indicates that it will be unlikely to move to the other node, whereas the `primitive` instance for the DMF HA service requires no constraints and has a `resource-stickiness` value that matches the rest of the resources in the `DMF-GROUP` resource (see "`dmf-group` Template" on page 268).

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

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

The `stop` operation does the following:

- Stops the COPAN MAID OpenVault client resource
- Fails if the COPAN MAID OpenVault client resource fails to stop

`cxfs`

This section discusses the following:

- "`cxfs` Purpose" on page 242
- "`cxfs` Requirements" on page 242
- "Testing Before Applying HA" on page 242
- "HA Control of the Service" on page 243
- "`cxfs` Template" on page 243

`cxfs` Purpose

The `cxfs` resource controls the CXFS metadata location in a DMF HA service. See Chapter 6, "Create the DMF HA Service" on page 75.

Figure 6-2 on page 101 depicts the order in which this resource falls in the configuration process.

`cxfs` Requirements

See the following:

"CXFS Requirements" on page 76:

- "CXFS Server-Capable Administration Nodes" on page 76
- "CXFS Relocation Support" on page 77
- "Applications that Depend Upon CXFS Filesystems" on page 77
- "CXFS and System Reset" on page 77
- "CXFS Start/Stop Issues" on page 77
- "CXFS Volumes and Filesystems Managed by DMF" on page 78

Testing Before Applying HA

See "Configure and Test CXFS Before Applying an HA Environment" on page 86.

HA Control of the Service

Do not disable the `cxfs` and `cxfs_cluster` services. Unlike other services, these services will not be controlled by HA software in an HA cluster.

`cxfs` Template

Template location:

`/usr/share/doc/sgi-ha/templates/cxfs`

Use the following:

```
primitive CXFS ocf:sgi:cxfs \
   op monitor interval="120s" timeout="180s" on-fail="restart" \
   op monitor interval="0" timeout="180s" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="60s" on-fail="fence" \
   params volnames="VOLUME-LIST" \
   meta resource-stickiness="1" migration-threshold="1"
```

Variable	Description
CXFS	Name of this resource instance, such as `CXFS`
VOLUME-LIST	Comma-separated list of volumes under the `/dev/cxvm` directory, such as: `cxfsvol1,cxfsvol2` **Note:** Do not include any links or volumes that are filesystems managed by DMF.

For example:

```
primitive CXFS ocf:sgi:cxfs \
   op monitor interval="120s" timeout="180s" on-fail="restart" \
   op monitor interval="0" timeout="180s" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="60s" on-fail="fence" \
   params volnames="cxfsvol1,cxfsvol2" \
   meta resource-stickiness="1" migration-threshold="1"
```

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

The `start` operation does the following:

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

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

`cxfs-client`

This section discusses the following:

- "`cxfs-client` Purpose" on page 245
- "`cxfs-client` Requirements" on page 245
- "Testing Before Applying HA" on page 246
- "HA Control of the Service" on page 246
- "`cxfs-client` Template" on page 246

`cxfs-client` Purpose

The `cxfs-client` resource controls the CXFS client. It is used in the CXFS NFS edge-serving HA service and in the COPAN MAID OpenVault client HA service. See:

- Chapter 5, "Create the CXFS NFS Edge-Serving HA Service" on page 47
- Chapter 7, "Create the COPAN MAID OpenVault Client HA Service for Mover Nodes" on page 165

The following figures depict the order in which this resource falls in the configuration process:

- Figure 5-1 on page 54
- Figure 7-2 on page 173

`cxfs-client` Requirements

See the following:

- Edge-serving: "Edge-Serving Requirements" on page 48
- COPAN MAID OpenVault client: "COPAN MAID Requirements in a Mover-Node HA Cluster" on page 166

Testing Before Applying HA

See:

- "Configure and Test the CXFS Client Before Applying a COPAN MAID HA Environment" on page 167
- "Configure and Test the CXFS Client Before Applying the HA Environment" on page 50

HA Control of the Service

To use the `cxfs-client` resource, you must stop the `cxfs_client` service on each node before applying HA:

```
node1# chkconfig cxfs_client off
node1# service cxfs_client stop

node2# chkconfig cxfs_client off
node2# service cxfs_client stop
```

The HA software will control this service.

For related services that must be HA-controlled, see:

- Edge-serving: "Stop Services Before Applying an HA Environment" on page 52
- COPAN MAID OpenVault client: "Stop the `cxfs_client` and `openvault` Services Before Applying an HA Environment" on page 168

`cxfs-client` Template

Template location:

`/usr/share/doc/sgi-ha/templates/cxfs-client`

Use the following:

```
primitive CXFS-CLIENT ocf:sgi:cxfs-client \
   op monitor interval="0" timeout="30s" \
   op monitor interval="120s" timeout="30s" on-fail="restart" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="600s" on-fail="fence" \
params volnames="VOLUME-LIST"
```

Variable	Description
CXFS-CLIENT	Name of this resource instance, such as `CXFS-CLIENT`
VOLUME-LIST	Comma-separated list of volume names under the `/dev/cxvm` directory, as appropriate for the HA service:

- **CXFS NFS edge-serving HA service**: the CXFS filesystems to be served via NFS, such as `cxfsvo11,cxfsvol2`
- **COPAN MAID OpenVault client HA service**: all of the managed filesystems and the DMF administrative filesystems represented by the following parameters in the DMF configuration file:

  `CACHE_DIR`
  `SPOOL_DIR`
  `TMP_DIR`
  `MOVE_FS`
  `STORE_DIRECTORY` for a DCM MSP

  For example, suppose you have the following output:

  ```
  # ls /dev/cxvm
  cache         dmfusr2       move
  diskmsp       home          spool
  dmfusr1       journal       tmp
  ```

  Note: In a COPAN MAID OpenVault client HA service, you should not include the `HOME_DIR` or `JOURNAL_DIR` filesystems because a parallel data-mover node typically needs no access to these filesystems.

  You would likely enter the following (everything except `home` and `journal`):

  ```
  cache,diskmsp,dmfusr1,dmfusr2,move,spool,tmp
  ```

For example, for the **CXFS NFS edge-serving HA** service:

```
primitive CXFS-CLIENT ocf:sgi:cxfs-client \
   op monitor interval="0" timeout="30s" \
   op monitor interval="120s" timeout="30s" on-fail="restart" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="600s" on-fail="fence" \
   params volnames="cxfsvol1,cxfsvol2"
```

For example, for the **COPAN MAID OpenVault client HA service**:

```
primitive CXFS-CLIENT ocf:sgi:cxfs-client \
   op monitor interval="0" timeout="30s" \
   op monitor interval="120s" timeout="30s" on-fail="restart" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="600s" on-fail="fence" \
   params volnames="cache,diskmsp,dmfusr1,dmfusr2,move,spool,tmp"
```

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

The `start` operation does the following:

- Starts the CXFS client by calling the following:

  ```
  service cxfs_client start
  ```

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

The `stop` operation does the following:

- Stops the CXFS client by calling the following:

  ```
  service cxfs_client stop
  ```

- Fails if the CXFS client fails to stop

To add more resources, see "Map of Resources for the Edge-Serving HA Service" on page 53.

cxfs-client-clone

This section discusses the following:

- "`cxfs-client-clone` Purpose" on page 249
- "`cxfs-client-clone` Template" on page 249

cxfs-client-clone Purpose

The `clone` resource named by default `CXFS-CLIENT-CLONE` implements control of a CXFS client. It is used in a COPAN MAID OpenVault client HA service. See Chapter 7, "Create the COPAN MAID OpenVault Client HA Service for Mover Nodes" on page 165.

Figure 7-2 on page 173 depicts the order in which this resource falls in the configuration process.

cxfs-client-clone Template

Template location:

```
/usr/share/doc/sgi-ha/templates/cxfs-client-clone
```

Use the following:

```
clone CXFS-CLIENT-CLONE CXFS-CLIENT \
   meta clone-max="2" target-role="Stopped" interleave="true"
```

Variable	Description
CXFS-CLIENT-CLONE	Name of this `clone` instance, such as `CXFS-CLIENT-CLONE`
CXFS-CLIENT	Name of the `cxfs-client` resource instance (see "`cxfs-client` Template" on page 246), such as `CXFS-CLIENT`

For example:

```
clone CXFS-CLIENT-CLONE CXFS-CLIENT \
   meta clone-max="2" target-role="Stopped" interleave="true"
```

The `target-role` is initially set to `Stopped` because you do not want the `clone` to start until you are ready to test it.

To add more resources, see "Map of Resources for the COPAN MAID OpenVault HA Service" on page 170.

`cxfs-client-nfsserver`

This section discusses the following:

- "`cxfs-client-nfsserver` Purpose" on page 251
- "`cxfs-client-nfsserver` Requirements" on page 251
- "Testing Before Applying HA" on page 251
- "HA Control of the Service" on page 251
- "`cxfs-client-nfsserver` Template" on page 252

`cxfs-client-nfsserver` Purpose

A `cxfs-client-nfsserver` resource controls the NFS server running on a CXFS client. It is part of a `clone` resource that runs on both nodes in the cluster. It is used in a CXFS NFS edge-serving HA service. See Chapter 5, "Create the CXFS NFS Edge-Serving HA Service" on page 47.

Figure 5-1 on page 54 depicts the order in which this resource falls in the configuration process.

`cxfs-client-nfsserver` Requirements

See "Edge-Serving Requirements" on page 48.

Testing Before Applying HA

See "Configure and Test the NFS Edge Service Before Applying an HA Environment" on page 50.

HA Control of the Service

To use the `cxfs-client-nfsserver` resource, you must stop the NFS service on each node before applying an HA environment, according to the operating system:

- RHEL:

```
rhel_node1# chkconfig nfs off
rhel_node1# service nfs stop

rhel_node2# chkconfig nfs off
rhel_node2# service nfs stop
```

- SLES:

```
sles_node1# chkconfig nfsserver off
sles_node1# service nfsserver stop

sles_node2# chkconfig nfsserver off
sles_node2# service nfsserver stop
```

The HA software will control this service.

`cxfs-client-nfsserver` Template

Template location:

`/usr/share/doc/sgi-ha/templates/cxfs-client-nfsserver`

Use the following:

```
primitive EDGENFS-A ocf:sgi:cxfs-client-nfsserver \
   op monitor interval="0" timeout="60s" \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="300s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="600s" on-fail="restart" \
   params nfs_init_script="NFS-INIT-PATH" statedir="STATEDIR" \
          statefile="STATEFILE" volnames="VOLUME-LIST" \
          nfslock_init_script="NFSLOCK-INIT-PATH"
```

Variable	**Description**
EDGENFS-A	Name of this resource instance, such as `EDGENFS-A`
NFS-INIT-PATH	Path to the NFS initialization script: • RHEL: `/etc/init.d/nfs`

- SLES: `/etc/init.d/nfsserver`

NFSLOCK-INIT-PATH

Path to the `nfslock` initialization script:

- RHEL: `/etc/init.d/nfslock`
- SLES: (not used)

STATEDIR

Directory in the NFS filesystem that will be used to store NFS file-lock state for this HA cluster (equivalent to `/var/lib/nfs/` in a nonclustered configuration), such as:

```
/mnt/cxfsvol1/statd/nfs1-nfs2
```

Note: This value must be must be unique to this HA cluster.

STATEFILE

File in the NFS filesystem that will store NFS file-lock state for all of the NFS edge-serving HA clusters (equivalent to `/var/lib/nfs/state` in a nonclustered configuration), such as:

```
/mnt/cxfsvol1/statd/state
```

Note: This value must be identical to the *STATEFILE* value specified for `cxfs-client-smnotify`. All HA clusters within a CXFS cluster must have an identical *STATEFILE* value.

VOLUME-LIST

Comma-separated list of volume names containing CXFS filesystems that are to be served via NFS, such as:

```
cxfsvol1
```

For example:

```
primitive cxfs-client-nfsserver-1 ocf:sgi:cxfs-client-nfsserver \
   op monitor interval="0" timeout="60s" \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="300s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="600s" on-fail="restart" \
   params nfs_init_script="/etc/init.d/nfsserver" \
          statedir="/mnt/cxfsvol1/statd/nfs1-nfs2" \
          statefile="/mnt/cxfsvol1/statd/state" volnames="cxfsvol1"
```

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

The `start` operation does the following:

- Creates the *STATEDIR* directory and *STATEFILE* file as needed
- *(NFS v3 only)* 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 *VOLUME-LIST*
- Starts the NFS server:
  - RHEL:

    ```
    service nfs start
    ```
  - SLES:

    ```
    service nfsserver start
    ```
- Fails if the NFS server does not start or if the NLM grace notification cannot be enabled

The `stop` operation does the following:

- Stops the NFS server:
  - RHEL:

    ```
    service nfs stop
    ```

  - SLES:

    ```
    service nfsserver stop
    ```

- Disables NLM grace notification for all volumes in *VOLUME-LIST*
- Fails if the NFS server does not stop or if the NLM grace notification cannot be disabled

To add more resources, see "Map of Resources for the Edge-Serving HA Service" on page 53.

cxfs-client-smnotify

This section discusses the following:

- "cxfs-client-smnotify Purpose" on page 256
- "cxfs-client-smnotify Requirements" on page 256
- "cxfs-client-smnotify Template" on page 256

cxfs-client-smnotify Purpose

A `cxfs-client-smnotify` resource controls NFS client notification of NFS server failovers and restarts. It is used in a CXFS NFS edge-serving HA service as part of a `group` resource that includes an `IPaddr2` resource. See Chapter 5, "Create the CXFS NFS Edge-Serving HA Service" on page 47.

Figure 5-1 on page 54 depicts the order in which this resource falls in the configuration process.

cxfs-client-smnotify Requirements

See the following:

- "Edge-Serving Requirements" on page 48
- "Requirements for a Second Edge-Serving HA Cluster" on page 72

cxfs-client-smnotify Template

The templated is located in:

`/usr/share/doc/sgi-ha/templates/cxfs-client-smnotify`

Use the following:

```
primitive SMNOTIFY-X ocf:sgi:cxfs-client-smnotify \
   op monitor interval="0" timeout="60s" \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="30s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="30s" on-fail="fence" \
   params ipalias="IPADDRESS-ALIAS" statedir="STATEDIR" statefile="STATEFILE" \
          gracedir="GRACEDIR" hostname="IPALIAS-HOST" \
          seconds="120" volnames="VOLUME-LIST" \
   meta resource-stickiness="1" migration-threshold="1"
```

Variable	Description
SMNOTIFY-X	Name of this resource instance, such as `SMNOTIFY-1`
IPALIAS-ADDRESS	IP address of the 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 `128.162.244.244`
STATEDIR	Directory in the NFS filesystem that will be used to store NFS file-lock state for this NFS edge-serving HA cluster. **Note:** This value must be identical to the *STATEDIR* value specified for `cxfs-client-nfsserver`, and must be unique to this HA cluster.
STATEFILE	File in the NFS filesystem that will store NFS file-lock state for all of the NFS edge-serving HA clusters within one CXFS cluster (equivalent to `/var/lib/nfs/state` in a nonclustered configuration), such as: `/mnt/cxfsvol1/statd/state` **Note:** This value must be identical to the *STATEFILE* value specified for `cxfs-client-nfsserver`. All HA clusters within a CXFS cluster must have an identical *STATEFILE* value.
GRACEDIR	Directory on the NFS file-lock-state filesystem that specifies the directory containing the grace-period state (the grace period specified by the `seconds` attribute is the time during which the CXFS tokens owned by a

	given CXFS client will be maintained by that client to permit time for failover), such as: `/mnt/cxfsvol1/grace` **Note:** All HA clusters within a CXFS cluster must have an identical *GRACEDIR* value.
IPALIAS-HOST	Hostname of *IPALIAS-ADDRESS*, which must match what is in `/etc/hosts` or be resolvable with DNS, such as `hostalias1` or `hostalias2`
VOLUME-LIST	Comma-separated list of all volumes that will be served via *IPALIAS-HOST*, such as: `cxfsvol1`

For example:

```
primitive SMNOTIFY-1 ocf:sgi:cxfs-client-smnotify \
   op monitor interval="0" timeout="60s" \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="30s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="30s" on-fail="fence" \
   params ipalias="128.162.244.244" statedir="/mnt/cxfsvol1/statd/nfs1-nfs2" \
          statefile="/mnt/cxfsvol1/statd/state" \
          gracedir="/mnt/cxfsvol1/grace" hostname="myhost" \
          seconds="120" volnames="cxfsvol1" \
   meta resource-stickiness="1" migration-threshold="1"
```

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

The `start` operation does the following:

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

The `stop` operation does the following:

- Starts an NLM grace period for the IP address alias

- Drops all locks associated with the IP address alias
- Fails if locks cannot be dropped

`cxfs-nfs-clone`

This section discusses the following:

- "`cxfs-nfs-clone` Purpose" on page 260
- "`cxfs-nfs-clone` Template" on page 260

`cxfs-nfs-clone` Purpose

A `clone` resource named by default `CXFS-NFS-CLONE` implements a CXFS NFS edge-serving HA service. It consists of a group (such as `CXFS-NFS-GROUP`) constructed of two resources (such as `CXFS-CLIENT` and `NFS`). See Chapter 5, "Create the CXFS NFS Edge-Serving HA Service" on page 47.

Figure 5-1 on page 54 depicts the order in which this resource falls in the configuration process.

`cxfs-nfs-clone` Template

Template location:

```
/usr/share/doc/sgi-ha/templates/cxfs-nfs-clone
```

Use the following:

```
group CXFS-NFS-GROUP CXFS-CLIENT NFS
clone CXFS-NFS-CLONE CXFS-NFS-GROUP \
   meta clone-max="2" target-role="Stopped" interleave="true"
```

Variable	**Description**
CXFS-NFS-GROUP	Name of this `group` instance, such as `CXFS-NFS-GROUP`
CXFS-CLIENT	Name of the `cxfs-client` resource instance (see "`cxfs-client` Template" on page 246), such as `CXFS-CLIENT`
NFS	Name of the `nfsserver` resource instance (see "`nfsserver` Template" on page 296), such as `NFS`

CXFS-NFS-CLONE	Name of this `clone` instance, such as `CXFS-NFS-CLONE`

For example:

```
group CXFS-NFS-GROUP CXFS-CLIENT NFS
clone CXFS-NFS-CLONE CXFS-NFS-GROUP \
   meta clone-max="2" target-role="Stopped" interleave="true"
```

dmcopytool

This section discusses the following:

- "dmcopytool Purpose" on page 262
- "dmcopytool Requirements" on page 262
- "Testing Before Applying HA" on page 262
- "dmcopytool Template" on page 262

dmcopytool Purpose

A dmcopytool resource controls one instance of the SGI DMF copytool (lhsmtool_dmf), which is a version of the Lustre copytool (lhsmtool_posix) that has been tuned to work efficiently with the DMF archive system. Its primary function is to copy files between the Lustre filesystem and the DMF archive system. It may optionally be used in a DMF HA service.

dmcopytool Requirements

See "DMF Copytool Requirements" on page 84.

Testing Before Applying HA

See "Configure and Test Lustre Before Applying the DMF Copytool an HA Environment" on page 92

dmcopytool Template

The templated is located in:

`/usr/share/doc/sgi-ha/templates/dmcopytool`

Use the following:

```
primitive DMCOPYTOOL ocf:sgi:dmcopytool \
  op monitor interval="0" timeout="60s" \
  op monitor interval="120s" timeout="60s" on-fail="restart" \
  op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
  op stop interval="0" timeout="600s" on-fail="fence" \
  params archive="NUM" hsmroot="PATH" lustredevice="DEVICE" \
   dmctmount="DMCT-MOUNT" dmfmount="DMF-MOUNT" \
  meta resource-stickiness="1" migration-threshold="1"
```

Variable	**Description**
DMCOPYTOOL	Name of this resource instance, such as `DMCOPYTOOL`
NUM	The archive number registered with the Lustre HSM coordinator for this instance of the copytool. Each Lustre filesystem using the DMF copytool should be given its own unique archive number.
PATH	Path of a directory within a filesystem managed by DMF that is used for archiving or restoring Lustre files. This directory must already exist.
DEVICE	The Lustre filesystem device to be mounted for use by the DMF copytool.
DMCT-MOUNT	The Lustre filesystem mount point that will be used by the DMF copytool.
DMF-MOUNT	The Lustre filesystem mount point that will be used by DMF. This mount point must match the name of a `filesystem` object defined in the DMF configuration file with a `MIGRATION_LEVEL` parameter set to `archive`.

For example:

```
primitive DMCOPYTOOL ocf:sgi:dmcopytool \
  op monitor interval="0" timeout="60s" \
  op monitor interval="120s" timeout="60s" on-fail="restart" \
  op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
  op stop interval="0" timeout="600s" on-fail="fence" \
  params archive="2" hsmroot="/usr3" lustredevice="128.162.245.79@tcp0:/lustrefs" \
  dmctmount="/lfs_dmct" dmfmount="/lfs_dmf" \
  meta resource-stickiness="1" migration-threshold="1"
```

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

The `start` operation does the following:

- Mounts the Lustre filesystem to two mount points, defined by the *DMCT-MOUNT* and *DMF-MOUNT* values
- Starts the `lhsmtool_dmf` command
- Fails if either of the two `mount` operations fails or if the tool cannot be started

The `stop` operation does the following:

- Stops the `lhsmtool_dmf` command
- Unmounts the Lustre filesystem from the two mount points defined by the *DMCT-MOUNT* and *DMF-MOUNT* values
- Fails if tool could not be stopped or if either of the two `umount` operations fails

`dmf`

This section discusses the following:

- "`dmf` Purpose" on page 265
- "`dmf` Requirements" on page 265
- "Testing Before Applying HA" on page 265
- "HA Control of the Service" on page 266
- "`dmf` Template" on page 266

`dmf` Purpose

A `dmf` resource starts/stops DMF. It is used in a DMF HA service. See Chapter 6, "Create the DMF HA Service" on page 75.

Figure 6-5 on page 136 depicts the order in which this resource falls in the configuration process.

`dmf` Requirements

See "DMF Requirements" on page 81:

- "Potential DMF Server Requirements" on page 82
- "Ordering of Resources" on page 83
- "Virtual Hostname Requirement" on page 83
- "Parallel Data-Mover Option Requirements" on page 83
- "Control of the `dmf` Service" on page 83
- "Use of the Wildcard in OpenVault Configuration" on page 83

Testing Before Applying HA

See "Configure and Test DMF Before Applying an HA Environment" on page 89.

HA Control of the Service

To use the `dmf` resource, you must stop the `dmf` service on each node before applying an HA environment:

```
node1# chkconfig dmf off
node1# service dmf stop

node2# chkconfig dmf off
node2# service dmf stop
```

The HA software will control this service.

Note: There are other associated services that you must stop when using a DMF HA environment. See "Stop Services Related to DMF Before Applying an HA Environment" on page 95.

`dmf` Template

Template location:

```
/usr/share/doc/sgi-ha/templates/dmf
```

Use the following:

```
primitive DMF ocf:sgi:dmf \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   params monitor_level="0" \
   meta resource-stickiness="1" migration-threshold="1"
```

Variable	**Description**
DMF	Name of this resource instance, such as `DMF`

Note: In a CXFS environment, ensure that the `timeout` values are appropriate for your site; you must account for the time required to relocate the CXFS metadata server for the managed filesystems.

For example:

```
primitive DMF ocf:sgi:dmf \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   params monitor_level="0" \
   meta resource-stickiness="1" migration-threshold="1"
```

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

The `start` operation does the following:

- Starts DMF by calling the following:

  ```
  service 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

The `stop` operation does the following:

- Stops DMF by calling the following:

  ```
  service dmf stop
  ```

- Issues a `dmclrmount` command
- Fails if DMF could not be stopped

dmf-group

This section discusses the following:

- "dmf-group Purpose" on page 268
- "dmf-group Template" on page 268

dmf-group Purpose

The DMF HA implementation uses a single `group` resource that contains all of the other required resources, so that HA processes can coherently control all of the resources, starting and stopping them in the required order. The name of the group by default is `DMF-GROUP`, but it can be any name you choose. See Chapter 6, "Create the DMF HA Service" on page 75.

Figure 6-2 on page 101 depicts the order in which this resource falls in the configuration process.

dmf-group Template

Template location:

`/usr/share/doc/sgi-ha/templates/dmf-group`

Following are some sample `group` definitions, including optional resources:

- A CXFS environment using OpenVault:

```
group DMF-GROUP CXFS IP OV DMF NFSSERVER etc_samba var_lib_samba SMB NMB DMFMAN DMFSOAP
```

- A CXFS environment using TMF:

```
group DMF-GROUP CXFS IP TMF DMF NFSSERVER etc_samba var_lib_samba SMB NMB DMFMAN DMFSOAP
```

- A local XVM environment using OpenVault:

```
group DMF-GROUP LXVM dmfusr1 dmfusr2 home spool move_fs journals \
                tmp move_fs cache store IP OV DMF NFSSERVER etc_samba var_lib_samba \
                SMB NMB DMFMAN DMFSOAP
```

- A local XVM environment using TMF:

```
group DMF-GROUP LXVM dmfusr1 dmfusr2 home spool move_fs journals \
                tmp move_fs cache store IP TMF DMF NFSSERVER etc_samba var_lib_samba \
                SMB NMB DMFMAN DMFSOAP
```

dmfman

This section discusses the following:

- "dmfman Purpose" on page 270
- "dmfman Requirements" on page 270
- "Testing Before Applying HA" on page 270
- "HA Control of the Service" on page 270
- "dmfman Template" on page 271

dmfman Purpose

The dmfman resource controls DMF Manager. It may optionally be used in a DMF HA service. See Chapter 6, "Create the DMF HA Service" on page 75.

Figure 6-9 on page 159 depicts the order in which this resource falls in the configuration process.

dmfman Requirements

See "DMF Manager Requirements *(Optional)*" on page 84.

Testing Before Applying HA

See "Test DMF Manager Before Applying an HA Environment" on page 95.

HA Control of the Service

To use the dmfman resource, you must stop the dmfman service on both nodes:

```
node1# chkconfig dmfman off
node1# chkconfig dmfman off

node2# chkconfig dmfman off
node2# chkconfig dmfman off
```

The HA software will control this service.

Note: There are other associated services that you must stop. See "Stop Services Related to DMF Before Applying an HA Environment" on page 95.

`dmfman` Template

Template location:

`/usr/share/doc/sgi-ha/templates/dmfman`

Use the following:

```
primitive DMFMAN ocf:sgi:dmfman \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   meta resource-stickiness="1" migration-threshold="100"
```

Variable	Description
DMFMAN	Name of this resource instance, such as `DMFMAN`

For example:

```
primitive DMFMAN ocf:sgi:dmfman \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   meta resource-stickiness="1" migration-threshold="100"
```

The `monitor` operation probes to see if the resource is already running.

The `start` operation does the following:

- Starts DMF Manger by calling the following:

  `service dmfman start`

- Waits for DMF Manager to start successfully by calling the following in a loop:

  `service dmfman status`

- Fails if DMF Manager does not start successfully before the resource times out

The `stop` operation does the following:

- Stops DMF Manger by calling the following:

  ```
  service dmfman stop
  ```

- Verifies the DMF Manager status by calling the following:

  ```
  service dmfman status
  ```

- Fails if DMF Manager does not stop successfully

dmfsoap

This section discusses the following:

- "dmfsoap Purpose" on page 273
- "dmfsoap Requirements" on page 273
- "Testing Before Applying HA" on page 273
- "HA Control of the Service" on page 273
- "dmfsoap Template" on page 274

dmfsoap Purpose

The `dmfsoap` resource controls the DMF client Simple Object Access Protocol (SOAP) service. It may optionally be used in a DMF HA service. See Chapter 6, "Create the DMF HA Service" on page 75.

Figure 6-10 on page 162 depicts the order in which this resource falls in the configuration process.

dmfsoap Requirements

See "DMF Client SOAP Service Requirements *(Optional)*" on page 85.

Testing Before Applying HA

See "Test the DMF Client SOAP Service Before Applying an HA Environment" on page 95.

HA Control of the Service

To use the `dmfsoap` resource, you must stop the `dmfsoap` service on both nodes:

```
node1# chkconfig dmfsoap off
node1# chkconfig dmfsoap off

node2# chkconfig dmfsoap off
node2# chkconfig dmfsoap off
```

The HA software will control this service.

Note: There are other associated services that you must stop. See "Stop Services Related to DMF Before Applying an HA Environment" on page 95.

`dmfsoap` Template

Template location:

```
/usr/share/doc/sgi-ha/templates/dmfsoap
```

Use the following:

```
primitive DMFSOAP ocf:sgi:dmfsoap \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   meta resource-stickiness="1" migration-threshold="100"
```

Variable	**Description**
DMFSOAP	Name of this resource instance, such as `DMFSOAP`

For example:

```
primitive DMFSOAP ocf:sgi:dmfsoap \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   meta resource-stickiness="1" migration-threshold="100"
```

The `monitor` operation probes to see if the resource is already running.

The `start` operation does the following:

- Starts the DMF client SOAP service by calling the following:

  ```
  service dmfsoap start
  ```

- Waits for DMF client SOAP service to start successfully by calling the following in a loop:

  ```
  service dmfsoap status
  ```

- Fails if DMF client SOAP service does not start successfully before the resource times out

The `stop` operation does the following:

- Stops the DMF client SOAP service by calling the following:

  ```
  service dmfsoap stop
  ```

- Verifies the DMF client SOAP service status by calling the following:

  ```
  service dmfsoap status
  ```

- Fails if the DMF client SOAP service does not stop successfully

`fence_ipmilan`

This section discusses the following:

- "`fence_ipmilan` Purpose" on page 276
- "Testing Before Applying HA" on page 276
- "`fence_ipmilan` Template" on page 276

`fence_ipmilan` Purpose

A `fence_ipmilan` resource implements STONITH for one RHEL node in the cluster. It may be used in any HA service for a RHEL cluster. See "RHEL STONITH" on page 185.

Note: This resource does not apply to SLES nodes.

Testing Before Applying HA

Verify that the IP address for the BMC of the node that this resource will control is accessible, such as by using the `ping`(1) command.

`fence_ipmilan` Template

Note: This resource provides STONITH for RHEL nodes. It does not apply to SLES nodes.

Template location:

`/usr/share/doc/sgi-ha/templates/fence_ipmilan`

Use the following:

```
primitive STONITH-NODE stonith:fence_ipmilan \
   params pcmk_host_list="NODE" ipaddr="BMC-IP" login="admin" \
          passwd="admin" delay="DELAY"
location STONITH-NODE-LOCATION STONITH-NODE -inf: NODE
```

Variable	Description
STONITH-NODE	Name of this resource instance, such as `STONITH-node1`
NODE	Hostname of the RHEL node that this resource will control, such as `node1`
BMC-IP	The IP address for the BMC for *NODE*, such as `128.162.245.197`
DELAY	The number of seconds by which the action should be delayed. The nodes must have different values, such as `0` and `15`, so that they will not begin the reset process at the same time. This will ensure that both nodes cannot power each other off nearly simultaneously, so that neither is remaining to turn the other back on.
STONITH-NODE-LOCATION	Name of this `location` instance if you want to use something other than the default

Note: The values shown for the BMC user ID and password are typical, but you must verify the values at your site.

Ensure that the `acpid` service is turned off and will not restart.

For example, for two nodes (`node1` and `node2`):

```
primitive STONITH-node1 stonith:fence_ipmilan \
   params pcmk_host_list="node1" ipaddr="128.162.245.197" login="admin" \
          passwd="admin" delay="0"
primitive STONITH-node2 stonith:fence_ipmilan \
   params pcmk_host_list="node2" ipaddr="128.162.245.199" login="admin" \
          passwd="admin" delay="15"
location STONITH-node1-LOCATION STONITH-node1 -inf: node1
location STONITH-node2-LOCATION STONITH-node2 -inf: node2
```

The `location` constraints ensure that the `STONITH-node1` resource will never be located on `node1` and that the `STONITH-node2` resource will never be located on `node2`. This ensures that the power reset is performed via IPMI from another system.

`Filesystem`

This section discusses the following:

- "`Filesystem` Purpose" on page 279
- "`Filesystem` Requirements" on page 279
- "Testing Before Applying HA" on page 280
- "`Filesystem` Template" on page 280

`Filesystem` Purpose

A `Filesystem` resource implements the mount control for a single filesystem.

A `Filesystem` resource is used the following HA services:

- DMF HA service in a local XVM environment for the following:
  - Filesystems managed by DMF
  - DMF administrative filesystems
  - Dedicated filesystem for the OpenVault `serverdir` directory (optional)
  - With Samba, for the Samba directories that are `bind`-mounted on `/etc/samba` and `/var/lib/samba`
- DMF HA service with Samba in a CXFS environment, for the Samba directories that are `bind`-mounted on `/etc/samba` and `/var/lib/samba`

Note: The `Filesystem` resources used with Samba require `fstype="none"`, which differs from the template below, and `options="bind"`.

Figure 6-2 on page 101 depicts the order in which this resource falls in the configuration process.

`Filesystem` Requirements

See "Local Filesystem Requirements" on page 79.

Testing Before Applying HA

See "Configure and Test Local XVM Before Applying an HA Environment" on page 86.

`Filesystem` Template

Template location:

`/usr/share/doc/sgi-ha/templates/Filesystem`

Use the following:

```
primitive FILESYSTEM ocf:heartbeat:Filesystem \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   params device="DEVICE-FILE" directory="MOUNT-POINT" fstype="xfs" \
          options="MOUNT-OPTIONS" \
   meta resource-stickiness="1" migration-threshold="1"
```

Variable	**Description**
FILESYSTEM	Name of this resource instance, such as: • `dmfusr1` for a managed filesystem • `dmf_spool_fs` for the DMF administrative filesystem defined by the `SPOOL_DIR` parameter in the DMF configuration file • `openvaultfs` for the filesystem to use as the OpenVault `serverdir` directory
DEVICE-FILE	The `/dev/lxvm` volume name of the DMF filesystem device, such as `/dev/lxvm/dmfusr1`, `/dev/lxvm/dmf_spool`, or `/mnt/data/.ha/etc_samba`
MOUNT-POINT	The mount point for the DMF filesystem, such as `/dmfusr1`, `/dmf/dmf_spool`, or `/etc/samba`. This mount point must already exist on all nodes in the HA cluster before you load this text into the CIB.
MOUNT-OPTIONS	The mount options for the filesystem.

Note: Filesystems that must be mounted with the `dmi` mount option should also have an `mtpt` mount option whose value matches the filesystem's *MOUNT_POINT* value. This includes the *MOVE_FS* filesystem and all filesystems with `filesystem` stanzas (other than those with a `MIGRATION_LEVEL` setting of `archive`) in the DMF configuration file.

The following provide examples of `Filesystem` resources:

- "`Filesystem` Example for a Managed Filesystem" on page 281
- "`Filesystem` Example for the Samba Directory `bind`-Mounted on `/etc/samba`" on page 282

`Filesystem` Example for a Managed Filesystem

The following examples for a managed filesystem:

```
primitive dmfusr1 ocf:heartbeat:Filesystem \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   params device="/dev/lxvm/dmfusr1" directory="/dmfusr1" fstype="xfs" \
          options="rw" \
   meta resource-stickiness="1" migration-threshold="1"
```

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

The `start` operation does the following:

- Mounts the filesystem
- Fails if the mount is unsuccessful

The `stop` operation does the following:

- Unmounts the filesystem
- Fails if the unmount is unsuccessful

Filesystem Example for the Samba Directory bind-Mounted on /etc/samba

The following example is for the Samba directory `bind`-mounted on `/etc/samba`:

```
primitive etc-samba ocf:heartbeat:Filesystem \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   params device="/mnt/data/.ha/etc-samba" directory="/etc/samba" fstype="none" options="bind\
   meta resource-stickiness="1" migration-threshold="1"
```

The operations and parameters here are similar to the previous example for the managed filesystem, except that this `Filesystem` resource requires `fstype="none"` and `options="bind"`.

`IPaddr2`

This section discusses the following:

- "`IPaddr2` Purpose" on page 283
- "`IPaddr2` Requirements" on page 283
- "`IPaddr2` Template" on page 283

`IPaddr2` Purpose

The `IPaddr2` resource controls the addition/deletion of an IP address alias on a network interface. It is used in the CXFS NFS edge-serving HA service and in the DMF HA service (in either a CXFS environment or a local XVM environment). See:

- Chapter 5, "Create the CXFS NFS Edge-Serving HA Service" on page 47
- Chapter 6, "Create the DMF HA Service" on page 75

The following figures depict the order in which this resource falls in the configuration process:

- Figure 5-1 on page 54
- Figure 6-3 on page 107

`IPaddr2` Requirements

See:

- Edge-serving: "IP Address Alias Requirements in an Edge-Serving HA Cluster" on page 49
- DMF: "IP Address Alias Requirements" on page 79

`IPaddr2` Template

Use the following:

```
primitive IP ocf:heartbeat:IPaddr2 \
   op monitor interval="0" timeout="30s" \
   op start interval="0" timeout="90s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="100s" on-fail="fence" \
   params ip="IP-ADDRESS" \
   meta resource-stickiness="1" migration-threshold="1"
```

Variable	Description
IP	Name of this resource instance, which must match the named used to define it in the associated `group` resource (for example, `IP` used DMF HA example in this guide or `IP-1` or `IP-2` used in the CXFS NFS edge-serving example in this guide).
IP-ADDRESS	IP address of the virtual channel, such as `128.162.244.240`

For example:

```
primitive IP-1 ocf:heartbeat:IPaddr2 \
   op monitor interval="0" timeout="30s" \
   op start interval="0" timeout="90s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="100s" on-fail="fence" \
   params ip="128.162.244.240" \
   meta resource-stickiness="1" migration-threshold="1"
```

The `monitor` operation probes to see if the resource is already running. Normally, this is the only `monitor` operation required for an `IPaddr2` resource.

The `start` operation does the following:

- Establishes the IP address alias
- Fails if the IP address alias is not established

The `stop` operation does the following:

- Removes the IP address alias
- Fails if the IP address alias is not removed

ipalias-group

This section discusses the following:

- "ipalias-group Purpose" on page 285
- "ipalias-group Template" on page 285

ipalias-group Purpose

A group resource named IPALIAS-GROUP controls the IPaddr2 and cxfs-client-smnotify resources. You will have a pair of IPALIAS-GROUP resources, each with their own set of colocation and order constraints. It is used in a CXFS NFS edge-serving HA service. See Chapter 5, "Create the CXFS NFS Edge-Serving HA Service" on page 47.

Figure 5-1 on page 54 depicts the order in which this resource falls in the configuration process.

ipalias-group Template

Use the following:

```
group IPALIAS-GROUP-X IP-X SMNOTIFY-X \
   meta target-role="Stopped"
colocation IPALIAS-WITH-NFS-X inf: IPALIAS-GROUP-X CXFS-NFS-CLONE
order NFS-BEFORE-IPALIAS-X inf: CXFS-NFS-CLONE IPALIAS-GROUP-X
```

Variable	**Description**
IPALIAS-GROUP-X	Name of this resource instance, such as IPALIAS-GROUP-1
IP-X	Name of the IPaddr2 resource instance (see "IPaddr2 Template" on page 283), such as IP-1
SMNOTIFY-X	Name of the cxfs-client-smnotify resource instance (see "cxfs-client-smnotify Template" on page 256), such as SMNOTIFY-1

IPALIAS-WITH-NFS-X	Name of this `colocation` constraint, such as `IPALIAS-WITH-NFS-1`
CXFS-NFS-CLONE	Name of the NFS `clone` (see "`cxfs-nfs-clone` Template" on page 260) such as `CXFS-NFS-CLONE`
NFS-BEFORE-IPALIAS-X	Name of this `order` constraint, such as `NFS-BEFORE-IPALIAS-1`

For example:

- For the first group:

```
group IPALIAS-GROUP-1 IP-1 SMNOTIFY-1 \
   meta target-role="Stopped"
colocation IPALIAS-WITH-NFS-1 inf: IPALIAS-GROUP-1 CXFS-NFS-CLONE
order NFS-BEFORE-IPALIAS-1 inf: CXFS-NFS-CLONE IPALIAS-GROUP-1
```

- For the second group:

```
group IPALIAS-GROUP-2 IP-2 SMNOTIFY-2 \
   meta target-role="Stopped"
colocation IPALIAS-WITH-NFS-2  inf: IPALIAS-GROUP-2 CXFS-NFS-CLONE
order NFS-BEFORE-IPALIAS-2 inf: CXFS-NFS-CLONE IPALIAS-GROUP-2
```

The `colocation` and `order` constraints ensure that `IPALIAS-GROUP-1` will only run on a server that is also running a `CXFS-NFS-CLONE` instance, and the `clone` instance will be started first.

`ipmi`

This section discusses the following:

- "`ipmi` Purpose" on page 287
- "Test the BMC IP Address Before Applying an HA Environment" on page 287
- "`ipmi` Template" on page 287

`ipmi` Purpose

An `ipmi` resource implements STONITH for one SLES node in the cluster.

Note: This resource does not apply to RHEL nodes.

Test the BMC IP Address Before Applying an HA Environment

Verify that the IP address for the BMC of the node that this resource will control is accessible, such as by using the `ping`(1) command.

`ipmi` Template

Note: This resource provides STONITH for SLES nodes. It does not apply to RHEL nodes.

Template location:

`/usr/share/doc/sgi-ha/templates/ipmi`

Use the following:

```
primitive STONITH-NODE stonith:external/ipmi \
   op monitor interval="0" timeout="60s" \
   op monitor interval="300s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="60s" on-fail="restart" \
   params hostname="NODE" ipaddr="BMC-IP" userid="admin" \
          passwd="admin" interface="lan"
```

`location` ***STONITH-NODE-LOCATION STONITH-NODE*** `-inf:` ***NODE***

Variable	**Description**
STONITH-NODE	Name of this resource instance, such as `STONITH-node1`
NODE	Name of the SLES node that this resource will control, such as `node1`
BMC-IP	The IP address for the BMC of the node this resource will control, such as `128.162.232.79`.
STONITH-NODE-LOCATION	Name of the `location` constraint if you want to use something other than the default

---

Note: You should verify that BMC `userid`, `passwd`, and `interface` values are correct for your site. (The BMC hardware at some sites may require the `lanplus` value for `interface`.)

For example:

```
primitive STONITH-node1 stonith:external/ipmi \
   op monitor interval="0" timeout="60s" \
   op monitor interval="300s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="60s" on-fail="restart" \
   params hostname="node1" ipaddr="128.162.232.79" userid="admin" \
          passwd="admin" interface="lan"
primitive STONITH-node2 stonith:external/ipmi \
   op monitor interval="0" timeout="60s" \
   op monitor interval="300s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="60s" on-fail="restart" \
   params hostname="node1" ipaddr="128.162.246.63" userid="admin" \
          passwd="admin" interface="lan"
location STONITH-node1-LOCATION STONITH-node1 -inf: node1
location STONITH-node2-LOCATION STONITH-node2 -inf: node2
```

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

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

The `location` constraint ensures that the `STONITH-node1` resource will never be located on `node1`. This ensures that the power reset is performed via IPMI from some other system.

lxvm

This section discusses the following:

- "lxvm Purpose" on page 290
- "lxvm Requirements" on page 290
- "Testing Before Applying HA" on page 290
- "lxvm Template" on page 290

lxvm Purpose

An lxvm resource implements node ownership control for local XVM volumes. It is used in a DMF HA service. See Chapter 6, "Create the DMF HA Service" on page 75

Figure 6-2 on page 101 depicts the order in which this resource falls in the configuration process.

lxvm Requirements

See "Local XVM Requirements" on page 78.

Testing Before Applying HA

See "Configure and Test Local XVM Before Applying an HA Environment" on page 86.

lxvm Template

Template location:

/usr/share/doc/sgi-ha/templates/lxvm

Use the following:

```
primitive LXVM ocf:sgi:lxvm \
   op monitor interval="120s" timeout="180s" on-fail="restart" \
   op monitor interval="0" timeout="180s" \
   op start interval="0" timeout="900s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="900s" on-fail="fence" \
   params physvols="PHYSVOL-LIST" volnames="VOLUME-LIST" \
   meta resource-stickiness="1" migration-threshold="1"
```

Variable	Description
LXVM	Name of this resource instance, such as `LXVM`
PHYSVOL-LIST	Comma-separated list of the physical volumes for the resource agent to steal, such as: `myCluster,myClusterStripe1,myClusterStripe2` **Note:** *PHYSVOL-LIST* must contain all of the physical volumes for every logical volume listed in *VOLUME-LIST*. All physical disks that belong to a logical volume in an HA cluster must be completely dedicated to that logical volume and no other.
VOLUME-LIST	Comma-separated list of volume names under `/dev/lxvm` to monitor, such as: `openvault,home,journals,spool,move,tmp,diskmsp,dmfusr1,dmfusr2`

Note: A 900-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`.

For example:

```
primitive LXVM ocf:sgi:lxvm \
   op monitor interval="120s" timeout="180s" on-fail="restart" \
   op monitor interval="0" timeout="180s" \
   op start interval="0" timeout="900s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="900s" on-fail="fence" \
   params physvols="myCluster,myClusterStripe1,myClusterStripe2" volnames="openvault,home,\
                    journals,spool,move,tmp,diskmsp,dmfusr1,dmfusr2" \
   meta resource-stickiness="1" migration-threshold="1"
```

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

The `start` operation does the following:

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

The `stop` operation does the following:

- Gives all physical volumes in *PHYSVOL-LIST* 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 *PHYSVOL-LIST* could not be given away

MailTo

This section discusses the following:

- "MailTo Purpose" on page 293
- "Testing Before Applying HA" on page 293
- "MailTo Template" on page 293

MailTo Purpose

A MailTo resource implements failover notification for a given resource. It may be used in any HA service. A message is sent whenever the specified resource starts or stops.

Testing Before Applying HA

Before applying an HA environment, ensure that the email addresses are valid.

MailTo Template

Template location:

/usr/share/doc/sgi-ha/templates/MailTo

Use the following:

```
primitive NOTIFY ocf:heartbeat:MailTo \
   params email="EMAIL-ADDRESS" subject="RESOURCE-TO-TRACK"
colocation NOTIFIER inf: RESOURCE-TO-TRACK NOTIFY
```

Variable	**Description**
NOTIFY	Name of this resource instance, such as NOTIFY
EMAIL-ADDRESS	The address to be sent notifications
RESOURCE-TO-TRACK	Name of the resource to be tracked, which may be an individual resource such as dmf or a group resource such as DMF-GROUP (see "dmf-group Template" on page 268)

NOTIFIER	Name of the `colocation` constraint, such as `NOTIFIER`

For example:

```
primitive NOTIFY ocf:heartbeat:MailTo \
   params email="admin@mycompany.com" subject="DMF-GROUP"
colocation NOTIFIER inf: DMF-GROUP NOTIFY
```

`nfsserver`

This section discusses the following:

- "`nfsserver` Purpose" on page 295
- "Testing Before Applying HA" on page 295
- "HA Control of the Service" on page 295
- "`nfsserver` Template" on page 296

`nfsserver` Purpose

The `nfsserver` resource controls the start/stop of NFS. It may be used in a DMF HA service. See Chapter 6, "Create the DMF HA Service" on page 75.

Figure 6-6 on page 142 depicts the order in which this resource falls in the configuration process.

Testing Before Applying HA

See "Configure and Test the NFS Service for DMF Use Before Applying an HA Environment" on page 89

HA Control of the Service

Using NFS with HA software requires the following:

- On all HA nodes during HA operation, disable the NFS service from being started automatically at boot time, according to the operating system:
  - RHEL:

    ```
    rhel_node1# chkconfig nfs off
    rhel_node1# service nfs stop

    rhel_node2# chkconfig nfs off
    rhel_node2# service nfs stop
    ```

– SLES:

```
sles_node1# chkconfig nfsserver off
sles_node1# service nfsserver stop

sles_node2# chkconfig nfsserver off
sles_node2# service nfsserver stop
```

The HA software will control the service.

`nfsserver` Template

Template location:

`/usr/share/doc/sgi-ha/templates/nfsserver`

Use the following:

```
primitive NFSSERVER ocf:heartbeat:nfsserver \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   params nfs_shared_infodir="STATEDIR" nfs_ip="IP ADDRESS-ALIAS" \
   meta resource-stickiness="1" migration-threshold="1"
```

Variable	**Description**
NFS	Name of this resource instance, such as `NFS`.
STATEDIR	Directory in the NFS filesystem that will be used to store NFS file-lock state for this HA cluster, such as the `/mnt/cxfsvol1/.nfs` subdirectory in the exported filesystem.
IP ADDRESS-ALIAS	IP address 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 this `nfsserver` resource), for example `128.162.244.244`. This IP address alias will be the same IP address specified for

IP-ADDRESS in one of the `IPaddr2` resources (see "`IPaddr2` Template" on page 283).

For example:

```
primitive NFSSERVER ocf:heartbeat:nfsserver \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="120s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="120s" on-fail="fence" \
   params nfs_shared_infodir="/mnt/cxfsvol1/.nfs" nfs_ip="128.162.232.79" \
   meta resource-stickiness="1" migration-threshold="1"
```

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

The `start` operation does the following:

- Starts the NFS server:
  - RHEL:

    ```
    service nfs start
    ```

  - SLES:

    ```
    service nfsserver start
    ```

- Notifies clients by calling the `nfs_notify_cmd` command
- Fails if the NFS server does not start

The `stop` operation does the following:

- Stops the NFS server:
  - RHEL:

    ```
    service nfs stop
    ```

  - SLES:

    ```
    service nfsserver stop
    ```

- Fails if the NFS server does not stop

nmb

This section discusses the following:

- "`nmb` Purpose" on page 298
- "`nmb` Requirements" on page 298
- "HA Control of the Service" on page 298
- "`nmb` Template" on page 298

nmb Purpose

An `nmb` resource controls the `nmbd` service for Samba. It may be used in a DMF HA service. See Chapter 6, "Create the DMF HA Service" on page 75

Figure 6-7 on page 147 depicts the order in which this resource falls in the configuration process.

nmb Requirements

See "Samba Requirements *(Optional)*" on page 84.

HA Control of the Service

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

```
ha# chkconfig nmb off
```

The HA software will control this service.

nmb Template

Template location:

```
/usr/share/doc/sgi-ha/templates/nmb
```

Use the following:

```
primitive NMB lsb:nmb \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="60s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="60s" on-fail="fence" \
   meta resource-stickiness="1" migration-threshold="1"
```

Variable	**Description**
NMB	Name of this resource instance, such as NMB.

For example:

```
primitive NMB lsb:nmb \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="60s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="60s" on-fail="fence" \
   meta resource-stickiness="1" migration-threshold="1"
```

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

The `start` operation does the following:

- Starts the `nmbd` service by calling the following:

  ```
  service nmb start
  ```

- Fails if the `nmbd` service does not start

The `stop` operation does the following:

- Stops the `nmbd` service by calling the following:

  ```
  service nmb stop
  ```

- Fails if the `nmbd` service does not stop

openvault

This section discusses the following:

- "openvault Purpose" on page 300
- "openvault Requirements" on page 300
- "Testing Before Applying HA" on page 300
- "HA Control of the Service" on page 300
- "openvault Template" on page 301

openvault Purpose

An openvault resource controls the start/stop of all OpenVault processes. It is used in a DMF HA service. See Chapter 6, "Create the DMF HA Service" on page 75.

Figure 6-4 on page 111 depicts the order in which this resource falls in the configuration process.

openvault Requirements

See "OpenVault Requirements" on page 80.

Testing Before Applying HA

See "Configure and Test OpenVault Before Applying an HA Environment" on page 87.

HA Control of the Service

To use the openvault resource, you must stop the openvault service on each node before applying HA:

```
node1# chkconfig openvault off
node1# service openvault stop

node2# chkconfig openvault off
node2# service openvault stop
```

The HA software will control this service.

`openvault` Template

Template location:

`/usr/share/doc/sgi-ha/templates/openvault`

Use the following:

```
primitive OV ocf:sgi:openvault \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="300s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="90s" on-fail="fence" \
   params virtualhost="VIRTUALHOSTVALUE" serverdir="SERVERDIR" \
   meta resource-stickiness="1" migration-threshold="1" is-managed="false"
```

Variable	Description
OV	Name of this resource instance, such as `OV`.
VIRTUALHOSTVALUE	Hostname where the OpenVault server will be listening (which must also have its own `IPaddr2` resource instance, see "`IPaddr2` Template" on page 283).
SERVERDIR	The directory that will eventually contain the OpenVault server configuration. *SERVERDIR* could be a directory that will be dedicated for OpenVault use (such as `/dmf/openvault`) or it could be an HA filesystem in the same resource group that has sufficient space (such as `/dmf/home`) to contain the subdirectory (such as `/dmf/home/openvault`) and its contents. The filesystem must be either: • A directory that will become a mountable CXFS filesystem managed by a `cxfs` resource • A directory that will become a mountable XFS filesystem managed by a `Filesystem` resource in the same resource group as the `openvault` resource

Note: As part of the conversion to an HA environment, OpenVault will create this directory and move its database and logs into the directory; OpenVault will fail if the directory already exists.

For example, if you define *SERVERDIR* as `/dmf/home/openvault`, the parent directory `/dmf/home` directory can exist but the entire path `/dmf/home/openvault` **must not** exist.

For example:

```
primitive OV ocf:sgi:openvault \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="300s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="90s" on-fail="fence" \
   params virtualhost="myvirtualhost" serverdir="/dmf/home/openvault" \
   meta resource-stickiness="1" migration-threshold="1" is-managed="false"
```

Note: For the initial configuration process, the setting for the `is-managed` attribute must be `false` as shown above. The step in section "Make the `openvault` Resource HA-Managed" on page 127 will reset this attribute to `true` so that the resource will run under HA control.

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

The `start` operation does the following:

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

  ```
  ov_start server clients
  ```
- Fails if either *SERVERDIR* or *VIRTUALHOSTVALUE* is unavailable, or if OpenVault does not start

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

ping

This section discusses the following:

- "ping Purpose" on page 304
- "Testing Before Applying HA" on page 304
- "ping Template" on page 304

ping Purpose

The HA software calculates a score for each node that determines where resources will run. A `ping` resource further influences the node on which a given resource will run. It may be used in any HA service.

Testing Before Applying HA

Before applying an HA environment, ensure that the IP addresses you want to include are valid and accessible with the `ping`(1) command.

ping Template

Template location:

`/usr/share/doc/sgi-ha/templates/ping`

Use the following cloned resource:

```
primitive PING ocf:pacemaker:ping \
   op monitor interval="0" timeout="60s" \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="60s" on-fail="restart" \
   params name="PINGSCORE" multiplier="NNNN" debug="true" \
   host_list="IP-PING-LIST"
clone PING-CLONE PING
```

The above creates an attribute called *PINGSCORE* on both nodes (because the *PING* resource is cloned). The value of the *PINGSCORE* attribute will be calculated as the number of IP addresses (from *IP-PING-LIST*) successfully pinged from that node times the *NNNN* multiplier.

To implement a connectivity influence constraint on *RESOURCE* in which the node that can successfully contact the most IP addresses in *IP-PING-LIST* will be chosen (assuming that all other elements of the cumulative node scores for *RESOURCE* are equal), use the following:

```
location BEST-LOCATION RESOURCE \
         rule PINGSCORE: defined PINGSCORE
```

This adds the value of *PINGSCORE* on each node to that node's overall score for *RESOURCE*, if the *PINGSCORE* attribute is defined (that is, the *PING* resource is running on the node).

To implement a connectivity influence constraint on *RESOURCE* that requires a node to meet a minimum-connectivity requirement in order to be eligible to run the *RESOURCE*, use the following constraint:

```
location THRESHOLD-LOCATION RESOURCE \
         rule -inf: not_defined PINGSCORE or PINGSCORE lt THRESHOLD-VALUE
```

This sets the node score for *RESOURCE* to `-INFINITY` if either the *PINGSCORE* attribute is not defined for a node (that is, the *PING* resource is not running on the node) or the value of *PINGSCORE* is less than the defined *THRESHOLD-VALUE* on that node.

Variable	**Description**
PING	Name of this resource instance, such as `PING`.
IP-PING-LIST	Comma-separated list of IP addresses on which to execute `ping`, such as: `28.162.246.63,128.162.246.61,128.162.232.79`
PINGSCORE	Name of the attribute (such as `PINGSCORE`) that will consist of a value computed by the `ping` resource agent.

Note: Because the *PING* resource is cloned, all nodes in the cluster will use the same attribute. On each node, the value of the attribute is calculated by the `ping` resource agent for the *PINGSCORE* parameter will be computed as the number of IP addresses (from the *IP-PING-LIST*) that were successfully accessed via the `ping` command multiplied by the *NNNN* value.

NNNN	The arbitrary value, such as `1000`, by which the number of the IP addresses from *IP-PING-LIST* which were successfully reached via `ping` will be multiplied.
PING-CLONE	Name of the `clone` instance
BEST-LOCATION	Name of the `location` constraint
THRESHOLD-LOCATION	Name of the `location` constraint
RESOURCE	Resource to be influenced, such as `DMF-GROUP` (see "`dmf-group` Template" on page 268)
THRESHOLD-VALUE	The minimum number of IP addresses that *RESOURCE* must be able to `ping`, multiplied by the *NNNN* multiplier value

For example, suppose that you want to ensure that a node running the `DMF-GROUP` is always able to successfully `ping` at least two of the IP addresses in *IP-PING-LIST*; using an *NNNN* multiplier value of `1000`, the *THRESHOLD-VALUE* would be `2000`. You could use the following:

```
primitive PING ocf:pacemaker:ping \
   op monitor interval="0" timeout="60s" \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="60s" on-fail="restart" \
   params name="PINGSCORE" multiplier="1000" debug="true" \
   host_list="128.162.246.63,128.162.246.61,128.162.232.79"
clone PING-CLONE PING
location THRESHOLD-LOCATION DMF-GROUP \
   rule -inf: not_defined PINGSCORE or PINGSCORE lt 2000
```

smb

This section discusses the following:

- "smb Purpose" on page 307
- "smb Requirements" on page 307
- "HA Control of the Service" on page 307
- "Testing Before Applying HA" on page 307
- "smb Template" on page 308

smb Purpose

The smb resource controls the smb service for Samba. It may be used in a DMF HA service. See Chapter 6, "Create the DMF HA Service" on page 75.

Figure 6-7 on page 147 depicts the order in which this resource falls in the configuration process.

smb Requirements

See "Samba Requirements *(Optional)*" on page 84.

HA Control of the Service

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

```
ha# chkconfig smb off
```

The HA software will control this service.

Testing Before Applying HA

See "Configure and Test Samba Before Applying an HA Environment" on page 91.

smb Template

Template location:

```
/usr/share/doc/sgi-ha/templates/smb
```

Use the following:

```
primitive SMB lsb:smb \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="60s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="60s" on-fail="fence" \
   meta resource-stickiness="1" migration-threshold="1"
```

Variable	Description
SMB	Name of this resource instance, such as SMB.

For example:

```
primitive SMB lsb:smb \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="60s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="60s" on-fail="fence" \
   meta resource-stickiness="1" migration-threshold="1"
```

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

The `start` operation does the following:

- Starts the `smbd` service by calling the following:

  ```
  service smb start
  ```

- Fails if the `smbd` service does not start

The `stop` operation does the following:

- Stops the `smbd` service by calling the following:

  ```
  service smb stop
  ```

- Fails if the `smbd` service does not stop

`tmf`

This section discusses the following:

- "`tmf` Purpose" on page 309
- "`tmf` Requirements" on page 309
- "Testing Before Applying HA" on page 309
- "HA Control of the Service" on page 309
- "`tmf` Template" on page 310

`tmf` Purpose

The `tmf` resource controls the start/stop of TMF. It may be used in a DMF HA service. See Chapter 6, "Create the DMF HA Service" on page 75.

Figure 6-4 on page 111 depicts the order in which this resource falls in the configuration process.

`tmf` Requirements

See "TMF Requirements" on page 79.

Testing Before Applying HA

See "Configure and Test TMF Before Applying an HA Environment" on page 87.

HA Control of the Service

To use the `tmf` resource, you can optionally stop the `tmf` service on each node before applying an HA environment:

```
node1# chkconfig tmf off
node1# service tmf stop

node2# chkconfig tmf off
node2# service tmf stop
```

The HA software will control this service.

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

Note: There are other associated services in a DMF environment that you must stop. See "Stop Services Related to DMF Before Applying an HA Environment" on page 95.

`tmf` Template

Template location:

`/usr/share/doc/sgi-ha/templates/tmf`

Use the following:

```
primitive TMF ocf:sgi:tmf \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="600s" on-fail="fence" \
   params devgrpnames="DEVGRPNAME-LIST" mindevsup="MINDEVSUP-LIST" \
          devtimeout="DEVTIMEOUT-LIST" loader_names="LOADERNAME-LIST" \
          loader_hosts="HOST-LIST" loader_users="USER-LIST" \
          loader_passwords="PASSWORD-LIST" admin_emails="EMAIL-ADDRESS-LIST" \
   meta resource-stickiness="1" migration-threshold="1"
```

Variable	**Description**
TMF	Name of this resource instance, such as `TMF`.
DEVGRPNAME-LIST	Comma-separated list of TMF device groups defined in the `tmf.config` file that are to be managed by HA software, such as: `ibm3592,t10ka`
MINDEVSUP-LIST	Comma-separated list of the minimum 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-LIST

Comma-separated list of device 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
```

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 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 value of 90+18+10=118.

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 value in *DEVTIMEOUT-LIST*. For example, 2*118=236. You should usually use the same `timeout` value for `start` and `stop`.

LOADERNAME-LIST	Comma-separated list of loader names configured in *DEVGRPNAME-LIST* `tmf.config` that correspond to the device groups listed in *DEVGRPNAME-LIST*, such as: `ibm3494,l700a`
HOST-LIST	Comma-separated list of hosts through which the corresponding loaders listed in *LOADERNAME-LIST* are controlled, such as: `ibm3494cps,stkacsls`
USER-LIST	Comma-separated list of user names that are used to log in to the corresponding hosts listed in *HOST-LIST*, such as: `root,acssa`
PASSWORD-LIST	Comma-separated list of passwords corresponding to the user names listed in *USER-LIST*, such as: `passwd1,passwd2`
EMAIL-ADDRESS-LIST	(Optional) Comma-separated list of administrator email addresses corresponding to the device groups listed in *DEVGRPNAME-LIST*, 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.

For example:

```
primitive TMF ocf:sgi:tmf \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op monitor interval="0" timeout="60s" \
   op start interval="0" timeout="600s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="600s" on-fail="fence" \
   params devgrpnames="ibm3592,t10ka" mindevsup="1,0" devtimeout="120,240" \
          loader_names="ibm3494,l700a" loader_hosts="ibm3494cps,stkacsls" \
          loader_users="root,acssa" loader_passwords="passwd1,passwd2" \
          admin_emails="root,admin1" \
   meta resource-stickiness="1" migration-threshold="1"
```

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

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

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

`winbind`

This section discusses the following:

- "`winbind` Purpose" on page 314
- "Testing Before Applying HA" on page 314
- "HA Control of the Service" on page 307
- "`winbind` Template" on page 314

`winbind` Purpose

If your Samba implementation uses an authentication type that requires the `winbind` daemon, you can use a `winbind` resource to control it. It may be used in a DMF HA service. See Chapter 6, "Create the DMF HA Service" on page 75.

Figure 6-7 on page 147 depicts the order in which this resource falls in the configuration process.

Testing Before Applying HA

See "Configure and Test the Winbind Service Before Applying an HA Environment" on page 91.

HA Control of the Service

If `winbind` is used for authentication, disable the `winbind` service from being started automatically at boot time on all HA nodes:

```
ha# chkconfig winbind off
```

The HA software will control this service.

`winbind` Template

Template location:

```
/usr/share/doc/sgi-ha/templates/winbind
```

Use the following:

```
primitive WINBIND lsb:winbind \
   op monitor interval="0" timeout="60s" \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="60s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="60s" on-fail="fence" \
   meta resource-stickiness="1" migration-threshold="1"
```

Variable	**Description**
WINBIND	Name of this resource instance, such as `WINBIND`

For example:

```
primitive WINBIND lsb:winbind \
   op monitor interval="0" timeout="60s" \
   op monitor interval="120s" timeout="60s" on-fail="restart" \
   op start interval="0" timeout="60s" on-fail="restart" requires="fencing" \
   op stop interval="0" timeout="60s" on-fail="fence" \
   meta resource-stickiness="1" migration-threshold="1"
```

The first `monitor` operation probes to see if the resource is already running and the second periodically verifies that it continues to run.

The `start` operation does the following:

- Starts the `winbind` service by calling the following:

  ```
  service smb winbind
  ```

- Fails if the `winbind` service does not start

The `stop` operation does the following:

- Stops the `smbd` service by calling the following:

  ```
  service smb winbind
  ```

- Fails if the `winbind` service does not stop

Glossary

active/active cluster

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 cluster

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.

control services

The set of underlying services that control the cluster, which vary by operating system:

- RHEL:
  - CMAN
  - Corosync
  - Pacemaker
- SLES:
  - OpenAIS
  - Corosync
  - Pacemaker

COPAN MAID

Power-efficient long-term data storage based on an enterprise massive array of idle disks (MAID) platform.

Corosync

The infrastructure 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 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*.

fail policy

A parameter defined in the CIB that determines what happens when a resource fails.

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, the state in which resources fail over from one node to another without disrupting services for clients.

HA 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. See also *non-HA service*.

High Availability Extension

SUSE Linux Enterprise product for HA.

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.

monitor operation

See *probe monitor operation* and *standard monitor operation*.

mover1

In the examples in this guide, the initial parallel data-mover node (which will later become a node in the HA cluster) which will be the initial owner node of shelves 0 and 1. See also *mover2*.

mover2

In the examples in this guide, the alternate parallel data-mover node in the HA cluster (which will later become a node in the HA cluster) which will be the initial owner node of shelves 2 and 3. See also *mover1*.

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 *node2*.

node2

In the examples in this guide, the alternate host in the HA cluster other than the first node (node1). See also *node1*.

non-HA service

A service such as DMF before an HA environment is applied. See also *HA service*.

NSM

Network Status Monitor.

OCF

Open Cluster Framework.

OpenVault

A 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 failover.

physvol

XVM physical volume.

primitive

Used to define a resource in the CIB.

probe `monitor` operation

A `monitor` operation that checks to see if the resource is already running.

resource

An application that is managed by HA software.

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 `monitor` operation

A `monitor` operation that periodically verifies that the resource continues to run.

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 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) and CXFS filesystems.

Index