High Availability Extension and SGI® InfiniteStorage

007–5617–003

New Features in this Guide

This revision contains the following:

- A description of the probe monitor operation specification for each resource
- Changes to the suggested values for the **Timeout** and **Interval** fields for many resource operations
- Removal of the **Start Delay** optional field for monitor operations
- Other clarifications and corrections

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

Contents

Figures

Tables

About This Guide

This publication provides information about creating resources for the high-availability (HA) SGI resource agents that SGI provides for use with the SUSE® Linux® Enterprise High Availability Extension (HAE) product.

Prerequisites

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

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

Related SGI® Publications

The following SGI publications contain additional information:

- *CXFS 6 Administration Guide for SGI InfiniteStorage*
- *CXFS 6 Client-Only Guide for SGI InfiniteStorage*
- *DMF 5 Administrator's Guide for SGI InfiniteStorage*
- *DMF 5 Filesystem Audit Guide for SGI InfiniteStorage*
- *OpenVault Operator's and Administrator's Guide*
- *SGI L1 and L2 Controller Software User's Guide*
- *SGI InfiniteStorage Software Platform* (ISSP) release note (`README.txt`)
- *TMF 5 Administrator's Guide for SGI InfiniteStorage*
- *XVM Volume Manager Administrator's Guide*
- The hardware guide for your SGI server

Obtaining SGI Publications

You can obtain SGI documentation as follows:

- See the SGI Technical Publications Library at http://docs.sgi.com. Various formats are available. This library contains the most recent and most comprehensive set of online books, man pages, and other information.
- You can view man pages by typing `man` *title* at a command line.
- The `/docs` directory on the ISSP DVD or in the Supportfolio™ download directory contains the following:
  - The ISSP release note: `/docs/README.txt`
  - Other release notes: `/docs/README_`*NAME*`.txt`
  - A complete list of the packages and their location on the media: `/docs/RPMS.txt`
  - The packages and their respective licenses: `/docs/PACKAGE_LICENSES.txt`
- The release notes and manuals are provided in the `noarch/sgi-isspdocs` RPM and will be installed on the system into the following location:

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

Conventions

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

The following conventions are used throughout this document:

Convention	Meaning
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
variable	Italic typeface denotes variable entries and words or concepts being defined.
`user input`	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. (Output is shown in nonbold, fixed-space font.)
[ ]	Brackets enclose optional portions of a command or directive line.
...	Ellipses indicate that a preceding element can be repeated.
`manpage`(*x*)	Man page section identifiers appear in parentheses after man page names.
GUI	This font denotes the names of graphical user interface (GUI) elements such as windows, screens, dialog boxes, menus, toolbars, icons, buttons, boxes, fields, and lists.

Reader Comments

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

You can contact SGI in any of the following ways:

- Send e-mail to the following address:

  techpubs@sgi.com

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

- Send mail to the following address:

  SGI
  Technical Publications
  46600 Landing Parkway
  Fremont, CA 94538

SGI values your comments and will respond to them promptly.

Introduction

This chapter discusses the following:

High Availability Extension

The SUSE® Linux® Enterprise High Availability Extension (HAE) product provides the infrastructure to fail over *highly available (HA) resources* that survive a single point of failure. A *resource* is a service, associated with an IP address, that is managed by HAE. A *resource group* is a set of resources that must be managed and failed over from one node to another as a set. HAE starts, monitors, and stops resources. A *resource agent* is the set of software that allows a service to be highly available without modifying the application itself.

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

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

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

SGI Resource Agents

Table 1-1 lists the resource agents that SGI provides in the **SGI ISSP High Availability** YaST pattern.

Table 1-1 SGI Resource Agents

Resource Agent	Description
`cxfs`	CXFS™ clustered filesystems whose metadata server location must follow the location of another service, such as DMF or NFS. See "Creating the CXFS Primitive" on page 61.
`cxfs-client`	CXFS clustered filesystems (mounted on a CXFS client-only node) that are required to support another service, such as those to be NFS-served from a CXFS client-only node. See "Creating the CXFS Client Primitive" on page 44.
`cxfs-client-nfsserver`	NFS server on a CXFS client-only node. See "Creating the CXFS Client NFS Server Primitive" on page 47.
`cxfs-client-smnotify`	Network Status Monitor (NSM) lock reclaim notification on a CXFS client-only node. See "Creating the CXFS Client NSM Notification Primitive" on page 53.
`dmf`	DMF server. See "Creating the DMF Primitive" on page 97.
`dmfman`	DMF Manager tool. See "Creating the DMF Manager Primitive" on page 104.
`dmfsoap`	DMF client Simple Object Access Protocol (SOAP) service. See "DMF Client SOAP Standard Service" on page 40.
`l2network`	STONITH node-level fencing for systems using L1/L2 controllers, such as SGI ia64 systems. See "Creating the L2 STONITH Primitive " on page 112.
`lxvm`	Local XVM volume manager. See "Creating the Local XVM Primitive" on page 64.
`openvault`	OpenVault mounting service for DMF. See "Creating the OpenVault Primitive" on page 84.

Resource Agent	Description
`sgi-ipmi`	STONITH node-level fencing for systems with a baseboard management controller (BMC) using intelligent platform management interface (IPMI), such as SGI x86-64. See "Creating the IPMI STONITH Primitive " on page 115.
`tmf`	Tape Management Facility (TMF) mounting service for DMF. See "Creating the TMF Primitive" on page 88.

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

- CXFS NFS edge-serving from CXFS client-only nodes in a two-node active/active HAE cluster. See "CXFS™ NFS Edge-Serving Example" on page 4.
- DMF in a two–node active/passive HAE cluster. See "DMF Failover Example" on page 6.

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

Note: The attributes and the various value recommendations listed in Chapter 6, "CXFS NFS Edge-Serving HA Resource Examples" and Chapter 7, "DMF HA Cluster Resource Examples" are in support of the examples used in this guide. If you are using the resources in a different manner, you must evaluate whether these recommendations and use of meta attributes apply to your intended site-specific purpose.

Failover Example Scenarios

This section discusses the following:

- "CXFS™ NFS Edge-Serving Example" on page 4
- "DMF Failover Example" on page 6

CXFS™ NFS Edge-Serving Example

As an example, Figure 1-1 and Figure 1-2 describe the process of failing over a CXFS NFS edge-serving configuration using active/active mode on a two-node HAE cluster.

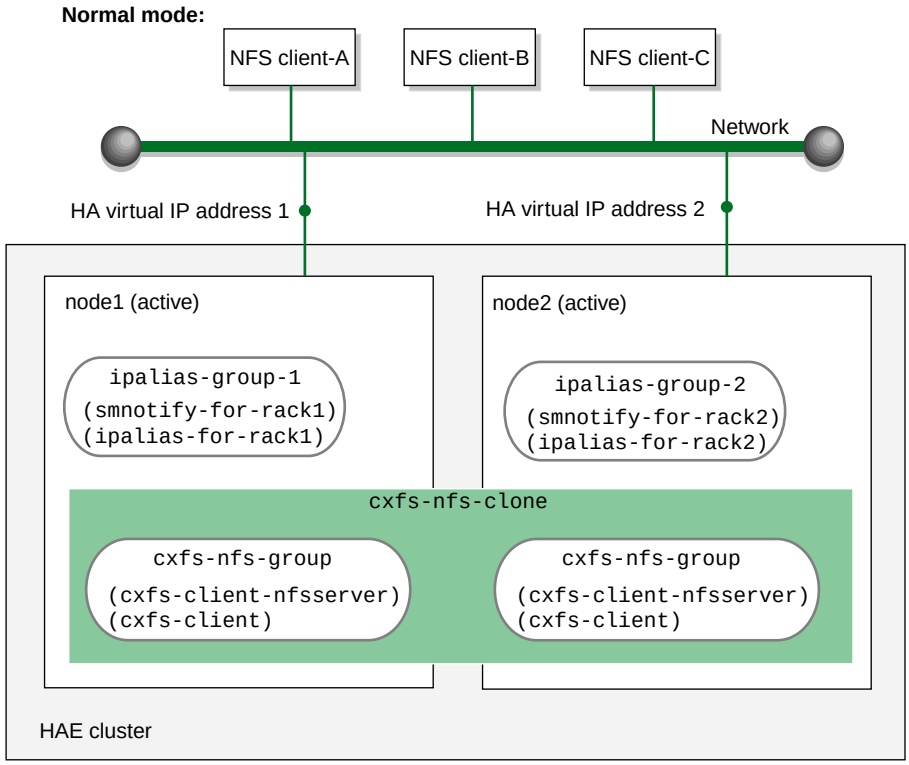

Figure 1-1 CXFS NFS Edge-Serving HA Example — Normal Mode

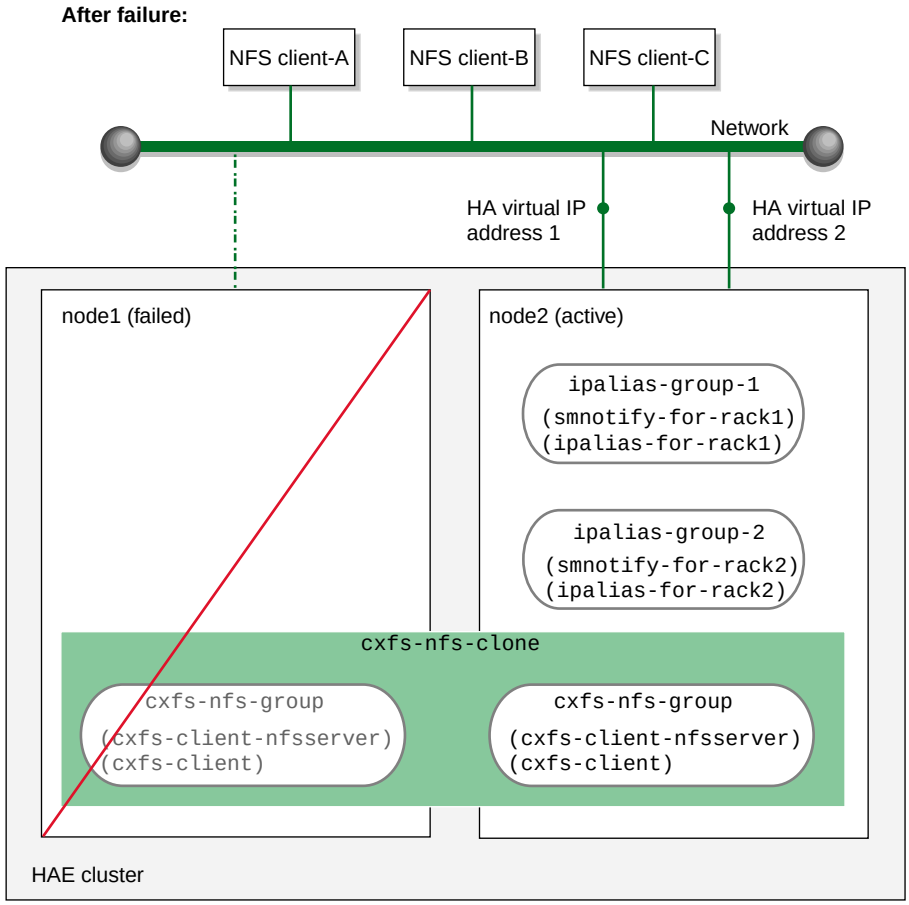

Figure 1-2 CXFS NFS Edge-Serving HA Example — After Failover

DMF Failover Example

As an example, Figure 1-3 and Figure 1-4 describe the process of failing over a DMF configuration using active/passive mode on a two-node HAE cluster.

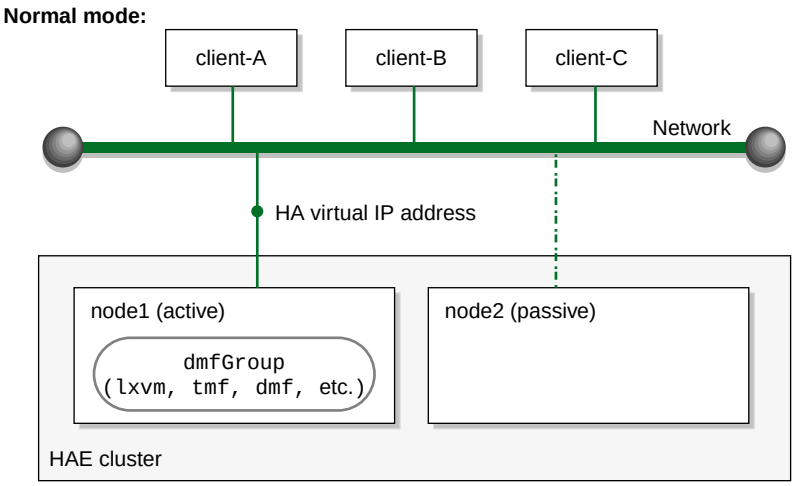

Figure 1-3 DMF HA Example — Normal Mode

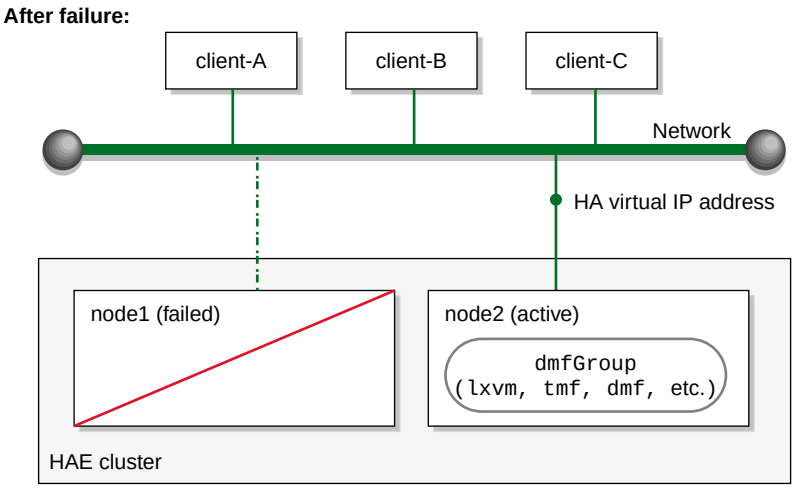

Figure 1-4 DMF HA Example — After Failover

Software Packages

Table 1-2 lists the RPMs contained in the **SGI ISSP High Availability** YaST pattern.

Table 1-2 ISSP HA RPMs

RPM	Contents

`sgi-ha-ocf-plugins`	Open Cluster Framework (OCF) resource agents:
	`cxfs`
	`cxfs-client`
	`cxfs-client-nfsserver`
	`cxfs-client-smnotify`
	`dmf`
	`dmfman`
	`dmfsoap`
	`lxvm`
	`openvault`
	`tmf`

`sgi-ha-stonith-plugins`	STONITH resource agents for node-level fencing:
	`l2network`
	`sgi-ipmi`

For information about software installation, see the *SGI InfiniteStorage Software Platform* (ISSP) release note and Chapter 4, "Outline of the Configuration Procedure" on page 27.

Configuration Tools

The procedures in this guide use the following tools as documented in the Novell *High Availability Guide*:

- YaST installation and configuration tool
- Linux HA Management Client graphical user interface (GUI) accessed with the `crm_gui` command
- Linux HA Management Client command-line administration tools (such as `cibadmin`, `crm_attribute`, and `crm_resource`)

Best Practices

The following are best practices when using SGI resource agents with High Availability Extension (HAE):

Preliminary Best Practices

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

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

  To do these things, review the overall procedure example in Chapter 4, "Outline of the Configuration Procedure" and then follow the detailed example steps in Chapter 5, "Standard Services" through Chapter 8, "STONITH Resource Examples".
- To use the HAE GUI (`crm_gui`), you can log in with any user ID that has access to the `haclient` group, but you must know the password for the user ID. By default, the GUI uses the `hacluster` user ID.

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

  – Add the `root` user to the `haclient` group

  – Set the password for the `hacluster` user before you start `crm_gui`.

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

HAE Configuration Best Practices

SGI recommends that you use the HAE GUI (`crm_gui`) to configure the HAE cluster. Note the following tips about the configuration process and using the configuration information in this guide:

- Use the sandbox feature to test configuration changes. See the `crm_shadow`(8) man page for more information.
- For a DMF HA cluster, place all resource primitives within one resource group. The resource group mechanism incorporates implied colocation constraints as well as resource order constraints. The resource group concept lets you control the resources as a single entity, which greatly simplifies administration.
- To keep things simple, avoid creating explicit location, colocation, or order constraints on individual resource primitives except as directed in this guide. In most cases, you should only place constraints on the resource group as a whole.

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

- Use unique IDs for all resource clones, groups, and primitives.
- Do not use spaces in resource IDs because this may cause HAE or other supporting software to behave in a confusing manner.
- When you enter a time value, the default is usually in milliseconds; to use seconds, such as 30 seconds, you must usually include the character "`s`" as in `30s`.

- Always use STONITH node-level fencing to protect data integrity in case of failure.
- You may want to examine the use of the `resource_stickiness` and `migration_threshold` attributes of resources to control how often and to what node failover will occur. The examples in this book result in the resource group failing over to the alternate node on the first failure of any of the resource primitive in the resource group. In this default setting, there is no automatic failback to the first node. For more information about score calculation, see the Novell documentation and the following website:

  http://www.clusterlabs.org/doc/en-US/Pacemaker/1.0/html/Pacemaker_Explained/index.html
- Values shown in *italic font* in this guide are site-specific and are therefore changeable; suggested starting values that you must enter are shown in `literalfont`. Most instance attributes and timeouts are site-specific. When possible, some guidance is given for determining appropriate values.
- Fields that are unnecessary or for which the GUI provides appropriate defaults are not addressed in this guide.
- Some **Operations** fields are accessed under the **Optional** tab. Those that are required for the SGI implementation of HAE are listed in this guide using the format **Optional > *Field Name***.

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

- Click **Apply** only after you have entered all of the required operations.
- In many cases, there are pull-down lists that contain possible values (shown in **boldface** in this guide). In other cases, you must enter in text (shown in `literalfont`).
- In general, you must use the values shown in this guide for meta attributes and for those values available from a pull-down list.
- You may need to resize the GUI to view some fields and tabs. In some cases, the cursor must be positioned on the left-edge of a tab in order to select it.

- **ID** is the unique identification of the clone, resource group, or resource primitive, such as `cxfs`. This can be any name you like as long as it does not include spaces. For the **monitor**, **start**, and **stop** operations, a unique ID will be generated for you based on the primitive name and interval, such as `cxfs_op_monitor_30s`.
- **Class**, **Provider**, and **Type** must use the exact values shown in this guide.
- **monitor** is the **name** value of the operation that determines if the resource is operating correctly. The resource will be monitored at an interval of the specified time (the interval begins at the end of the last monitor completion). Each **monitor** operation will timeout after the specified number of seconds. If the **monitor** operation fails, it will attempt to restart the resource.

  Note: To prevent a resource from being monitored and possibly triggering failovers, set **interval** to `0`. This may be useful for those resources that are not critical (such as DMF Manager) but should still move with the rest of the resource group.

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

  Note: *Fencing* in HAE terminology is not the same as *fencing* in CXFS terminology.

- **stop** is the **name** value of the operation that terminates or gives up control of the resource. It will timeout after the specified time. If the **stop** operation fails, it will attempt to fence the node on which the failure occurred. The **stop** fail policy must be set to **fence** and a STONITH facility must be configured according to the requirements for your site (see Chapter 8, "STONITH Resource Examples" on page 111.)
- **resource-stickiness** specifies a weight 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 goes into standby mode) or if there is a **start**, **monitor**, or **stop** failure for this resource or another resource in the same resource group.

- **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.
- Use the following command to verify changes you make to the CIB, with each resource that you define:

  ```
  # crm_verify -LV
  ```

Administrative Best Practices

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

The following are best practices for administering the HAE cluster:

- After you have successfully completed the initial configuration, make a backup copy of the working cluster information
  base (CIB) using the `cibadmin`(8) command so that you can return to the previous CIB if necessary.

  Before making changes to an existing HAE configuration, ensure that you have a good backup copy of the CIB so that you can return to it if necessary. After you establish that your changed configuration is good, make a new backup of the 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.

  For more information, see the Novell *High Availability Guide* and the `cibadmin`(8) man page.
- Periodically watch the output of the following commands for problems:

  ```
  # crm_mon -r1
  # crm_verify -LV
  ```

  Refer to the `/var/log/messages` system log periodically (to ensure that you are aware of operations automatically initiated by HAE) and if you notice errors. See "Reviewing the Log File" on page 120.

- You may add multiple `-V` options to many of the HAE commands in order to increase verbosity. For example:

  # **`crm_verify -LVVV`**

- After a failure, clear the resource primitive failcount values for a node immediately after resolving the cause of the failure (or reboot the system). See "Clearing the Resource Primitive Failcount" on page 120.

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

  # **`crm_resource -r`** ***resourceGROUP*** **`-M -H`** ***node***

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

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

- Remove resource group location constraints when they are no longer needed, such as after the resource group has successfully moved to the new node. Do the following:

  # **`crm_resource -r`** ***resourceGROUP*** **`-U`**

- Set the HAE `totem token` (core membership timeout) value to one that is significantly higher than the CXFS heartbeat timeout. For example:

  – For the default CXFS heartbeat timeout of 5s, set the HAE `totem token` value to at least 15s

  – For a CXFS heartbeat timeout of 60s, use an HAE `totem token` value of 90s

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

- When upgrading the software, follow the procedure in "Performing a Rolling Upgrade in an HAE Environment" on page 123.

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

Requirements

This chapter discusses the following requirements for a High Availability Extension (HAE) cluster using SGI resource agents:

HAE Support Requirements

HAE may in some cases require the purchase of additional support from Novell.

Licensing Requirements

All nodes in an HAE 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

Due to STONITH reset requirements, all nodes that might run SGI resource agents in an HAE cluster must be of the same system type, supporting just one of the following:

- An L2 with Ethernet connectivity
- A BMC supporting the IPMI protocol and administrative privileges

---

Note: If you form an HAE 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 HAE cluster. CXFS does not support these members as server-capable administration nodes in the CXFS cluster.

DMF supports only one instance running on a given node in an HAE cluster at any given time, thus active/active mode is not a possible configuration. If the cluster also runs CXFS, the DMF server nodes in the cluster must also be CXFS server-capable administration nodes. For additional requirements when using the DMF Parallel Data Mover Option, see *DMF 5 Administrator's Guide for SGI InfiniteStorage*.

System Reset Requirements

You must use STONITH node-level fencing to protect data integrity in case of failure. See Chapter 8, "STONITH Resource Examples" on page 111.

Note: `sgi-ipmi` requires the use of a BMC user account with administrative privileges. For more information, see the `ipmitool`(1) man page and the user guide or quick start guide for your system.

CXFS NFS Edge-Serving Requirements

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

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

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

- Due to the way that NLM grace notification is implemented, all of the server-capable administration nodes in the CXFS cluster must run the same version of CXFS in order to use CXFS relocation. This means that if you want to do a CXFS rolling upgrade of the metadata servers while running HA CXFS NFS edge-serving, you must use CXFS recovery and not CXFS relocation.
- During HA operation, the CXFS client services (`cxfs_client`) and NFS services (`nfsserver`) must be turned off on all CXFS NFS edge-server HA cluster systems:

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

  The HA software will start these services.
- Using NFS client services on a CXFS NFS edge server does not support monitored locking via `statd`.
- There must be a file (the `statefile` instance attribute for the CXFS client NFS server) located on shared storage on which to keep kernel state information:

  - In a non-HA cluster, this would be the `/var/lib/nfs/state` file
  - All systems in a single CXFS NFS edge-server HA cluster must share this file
  - If there are multiple CXFS NFS edge-server HA clusters within one CXFS cluster, all of the cluster systems must share this file
- There must be a directory (the `statedir` instance attribute for the CXFS client NFS server) located on shared storage used to store NFS lock state:
  - In a non-HA cluster, this would be the `/var/lib/nfs/` directory
  - All systems in a single CXFS NFS edge-server HA cluster must share this directory
  - If there are multiple CXFS NFS edge-server HA clusters within one CXFS cluster, each must have a separate state directory

Also see:

- "Preliminary Best Practices" on page 9
- "Instance Attributes for CXFS Client NFS Server" on page 47

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 18
- "CXFS Relocation Support" on page 19
- "Applications that Depend Upon CXFS Filesystems" on page 19
- "CXFS and System Reset" on page 20
- "CXFS Start/Stop Issues" on page 20

CXFS Server-Capable Administration Nodes

The HAE 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 HAE resource be provided by the same node; see "DMF Requirements" on page 23. 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 provided from a node that is not also the CXFS metadata server.)

Unless a resource must run on the CXFS metadata server, you should configure HAE so that it does not relocate the CXFS metadata server when it fails-over a resource.

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

CXFS 6 Administration Guide for SGI InfiniteStorage
CXFS 6 Client-Only Guide for SGI InfiniteStorage

CXFS Relocation Support

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

Applications that Depend Upon CXFS Filesystems

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

- The application will not run on a server-capable administration node that is not 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 system reset in order to prevent conflicts with CXFS I/O fencing methods. You must specify the following in the node definition for the server-capable administration nodes:

- Fail policy that includes `Reset` or `FenceReset`
- Reset method of `reset`

For more information, see Chapter 8, "STONITH Resource Examples" on page 111.

CXFS Start/Stop Issues

You should start CXFS cluster services and CXFS services before starting HAE. 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 HAE fail policy for any resource that could prevent relocation by holding files open must be `fence` and you must configure a STONITH facility according to the requirements for your site. See Chapter 8, "STONITH Resource Examples" on page 111.

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

Local XVM Requirements

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

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

- HAE 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 in HAE, resulting in node reset.

Filesystem Requirements

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

Virtual IP Address Requirements

The virtual IP served by the `IPaddr2` resource agent must use the same Ethernet device number (*X* value in `eth`*X*) on each host that may provide the virtual IP resource.

OpenVault™ Requirements

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

- If upgrading to an entirely new root filesystem, as would be required if upgrading from a SLES 10 system, you should create a copy of the OpenVault configuration directory (`/var/opt/openvault`) from the old root before upgrading the OS. You can then reinstall it on the new root so that you do not need to entirely reconfigure OpenVault. See the section about taking appropriate steps when upgrading DMF in the *DMF 5 Administrator's Guide for SGI InfiniteStorage*.
- Provide a directory for OpenVault's use within an HA filesystem in the DMF resource group. This is known as the *serverdir directory* (as specified in "OpenVault Resource" on page 78). The directory will hold OpenVault's database and logs. The directory can be either of the following:
  - Within the root of an HAE-managed filesystem dedicated for OpenVault use
  - Within another HAE-managed filesystem, such as the filesystem specified by the DMF configuration file `HOME_DIR` parameter

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

- Create a virtual hostname for use by OpenVault and either add it to your local DNS server or add it 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. The address associated with that virtual hostname must be a virtual address managed by a community `IPaddr2` resource within the same resource group as the `openvault` resource. You may also use the `IPaddr2` virtual address for other purposes, such as NFS. See also:
  - "Virtual IP Address Requirements" on page 21
  - "Virtual IP Address Resource" on page 75
- 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.
- The DMF application instances in OpenVault must be configured to use a wildcard ("`*`") for the hostname and instance name. For more information, see the procedure about configuring DMF to use OpenVault in the *DMF 5 Administrator's Guide for SGI InfiniteStorage*.
- During HA operation, the OpenVault service (`openvault`) must be turned off on all HA nodes :

  ```
  # chkconfig openvault off
  ```

  The HA software will start this service.

TMF Requirements

All tape devices should be configured as `DOWN` in the `tmf.config` file on all nodes. The loaders may be configured as `UP` and the `tmf` service may restart automatically (`chkconfig tmf on`) for all nodes. (However, the resource agent will start `tmf` and configure the loader up if necessary.)

DMF Requirements

Using DMF with HAE requires the following:

- The HAE cluster must contain all nodes that could be DMF servers.
- Each DMF server must run the required product and HA software.
- All DMF server nodes must have connectivity to all of the XFS® filesystems that DMF either depends upon or manages. All 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 filesystems must be managed by a community `Filesystem` resource in that resource group.

  The DMF filesystems to be managed are:

  – The DMF-managed user filesystems

  – 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 DCMs and disk MSPs using local disk storage

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

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

  For more information about DMF filesystems, see the *DMF 5 Administrator's Guide for SGI InfiniteStorage*.

- All DMF server nodes in the HAE cluster must have connectivity to the same set of tape libraries and drives. If one node only has access to a subset of the drives, and the DMF server is failed over to that node, DMF would then not be able to access data on tapes left mounted in inaccessible drives.

- 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 tape resources it uses are available (which also implies that the `dmf` resource must be stopped before those resources are stopped).
- Create a virtual hostname for use by DMF and either add it to your local DNS server or add it to the `/etc/hosts` file on all hosts in the cluster that could be a DMF server and on any DMF parallel data mover nodes. The address associated with that virtual hostname must be a virtual address managed by a community `IPaddr2` resource within the same resource group as the `dmf` resource. You may also use the `IPaddr2` virtual address for other purposes, such as NFS. See also:
  - "Virtual IP Address Requirements" on page 21
  - "Virtual IP Address Resource" on page 75
- If using the DMF Parallel Data Mover Option, set the `HA_VIRTUAL_HOSTNAME` parameter for potential DMF server nodes to the same virtual hostname used for `SERVER_NAME` in the `base` object of the DMF configuration file. See "DMF Resource" on page 94.
- During HA operation, DMF service (`dmf`) must be turned off on all HA nodes:

  # **`chkconfig dmf off`**

  The HA software will start this service.

Also see:

- "DMF Manager Requirements" on page 25
- "DMF Client SOAP Service Requirements" on page 25

NFS Requirements

During HA operation, the NFS service (`nfsserver`) must be turned off on all HA nodes :

`chkconfig nfsserver off`

The HA software will start this service.

DMF Manager Requirements

During HA operation, the DMF Manager service (`dmfman`) must be turned off on all HA nodes :

```
# chkconfig dmfman off
```

The HA software will start this service.

DMF Client SOAP Service Requirements

During HA operation, the DMF client SOAP service (`dmfsoap`) must be turned off on all HA nodes:

```
# chkconfig dmfsoap off
```

The HA software will start this service.

Outline of the Configuration Procedure

This chapter summarizes the recommended steps to configure a High Availability Extension (HAE) cluster for use with SGI InfiniteStorage products by using an example two-node HAE cluster.

Do the following:

1. Understand the requirements for the SGI products you want to include in your HAE cluster. See Chapter 3, "Requirements" on page 15.

2. Ensure that you have installed the required SGI products for your cluster (including the **SGI ISSP High Availability** YaST pattern) according to the installation procedure in the *SGI InfiniteStorage Software Platform* release note.

3. Configure and test each of the standard SGI product services before making them highly available. Do this using one host (which will later become a node in the HAE cluster) on which all of the filesystems will be mounted and on which all tape drives and libraries are accessible, known in this guide as *node1*. See Chapter 5, "Standard Services" on page 35.

   If you already have stable systems configured, you can skip this step and proceed to step 4.

4. Stop the standard services other than those for CXFS (`cxfs` and `cxfs_cluster`) by using the `service` command (execute on both nodes):

   # **`service`** ***`servicename`*** **`stop`**

5. Prevent the standard services other than those for CXFS (`cxfs` and `cxfs_cluster`) from restarting by using the `chkconfig` command (execute on both nodes):

   # **`chkconfig`** ***`servicename`*** **`off`**

6. Install the SUSE HAE software as documented in the Novell *High Availability Guide* provided by the following Novell, Inc., website:

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

7. Follow the instructions in the Novell *High Availability Guide* to initialize the cluster and configure node1. See the information about the setting the password and using the HAE GUI (`crm_gui`) in:

   - "Preliminary Best Practices" on page 9
   - "HAE Configuration Best Practices" on page 10

   During the initialization process, do the following:

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

   b. Set **Bind Network Address** to the network that will support the cluster heartbeat (for example, the CXFS private network, such as `128.162.244.0`), which is different from the IP address to be failed over.

   c. Set **Multicast Address** to a multicast address (for example, `226.94.1.1`) .

   d. Set **Multicast Port** to a multicast port (for example, `5405`).

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

   ---

   e. Explicitly set the node ID or use **Auto Generate Node ID**. (Each node must have a unique node ID number.)

   f. Set security on.

   g. Select **Generate Auth Key File** on node1 only. It will be created in `/etc/corosync/authkey`.

   Note: This process can take several minutes to complete.

   Do not create a new key on node2.

   Change the permission on `/etc/corosync/authkey` to allow read and write permission for the root user only:

   ```
   # chmod 0600 /etc/corosync/authkey
   ```

   h. *(Optional)* Set `openais` to start at boot time. (You can also do this manually later by setting `chkconfig openais on`.)

i. *(Optional but recommended)* Add directives to `/etc/sysctl.conf` for the `kernel.core_pattern` and `kernel.core_uses_pid` variables. For example:

```
kernel.core_pattern = core.%p.%t
kernel.core_uses_pid = 1
```

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:

```
# echo 1 > /proc/sys/kernel/core_uses_pid
# echo "core.%p.%t" > /proc/sys/kernel/core_pattern
```

8. Follow the instructions in the Novell *High Availability Guide* and in step 7 above to create node2 as appropriate.

Note: If you set the node ID explicitly in step 7e above, you must also set the node ID explicitly on node2 so that it is different from the node ID on node1. (Each node must have a unique node ID number.) If you set explicit node IDs, you should not use `csync2` as directed in the Novell *High Availability Guide* because it will result in duplicate node IDs; instead, you must copy the `/etc/corosync/authkey` and `/etc/corosync/corosync.conf` files from node1 to node2 and set the mode for these files to `0600`.

---

9. Configure the `logd` and the HAE `openais` services so that they will start upon reboot (execute on both nodes):

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

10. Start the `logd` and the HAE `openais` services (execute on both nodes):

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

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

```
node1# crm_mon -ri1
```

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

```
node1# crm_attribute -t crm_config -n stonith-enabled -v false
```

Note: You will reenable system reset in step 15 after testing all of the SGI resource primitives in step 14.

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

```
node1# crm_attribute -t crm_config -n no-quorum-policy -v ignore
```

14. Configure and test the required resources for your configuration. Proceed to the next resource primitive only if the current resource is behaving as expected, as defined by the documentation.

Note: Using the instructions in this guide, you must configure resources in the specific order shown.

For example, see the following:

- Chapter 6, "CXFS NFS Edge-Serving HA Resource Examples" on page 41:
  1. Create a clone containing a group and service resources:
     a. "CXFS Client Resource" on page 44
     b. "CXFS Client NFS Server Resource" on page 46
  2. Create two IP alias groups, one for each rack:
     a. "Virtual IP Address Resource" on page 75
     b. "CXFS Client NSM Notification Resource" on page 53
  3. Create constraints for resource order, colocation, and location

Figure 4-1 on page 32 shows a map of the configuration procedure for CXFS NFS edge-serving, referring to resource agent type names such as `cxfs-client` and `IPaddr2`.

- Chapter 7, "DMF HA Cluster Resource Examples" on page 59:

  1. Filesystems. Either:

     – "CXFS Resource" on page 61

     – "Local XVM Resource" on page 64 and one or more "Filesystem Resources" on page 67

  2. "Virtual IP Address Resource" on page 75

  3. A mounting service, either:

     – "OpenVault Resource" on page 78

     – "TMF Resource" on page 88

  4. "DMF Resource" on page 94

  5. "NFS Resource" on page 99

  6. "DMF Manager Resource" on page 104

  7. "DMF Client SOAP Service Resource" on page 106

  Figure 4-2 on page 33 shows an example map of the configuration procedure for DMF, referring to resource agent type names such as `lxvm` and `IPaddr2`.

15. Configure and reenable node-level fencing. See Chapter 8, "STONITH Resource Examples" on page 111.

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

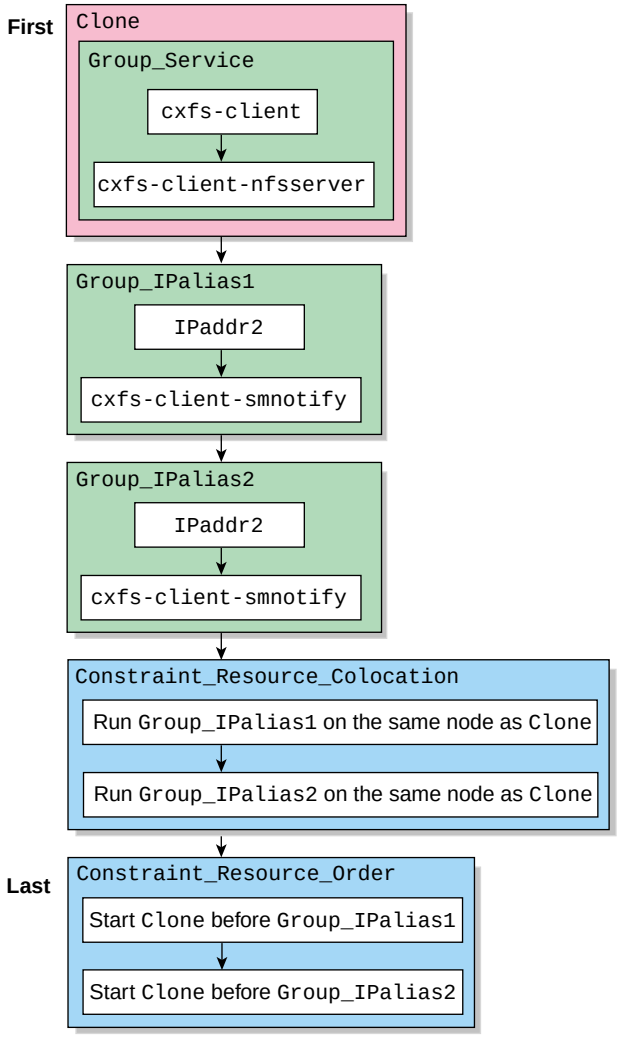

Figure 4-1 CXFS NFS Edge-Server HA Resource Configuration Process Map

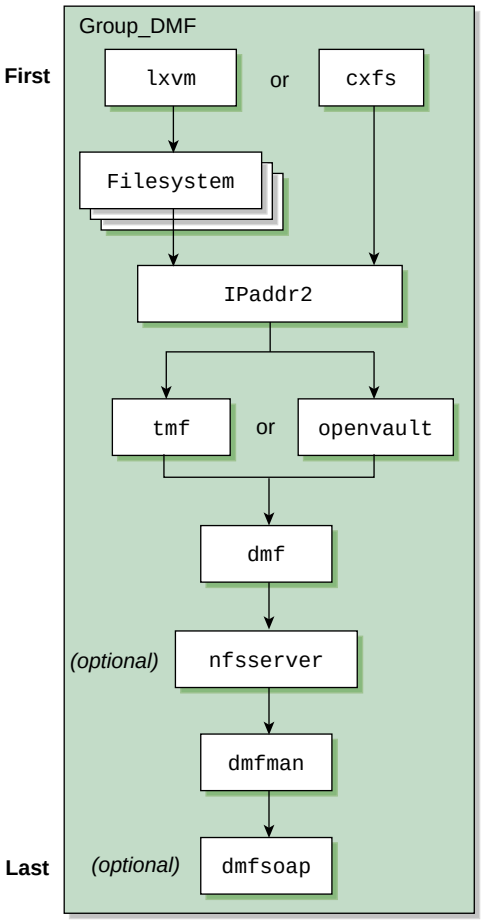

Figure 4-2 DMF HA Resource Configuration Process Map

Standard Services

You should configure and test all standard services before applying high availability. In general, you should do this on one host (known in this guide as *node1*). Node1 will later become a node in the High Availability Extension (HAE) cluster, on which all of the filesystems will be mounted and on which all tape drives and libraries are accessible. If you already have a stable configuration, you can skip the steps in this chapter.

This chapter discusses the following:

CXFS NFS Edge-Serving Standard Service

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.

To test the NFS service, do the following on both nodes:

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

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

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

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

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

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

CXFS Standard Service

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

1. Configure CXFS on node1 (which must be a CXFS server-capable administration node), according to the instructions in the following:
   - "CXFS Requirements" on page 18
   - *CXFS 6 Administration Guide for SGI InfiniteStorage*
   - *CXFS 6 Client-Only Guide for SGI InfiniteStorage*
2. Start CXFS services and CXFS cluster services. For more information, see *CXFS 6 Administration Guide for SGI InfiniteStorage.*

3. Verify that the filesystem in question mounts on all applicable nodes. For example, use the `cxfs_admin` command:

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

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

Local XVM Standard Service

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

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

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

OpenVault Standard Service

Configure OpenVault on node1, according to the instructions in the *OpenVault Operator's and Administrator's Guide*. This means that you will use the **actual** hostname as reported by the `hostname`(1) command when using `ov_admin`. Configure all local libraries and tape drives.

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

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

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

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

TMF Standard Service

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

```
node1# chkconfig tmf on
```

Note: In the `tmf.config` file, drives in drive groups managed by HAE 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 TMF, do the following:

1. Use `tmstat` to verify that all 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
   ```

DMF Standard Service

Configure DMF according to the instructions in the *DMF 5 Administrator's Guide for SGI InfiniteStorage*.

To test DMF, do the following:

1. Migrate a few test files:

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

2. Force tapes to be immediately written:

   ```
   node1# dmdidle
   ```

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

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

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

   ```
   node1# dmget files_to_test
   ```

NFS Standard Service

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

To test the NFS service, do the following:

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

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

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

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

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

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

DMF Manager Standard Service

To verify that DMF Manager is operational, start it according to the directions in *DMF 5 Administrator's Guide for SGI InfiniteStorage* and then access it by pointing your browser to the following address:

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

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

DMF Client SOAP Standard Service

To verify that DMF client SOAP service is operational, start it according to the directions in *DMF 5 Administrator's Guide for SGI InfiniteStorage* and then access it by pointing your browser to the following address:

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

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

CXFS NFS Edge-Serving HA Resource Examples

As an example, this chapter tells you how to configure the set of resources required to use CXFS NFS edge-serving in a two-node High Availability Extension (HAE) cluster.

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

This chapter contains the following sections:

CXFS NFS Edge-Serving HAE Example Procedure

To use CXFS NFS edge-serving in an active/active HAE environment, do the following:

1. Invoke the HAE GUI:

   ```
   node1# crm_gui
   ```

2. Log in to the initialized cluster (see step 7 in Chapter 4, "Outline of the Configuration Procedure" on page 27).

3. Create an anonymous clone containing a group and resources:

   a. Select **Resources** in the left-hand navigation panel.

   b. Click the **Add** button, select **Clone**, and click **OK**.

   c. Enter the ID of the clone, such as `cxfs-nfs-clone`.

d. Select **Stopped** for the **Initial state of resource** and click **Forward**.

e. Select the sub-resource **Group** and click **OK**.

f. Enter the ID of the resource group (such as `cxfs-nfs-group`). Use the default initial state of **Defaults to "Stopped" or inherit from its parent** and click **Forward**.

g. Select the sub-resource **Primitive** and click **OK**.

h. Add the primitive for "CXFS Client Resource" on page 44.

i. Add the primitive for "CXFS Client NFS Server Resource" on page 46.

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

k. Test the resources added in steps h and i:

- "Testing the CXFS Client Resource" on page 46
- "Testing the CXFS Client NFS Server Resource" on page 49

4. Create two IP alias groups, one for each rack, using the following sequence for each:

   a. Select **Resources** in the left-hand navigation panel.

   b. Click the **Add** button, select **Group**, and click **OK**.

   c. Enter the ID of the group, such as `ipalias-group-1`. Use the default initial state of **Stopped** and click **Forward**.

   d. Select the sub-resource **Primitive** and click **OK**.

   e. Add the primitive for "Virtual IP Address Resource" on page 50.

   f. Add the primitive for "CXFS Client NSM Notification Resource" on page 53.

   g. Click **Cancel** to stop adding primitives.

5. Click **Apply** to apply the resource group.

6. Select **Constraints** in the left-hand navigation panel.

7. Click the **Add** button, select **Resource Colocation**, and click **OK**.

Create two colocation constraints so that the virtual IP addresses must run on a system that has NFS, one constraint for each rack, using the following sequence for each:

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

b. For **Resource**, select the primitive created in step 4e above, such as **ipalias-rack1**.

c. For **With Resource**, select the clone created in step 3c above, such as **cxfs-nfs-clone**

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

e. Click **OK**.

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

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

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

   b. For **First**, select the name of the clone created in step 3c above, such as **cxfs-nfs-clone**.

   c. For **Then**, select the notification primitive created in step 4c, such as **ipalias-group-1**.

   d. Under **Optional**, set **Symmetrical** to **true**.

   e. Click **OK**.

9. Test the resources added in steps 4e and 4f:

   - "Testing the Virtual IP Address Resource" on page 52
   - "Testing the CXFS Client NSM Notification Resource" on page 56

CXFS Client Resource

This section discusses the following:

Creating the CXFS Client Primitive

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

Note: When upgrading CXFS software, `cxfs_client` is automatically set to turn on after reboot. However, only the HAE software must control the starting of the `cxfs_client` service. You must do the following:

1. Run the following commands before attempting to start the `cxfs_client` resource:

   ```
   # service cxfs_client stop
   # chkconfig cxfs_client off
   ```

2. Re-run the following command after upgrading:

   ```
   # chkconfig cxfs_client off
   ```

Required Fields for CXFS Client

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

Instance Attributes for CXFS Client

volnames	*Comma-separated list of XVM volume names containing CXFS filesystems (one to be served via NFS, the other to store the NFS state) such as:* `cxfsvol1,cxfsvol2`

Operations for CXFS Client

Monitor operation:

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

Probe monitor operation:

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

Start operation:

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

Optional > On Fail	**restart**

Stop operation:

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

Testing the CXFS Client Resource

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

1. Ensure that the resource is started by selecting it and clicking the **Play** button.
2. Reboot the active edge server node.

   Alternatively, you could do the following:

   a. Select the active edge server node and click **Make the node standby** in the GUI, so that `cxfs-client` goes down, or use a command such as the following if node1 is the active node:

      # **`crm_standby -U node1 -v true`**

   b. Select the node and click **Make the node active** in the GUI, or use a command such as the following to change node1 from standby to active:

      # **`crm_standby -U node1 -v false`**

3. Verify that the filesystem mounts after the edge server comes back up.

CXFS Client NFS Server Resource

This section discusses the following:

Configuring CXFS Client NFS for HA

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

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

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

   ---

2. Disable the NFS server from starting automatically on boot on both node1 and node2:

   ```
   node1# chkconfig nfsserver off
   node2# chkconfig nfsserver off
   ```

3. Add the CXFS client NFS resource primitive. See "Creating the CXFS Client NFS Server Primitive" on page 47.

Creating the CXFS Client NFS Server Primitive

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

Required Fields for CXFS Client NFS Server

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

Instance Attributes for CXFS Client NFS Server

nfs_init_script	*Location of the NFS initialization script, such as:* `/etc/init.d/nfsserver`

statedir	*Directory located on the NFS filesystem used to store NFS state (equivalent to* `/var/lib/nfs/` *in a nonclustered configuration), such as:* `/mnt/cxfsvol2/statd/nfs2-nfs3`
	Note: If you have multiple HAE clusters for CXFS NFS edge-serving within one CXFS cluster, each must have a separate state directory and a unique **statedir** value.
statefile	*Filename located on the NFS state filesystem (equivalent to* `/var/lib/nfs/state` *in a nonclustered configuration), such as:* `/mnt/cxfsvol2/statd/state`
	Note: All of the resources must share this file and use the same **statefile** value. (This applies if there is one HAE cluster or if there are multiple HAE clusters.)
volnames	*Comma-separated list of all CXFS filesystems that will be edge-served via NFS*

Operations for CXFS Client NFS Server

Monitor operation:

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

Probe monitor operation:

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

Timeout	*Timeout, such as* `60s`
Start operation:	
ID	*(Generated based on the primitive name and interval)*
name	**start**
Timeout	*Timeout, such as* `300s`
Optional > Requires	**fencing**
Optional > On Fail	**restart**
Stop operation:	
ID	*(Generated based on the primitive name and interval)*
name	**stop**
Timeout	*Timeout, such as* `600s`
Optional > On Fail	**fence**

Testing the CXFS Client NFS Server Resource

To test the `cxfs-client-nfsserver` client resource, do the following:

1. Ensure that the resource is started by selecting it and clicking the **Play** button.
2. Select the active node and click **Make the node standby** in the GUI, or use a command such as the following if node1 is the active node:

   ```
   # crm_standby -U node1 -v true
   ```

3. Verify that the status is `unused`:

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

4. Select the node and click **Make the node active** in the GUI, or use a command such as the following to change node1 from standby to active:

   ```
   # crm_standby -U node1 -v false
   ```

5. Verify that the status is `running`:

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

Virtual IP Address Resource

This section discusses the following:

Creating the Virtual IP Address Primitive

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

Required for Virtual IP Address

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

Instance Attributes for Virtual IP Address

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

broadcast	*(Optional) Broadcast IP address, such as* `128.162.244.255`

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

Meta Attributes for Virtual IP Address

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

Operations for Virtual IP Address

Probe monitor operation:

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

Start operation:

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

Stop operation:

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

Optional > On Fail fence

Testing the Virtual IP Address Resource

To test the IPaddr2 resource, do the following:

1. Ensure that the resource is started by selecting it and clicking the **Play** button.
2. Verify that the IP address is configured correctly on node1:

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

3. Verify that node2 does not accept the IP address packets by running the following command on node2 (the output should be 0):

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

4. 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:

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

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

```
node1# crm_resource -r ipalias-group-1 -M -H node2
```

6. 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
```

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

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

8. 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`:

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

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

```
node1# crm_resource -r ipalias-group-1 -M -H node1
```

10. Test again as in steps 2-4 above.

11. Remove the resource location constraints generated by the administrative move commands executed above:

```
node1# crm_resource -r ipalias-group-1 -U
```

CXFS Client NSM Notification Resource

This section discusses the following:

Creating the CXFS Client NSM Notification Primitive

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

Required Fields for CXFS Client NSM Notification

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

Type	**cxfs-client-smnotify**

Instance Attributes for CXFS Client NSM Notification

ipalias	*IP address of the IP alias associated with the NFS client lock state, which is reclaimed by the NFS client from the NFS server when it receives the NSM reboot notification that is initiated by the* `cxfs-client-smnotify` *resource agent, for example for hostalias1* `128.162.244.244`
statedir	*Identical to the* **statedir** *value specified above for* **cxfs-client-nfsserver** (see "Instance Attributes for CXFS Client NFS Server" on page 47)
statefile	*Identical to the* **statefile** *value specified above for* **cxfs-client-nfsserver**
pidfile	*Filename located on the NFS state filesystem that specifies the process ID, such as:* `/mnt/cxfsvol2/statd/smnotify-rack1.pid` **Note:** There must be a separate file and unique **pidfile** value for each **cxfs-client-smnotify** primitive.
gracedir	*Directory on the NFS state filesystem that specifies the file containing the grace-period state, such as:* `/mnt/cxfsvol2/grace` **Note:** If you have multiple HAE clusters for CXFS NFS edge-serving within one CXFS cluster, all of the clients must share this file and use the same **gracedir** value.
hostname	*Hostname of* **ipalias***, which must match what is in* `/etc/hosts` *or be resolvable with DNS*
seconds	*The number of seconds before the grace period expires (must be the same on all resources in the cluster), such as* `120` **Note:** This value is always in seconds, unlike other timeout values, so you must not include the `s` identifier.

volnames	*Comma-separated list of all volumes that will be served via* **ipalias**, *such as* `cxfsvol2`

Meta Attributes for CXFS Client NSM Notification

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

Operations for CXFS Client NSM Notification

Monitor operation:

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

Probe monitor operation:

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

Start operation:

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

Stop operation:

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

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

Testing the CXFS Client NSM Notification Resource

To test the `cxfs-client-smnotify` resource, do the following:

1. Ensure that the resource is started by selecting it and clicking the **Play** button.
2. Mount the filesystem via each **ipalias hostname** on a system that is outside the HAE cluster (for example, `nfsclient`). For example:

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

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

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

4. Acquire locks:

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

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

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

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

7. Select one node and click **Make the node standby** in the GUI or use a command such as the following, for example to make node1 standby:

   # **crm_standby -U node1 -v true**

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

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

DMF HA Cluster Resource Examples

As an example, this chapter tells you how to configure the set of resources required to use DMF in a two-node active/passive High Availability Extension (HAE) cluster.

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

This chapter contains the following sections:

DMF HAE Example Procedure

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

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

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

1. Invoke the HAE GUI:

   ```
   node1# crm_gui
   ```

   See the information about the setting the password and using the HAE GUI (`crm_gui`) in:

   - "Preliminary Best Practices" on page 9
   - "HAE Configuration Best Practices" on page 10

2. Log in to the initialized cluster (see step 7 in Chapter 4, "Outline of the Configuration Procedure" on page 27).
3. Select **Resources** in the left-hand navigation panel.
4. Click the **Add** button, select **Group**, and click **OK**.
5. Enter the ID of the resource group (such as `dmfGroup`).
6. Set the **target-role** meta attribute for `dmfGroup` to **Started** and click **Forward**.
7. Select **OK** to add a **Primitive**. Add and test the CXFS or local XVM primitive, according to the steps described in either:

   - "CXFS Resource" on page 61
   - "Local XVM Resource" on page 64 and "Filesystem Resources" on page 67

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

   a. "Virtual IP Address Resource" on page 75

b. A mounting service, either:

- "OpenVault Resource" on page 78
- "TMF Resource" on page 88

c. "DMF Resource" on page 94

d. "NFS Resource" on page 99

e. "DMF Manager Resource" on page 104

f. "DMF Client SOAP Service Resource" on page 106

CXFS Resource

This section discusses examples of the following:

- "Creating the CXFS Primitive" on page 61
- "Testing the CXFS Resource" on page 63

Creating the CXFS Primitive

Required Fields for CXFS

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

Instance Attributes for CXFS

volnames	*Comma-separated list of CXFS volume names, such as:* `cxfsvol1,cxfsvol2`

Meta Attributes for CXFS

resource-stickiness	**1**

migration_threshold	**1**

Operations for CXFS

Monitor operation:

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

Probe monitor operation:

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

Start operation:

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

Stop operation:

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

Optional > On Fail fence

Testing the CXFS Resource

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

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

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

   ```
   node1# df -lh
   ```

   b. Display the current metadata server for the filesystems:

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

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

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

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

   ```
   node1# crm_resource -r dmfGroup -M -H node2
   ```

3. Verify that CXFS is working on node2:

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

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

   ```
   node1# crm_resource -r dmfGroup -M -H node1
   ```

5. Verify that CXFS is working again on node1:

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

6. Remove the resource location constraints generated by the administrative move commands executed above:

```
node1# crm_resource -r dmfGroup -U
```

Local XVM Resource

This section discusses examples of the following:

Creating the Local XVM Primitive

Required Fields for Local XVM

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

Instance Attributes for Local XVM

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

```
openvault,home,journals,spool,move,tmp,diskmsp,dmfusr1dmfusr3
```

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

```
myCluster,myClusterStripe1,myClusterStripe2
```

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

Meta Attributes for Local XVM

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

Operations for Local XVM

Monitor operation:

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

Probe monitor operation:

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

Start operation:

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

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

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

Stop operation:

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

Testing the Local XVM Resource

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

1. On node1, unmount all of the filesystems for which a `Filesystem` primitive will be defined (in "Filesystem Resources" on page 67).
2. Move the resource group containing the `lxvm` resource from node1 to node2:

   ```
   node1# crm_resource -r dmfGroup -M -H node2
   ```

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

   ```
   node1# crm_mon -1 | grep Timed
       lxvm_start_0 (node=node1, call=222, rc=-2): Timed Out

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

3. Look at `/dev/lxvm` on node2 to verify the change. Each of the volumes listed in the `volnames` attribute should appear within the `/dev/lxvm` directory. (There may be additional volumes within the directory.)
4. On node2, unmount all of the filesystems for which a `Filesystem` primitive will be defined (in "Filesystem Resources" on page 67).
5. Move the resource group containing the `lxvm` resource back to node1:

   ```
   node1# crm_resource -r dmfGroup -M -H node1
   ```

6. Look at `/dev/lxvm` on node1 to verify the change as described in step 3.
7. Remove the resource location constraints generated by the administrative move commands executed above:

   ```
   node1# crm_resource -r dmfGroup -U
   ```

Filesystem Resources

This section discusses examples of the following:

Filesystems Supported

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

- DMF-managed user filesystems

Note: The optional `MOVE_FS` DMF administrative filesystem and all DMF-managed user filesystems require `dmi` and `mtpt` mount options to be specified when configuring their resources.

- 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 DCMs and disk MSPs using local disk storage

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

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

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

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

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

- Edit `/etc/fstab` and remove all of the filesystems that you will manage with HAE
- Add a filesystem primitive using the values show in one of the following, as appropriate:
  - "Creating a DMF-Managed User Filesystem Primitive" on page 69
  - "Creating a DMF Administrative Filesystem Primitive" on page 71

– "Creating a Dedicated OpenVault Server Filesystem Primitive *(Optional)*" on page 72

Creating a DMF-Managed User Filesystem Primitive

Required Fields for DMF-Managed User Filesystem

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

Instance Attributes for DMF-Managed User Filesystem

device	*The* `/dev/lxvm` *volume name of the DMF spool filesystem device, such as* `/dev/lxvm/dmfusr1`
directory	*The mount point for the DMF spool filesystem, such as* `/dmfusr1`
options	*The mount options*
	Note: The value of the `mtpt` mount option must match the value used for the **directory** attribute, such as `/dmfusr1`.
fstype	**xfs**

Meta Attributes for DMF-Managed User Filesystem

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

Operations for DMF-Managed User Filesystem

Monitor operation:

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

Probe monitor operation:

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

Start operation:

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

Stop operation:

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

Creating a DMF Administrative Filesystem Primitive

Required Fields for DMF Administrative Filesystem

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

Instance Attributes for DMF Administrative Filesystem

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

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

You do not need the **options** attribute for any DMF administrator filesystem except the optional `MOVE_FS` filesystem, for which you must specify the `dmi` and `mtpt` mount options.

---

fstype	**xfs**

Meta Attributes for DMF Administrative Filesystem

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

Operation for DMF Administrative Filesystem

Monitor operation:

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

Interval	*Interval time, such as* `120s`
Timeout	*Timeout, such as* `60s`
Optional > On Fail	**restart**

Probe monitor operation:

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

Start operation:

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

Stop operation:

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

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

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

Required Fields for OpenVault Server Filesystem

ID	*Unique ID such as* `openvaultfs`

Class	**ocf**
Provider	**heartbeat**
Type	**Filesystem**

Instance Attributes for OpenVault Server Filesystem

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

Meta Attributes for OpenVault Server Filesystem

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

Operations for OpenVault Server Filesystem

Monitor operation:

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

Probe monitor operation:

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

Start operation:

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

Stop operation:

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

Testing Filesystem Resources

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

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

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

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

   ```
   node1# crm_resource -r dmfGroup -M -H node2
   ```

3. 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.
4. Move the resource group containing all of the `Filesystem` resources back to node1:

   ```
   node1# crm_resource -r dmfGroup -M -H node1
   ```
5. 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.
6. Remove the resource location constraints generated by the administrative move commands executed above:

   ```
   node1# crm_resource -r dmfGroup -U
   ```

Virtual IP Address Resource

This section discusses examples of the following:

- "Creating the Virtual IP Address Primitive" on page 75
- "Testing the Virtual IP Address Resource" on page 77

Creating the Virtual IP Address Primitive

Required for Virtual IP Address

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

Type	**IPaddr2**

Instance Attributes for Virtual IP Address

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

Meta Attributes for Virtual IP Address

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

Operations for Virtual IP Address

Probe monitor operation:

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

Start operation:

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

Stop operation:

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

Testing the Virtual IP Address Resource

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

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

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

2. Verify that node2 does not accept the IP address packets by running the following command on node2 (the output should be null):

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

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

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

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

```
node1# crm_resource -r dmfGroup -M -H node2
```

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

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

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

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

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

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

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

```
node1# crm_resource -r dmfGroup -M -H node1
```

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

10. Remove the resource location constraints generated by the administrative move commands executed above:

```
node1# crm_resource -r dmfGroup -U
```

OpenVault Resource

This section discusses examples of the following:

Configuring OpenVault for HA

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

2. Add the primitive using the values shown in "Creating the OpenVault Primitive" on page 84.

3. Run `ov_admin` on node1:

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

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

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

b. Creating the directory specified by `serverdir`.

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

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

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

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

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

5. Configure the other DMF node by using the following steps, repeating the entire sequence for node2 before moving to step 6. Whenever `ov_admin` asks for the server hostname, use the virtual hostname, on both on node1 and node2.

   Complete the following series of steps for node2:

   a. On node1:

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

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

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

b. On node2:

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

```
node2# ov_admin
...
What is (or will be) the name of the OpenVault server? [virtualhostname]
```

ii. Configure drives by selecting:

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

iii. Configure libraries by selecting:

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

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

- When asked for the name of the device, use the same library name that was used on node1. The LCP instance name will automatically reflect node2 name (for example, for the `l700a` library, the LCP instance name on node1 is `l700a@node1` and the LCP instance name on node2 will be `l700a@node2`).
- When prompted with `Library 'libname' already exists in OpenVault catalog; create LCP anyway?`, respond `yes`.

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

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

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

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

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

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

d. On node2:

i. Verify that the DCPs are running successfully. For example, the following output shows under `DCPHost` and `DCPStateSoft` columns that the DCP is running and connected to the OpenVault server (`ready`) on the active HA node (`node1`) and running in standby mode (`disconnected`) on the standby HA node (`node2`):

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

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

ii. Verify that the LCPs are running. For example, the following output shows under `LCPHost` and `LCPStateSoft` columns that the LCP is

running and connected to the OpenVault server (`ready`) on the active HA node (`node1`) and running in standby mode (`disconnected`) on the standby HA node (`node2`):

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

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

iii. Stop all DCPs and LCPs on node2:

```
node2# ov_stop
```

iv. Disable OpenVault from being started automatically during the boot process:

```
node2# chkconfig openvault off
```

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

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

```
parallel_node# ov_admin
...
What is (or will be) the name of the OpenVault server?  [servername] virtualhostname
What port number is the OpenVault server on virtualhostname using? [44444]
What security key would you like the admin commands to use? [none]
```

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

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

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

```
parallel_node# service openvault stop
parallel_node# service openvault start
```

7. On node1, stop the OpenVault server and any DCPs and LCPs and turn OpenVault off on node1 upon reboot:

```
node1# ov_stop
node1# chkconfig openvault off
```

8. Update the contents of the OpenVault resource so that `is-managed` is set to `true` in the CIB, thus allowing HAE to manage the resource immediately:

```
node1# crm_resource -r OpenVault_resourcePRIMITIVE -m -p is-managed -v true
```

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

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

Do the following for each OpenVault client:

a. On node1:

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

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

```
23 - Manage OpenVault Client Machines
        1 - Activate an OpenVault Client Machine
```

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

```
node2# ov_admin
...
What is (or will be) the name of the OpenVault server? [virtualhostname]
What port number is the OpenVault server on virtualhostname using? [44444]
What security key is used for admin commands on the HA OpenVault servers? [none]
```

Creating the OpenVault Primitive

Required Fields for OpenVault

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

Instance Attributes for OpenVault

virtualhost	*Hostname that will resolve as defined in* `/etc/nsswitch.conf` *to the IP address configured in* "Virtual IP Address Resource" on page 75*, such as* `128.162.244.240`
serverdir	*Directory containing the OpenVault server configuration, such as* `/dmf/home/openvault/server`

Note: serverdir must be a path on a mountable CXFS filesystem that is being managed by a `cxfs` resource or XFS filesystem that is being managed by a `Filesystem` resource in the same resource group as the `openvault` resource. The filesystem could be one dedicated for OpenVault use (such as `/dmf/openvault`) or it could be an HAE-managed filesystem in the same resource group that has sufficient space (such as `/dmf/home/`). As part of the conversion to HA, OpenVault will create this directory and move its database and logs into the directory, so the directory must not already exist or OpenVault will fail.

Meta Attributes for OpenVault

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

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

Operations for OpenVault

Monitor operation:

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

Probe monitor operation:

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

Start operation:

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

Stop operation:

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

Testing the OpenVault Resource

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

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

   ```
   node1# crm_resource -r dmfGroup -M -H node2
   ```

2. Verify that all the tape drives become available after a few moments. For example:

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

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

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

```
node1# crm_resource -r dmfGroup -M -H node1
```

4. Verify that all the tape drives become available after a few moments. For example:

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

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

5. Remove the resource location constraints generated by the administrative move commands executed above:

```
node1# crm_resource -r dmfGroup -U
```

TMF Resource

This section discusses examples of the following:

Configuring TMF for HA

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

1. Copy the following file from node1 to node2:

   `/etc/tmf/tmf.config`

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

2. On node2, execute the following to enable TMF startup during the boot process:

   ```
   node2# chkconfig tmf on
   ```

3. Create the TMF resource primitive with the fields shown in "Creating the TMF Primitive" on page 88.

Creating the TMF Primitive

Required Fields for TMF

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

Type	**tmf**

Instance Attributes for TMF

devgrpnames

Comma-separated list of TMF device groups defined in the `tmf.config` *file that are to be managed by HAE, such as:*

```
ibm3592,t10ka
```

mindevsup

Comma-separated list of the number of devices, one per device group, that must be configured up successfully within the corresponding device group in order to count the group as being highly available, such as:

```
1,0
```

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

---

devtimeout

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

```
120,240
```

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

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

admin_emails

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

```
root,admin1
```

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

loader_names

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

```
ibm3494,l700a
```

loader_hosts	*Comma-separated list of hosts through which the corresponding loaders listed in* **loader_names** *are controlled, such as:* `ibm3494cps,stkacsls`
loader_users	*Comma-separated list of user names that are used to log in to the corresponding hosts listed in* **loader_hosts**, *such as:* `root,acssa`
loader_passwords	*Comma-separated list of passwords corresponding to the user names listed in* **loader_users**, *such as:* `passwd1,passwd2`

Meta Attributes for TMF

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

Operations for TMF

Monitor operation:

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

Probe monitor operation:

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

Start operation:

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

name	**start**
Timeout	*Timeout, such as* `236s`

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

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

Stop operation:

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

Testing the TMF Resource

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

1. Use `tmstat` to verify that all of the tape drives in all HAE-managed device groups are in `assn` or `idle` status on node1.
2. Move the resource group containing the `tmf` resource to node2:

   ```
   node1# crm_resource -r dmfGroup -M -H node2
   ```

3. 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`

4. Verify that the `timeout` values for the `start`, `stop`, and `monitor` operations are appropriate. Do the following:

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

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

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

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

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

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

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

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

5. If you need to change any values, edit the primitive using `crm_gui`.

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

   ```
   node1# crm_resource -r dmfGroup -M -H node1
   ```

7. 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`

8. Remove the resource location constraints generated by the administrative move commands executed above:

   ```
   node1# crm_resource -r dmfGroup -U
   ```

DMF Resource

This section discusses examples of the following:

- "Configuring DMF for HA" on page 94
- "Creating the DMF Primitive" on page 97
- "Testing the DMF Resource " on page 98

Configuring DMF for HA

Note: The following procedure requires that the DMF application instances in OpenVault are configured to use a wildcard ("*") for the hostname and instance name. For more information, see the procedure about configuring DMF to use OpenVault in the *DMF 5 Administrator's Guide for SGI InfiniteStorage.*

To configure DMF for HA, do the following:

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

   The backup of DMF-managed user filesystems and DMF administrative filesystems is always performed on the active DMF server based upon parameters in the DMF configuration file. The `xfsdump` command maintains an inventory of all backups performed within the directory `/var/lib/xfsdump`, but in an HAE 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 HAE-managed DMF administrative filesystem within the same resource group as DMF. For example, create a site-specific directory in DMF's HOME_DIR, such as /dmf/home/site_specific:

- On node1 (which currently contain the inventory), enter the following:

```
node1# cd /var/lib
node1# cp -r xfsdump /dmf/home/site_specific/xfsdump
node1# mv xfsdump xfsdump.bak
node1# ln -s /dmf/home/site_specific/xfsdump xfsdump
```

  Note: In a brand-new DMF installation, the /var/lib/xfsdump 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 dmf.conf configuration file as follows:

   - Set the MAX_MS_RESTARTS parameter in the appropriate drive group objects to 0 so that DMF will not restart the mounting service.
   - Set the DUMP_INVENTORY_COPY parameter so that it uses a DMF HA administrative filesystem that is on a different disk from the live inventory created above in step 1. If the live inventory in /dmf/home/site_specific/xfsdump is lost, you can then recreate it from the inventory backup in DUMP_INVENTORY_COPY. For example, you could create the directory /dmf/journal/site_specific/inventory_copy for use in DUMP_INVENTORY_COPY.

- If you are using OpenVault, set the MSG_DELAY parameter in the drivegroup objects to a value of slightly more than 2 minutes.
- Set the SERVER_NAME parameter for the base object to the HA virtual hostname of the DMF server.

  Note: If you change this parameter, you must copy the dmf.conf file manually to each parallel data mover node and then restart the DMF services. Do not change this parameter while DMF is running.

- If using the DMF Parallel Data Mover Option:
  - Create node objects for each server in the HAE cluster.
  - Set the HA_VIRTUAL_HOSTNAME parameter for potential DMF server nodes to the same virtual hostname used for SERVER_NAME for the base object. Parallel data mover nodes should not define this parameter.

For more information, see the dmf.conf(5) man page and the *DMF 5 Administrator's Guide for SGI InfiniteStorage.*

3. Copy the DMF configuration file (/etc/dmf/dmf.conf) from node1 to node2 and to any parallel data mover nodes in the DMF configuration. You may wish to use a symbolic link on node1 and on node2 that points to a shared location in the HOME_DIR directory. For example:

   ```
   # 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, edit the ov_keys file and replace the hostname in field 1 of the DMF lines with the OpenVault virtual hostname. For example, if the virtual hostname is virtualhost:

   ```
   virtualhost   dmf   *  CAPI none
   virtualhost   dmf   *  AAPI none
   ```

   Note: If you used a wildcard hostname (*) when you defined your ov_keys file during initial OpenVault setup, there is no need to edit this file.

5. Disable the automatic start of DMF during boot on all DMF server nodes in the HAE cluster:

```
dmfserver# chkconfig dmf off
dmfserver# chkconfig dmfman off
dmfserver# chkconfig dmfsoap off
```

6. Create the DMF resource with the fields shown in "Creating the DMF Primitive" on page 97.

Creating the DMF Primitive

Required Fields for DMF

ID	*Unique ID, such as* `dmf`
Class	**ocf**
Provider	**sgi**
Type	**dmf**

Instance Attributes for DMF

monitor_level	*Level that specifies whether to check for the existence of the dmfdaemon process only (*`0`*) or also run an additional check using* `dmstat` *(*`1`*)*

Meta Attributes for DMF

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

Operations for DMF

Monitor operation:

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

Optional > On Fail	**restart**

Probe monitor operation:

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

Start operation:

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

Stop operation:

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

Testing the DMF Resource

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

1. Verify that DMF has started by using the `dmdstat -v` command and manual `dmput` and `dmget` commands on node1:

   ```
   node1# dmdstat -v
   node1# xfs_mkfile size test_file
   node1# dmput -r test_file
   node1# dmdidle
   (wait a bit to allow time for the tape to be written and unmounted)
   ```

```
node1# dmget test_file
node1# rm test_file
```

2. Move the resource group containing the dmf resource to node2 (because the mounting service is in the same resource group, it must be colocated and thus should failover with DMF to the new node):

```
node1# crm_resource -r dmfGroup -M -H node2
```

3. Verify that DMF has started on the new node by using the dmdstat -v command and manual dmput and dmget commands on node2:

```
node2# dmdstat -v
node2# xfs_mkfile size another_test_file
node2# dmput -r another_test_file
node2# dmdidle
(wait a bit to allow time for the tape to be written and unmounted)
node2# dmget another_test_file
node2# rm another_test_file
```

4. Move the resource group containing the dmf resource back to node1:

```
node1# crm_resource -r dmfGroup -M -H node1
```

5. Verify that DMF has started by using the dmdstat -v command and manual dmput and dmget commands on node1:

```
node1# dmdstat -v
node1# xfs_mkfile size test_file
node1# dmput -r test_file
node1# dmdidle
(wait a bit to allow time for the tape to be written and unmounted)
node1# dmget test_file
node1# rm test_file
```

6. Remove the resource location constraints generated by the administrative move commands executed above:

```
node1# crm_resource -r dmfGroup -U
```

NFS Resource

This section discusses examples of the following:

Configuring NFS for HA

To configure NFS for HA, do the following:

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

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

2. Disable the NFS server from starting automatically on boot on both node1 and node2:

   ```
   node1# chkconfig nfsserver off
   node2# chkconfig nfsserver off
   ```

3. Add the NFS resource primitive. See "Creating the NFS Primitive" on page 100.

Creating the NFS Primitive

Required Fields for NFS

ID	*Unique ID, such as* `nfs`
Class	**ocf**
Provider	**heartbeat**
Type	**nfsserver**

Instance Attributes for NFS

nfs_init_script	*NFS initialization script, such as* `/etc/init.d/nfsserver`
nfs_notify_cmd	*NFS notification command, such as* `/usr/sbin/sm-notify`

nfs_shared_infodir	*Site-specific NFS shared-information directory, such as a* `/mnt/cxfsvol1/.nfs` *subdirectory in the exported filesystem*
nfs_ip	*The same IP address specified for the* **ip** *field for the virtual IP resource* (see "Instance Attributes for Virtual IP Address" on page 76)

Meta Attributes for NFS

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

Operations for NFS

Monitor operation:

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

Probe monitor operation:

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

Start operation:

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

Optional > On Fail	**restart**

Stop operation:

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

Testing the NFS Resource

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

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

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

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

```
otherhost# mount node1:/nfsexportedfilesystem /mnt/test
```

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

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

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

```
node1# crm_resource -r dmfGroup -M -H node2
```

5. Run the following command on node2 to verify that the NFS filesystems are exported:

```
node2# exportfs -v
/work.mynode1   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=8001)
/work.mynode2   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=8002)
/work.mynode3   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=8003)
/work.mynode4   <world>(rw,wdelay,root_squash,no_subtree_check,fsid=8004)
/mirrors        <world>(ro,wdelay,root_squash,no_subtree_check,fsid=8005)
/               <world>(ro,wdelay,root_squash,no_subtree_check,fsid=8006)
```

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

```
otherhost# echo "test data for another test file" > /mnt/test/testFile1B
otherhost# cat /mnt/test/testFile1B
test data for another test file
```

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

```
node1# crm_resource -r dmfGroup -M -H node1
```

8. Remove the resource location constraints generated by the administrative move commands executed above:

```
node1# crm_resource -r dmfGroup -U
```

DMF Manager Resource

This section discusses examples of the following:

Configuring DMF Manager for HA

To configure DMF Manager for HA, do the following:

1. Disable the DMF Manager tool from starting automatically on boot (execute on both nodes):

   # **chkconfig dmfman off**

2. Add a primitive for DMF Manager using the values shown in "Creating the DMF Manager Primitive" on page 104.

Creating the DMF Manager Primitive

Required Fields for DMF Manager

ID	*Unique ID, such as* dmfman
Class	**ocf**
Provider	**sgi**
Type	**dmfman**

Meta Attributes for DMF Manager

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

Note: You might want to set the **migration-threshold** higher than 1 for this resource so that it will simply restart in place.

Operations for DMF Manager

Note: You should only define a **monitor** operation for the `dmfman` resource if you want a failure of DMF Manager to cause a failover for the entire resource group.

Monitor operation:

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

Probe monitor operation:

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

Start operation:

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

Stop operation:

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

Testing the DMF Manager Resource

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

1. Point your browser at `https://`*virtualIPaddress*`:1179` and verify that you can log in and use DMF Manager, such as viewing the **Overview** panel. For more information about using DMF Manager, see *DMF 5 Administrator's Guide for SGI InfiniteStorage*.
2. Move the resource group containing the `dmfman` resource from node1 to node2:

   ```
   node1# crm_resource -r dmfGroup -M -H node2
   ```

3. Repeat step 1 to verify that DMF Manager is still available.
4. Move the resource group containing the `dmfman` resource back to node1:

   ```
   node1# crm_resource -r dmfGroup -M -H node1
   ```

5. Remove the resource location constraints generated by the administrative move commands executed above:

   ```
   node1# crm_resource -r dmfGroup -U
   ```

DMF Client SOAP Service Resource

This section discusses examples of the following:

Configuring DMF Client SOAP Service for HA

To configure DMF client SOAP service for HA, do the following:

1. Disable the DMF client SOAP service tool from starting automatically on boot (execute on both nodes):

   # **chkconfig dmfsoap off**

2. Add a primitive for DMF client SOAP service using the values shown in "Creating the DMF Client SOAP Service Primitive" on page 107.

Creating the DMF Client SOAP Service Primitive

Required Fields for DMF Client SOAP Service

ID	*Unique ID, such as* `dmfsoap`
Class	**ocf**
Provider	**sgi**
Type	**dmfsoap**

Meta Attributes for DMF Client SOAP Service

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

Note: You might want to set the **migration-threshold** higher than 1 for this resource so that it will simply restart in place.

Operations for DMF Client SOAP Service

Note: You should only define a **monitor** operation for the `dmfsoap` resource if you want a failure of DMF client SOAP service to cause a failover for the entire resource group.

Monitor operation:

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

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

Probe monitor operation:

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

Start operation:

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

Stop operation:

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

Testing the DMF Client SOAP Service Resource

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

1. Point your browser at `https://`*virtualIPaddress*`:1180/server.php` and verify that you can access the GUI and view the WSDL for one of the DMF client functions. For more information, see *DMF 5 Administrator's Guide for SGI InfiniteStorage*.
2. Move the resource group containing the `dmfsoap` resource from node1 to node2:

   ```
   node1# crm_resource -r dmfGroup -M -H node2
   ```

3. Repeat step 1 to verify that DMF client SOAP service is still available.
4. Move the resource group containing the `dmfsoap` resource back to node1:

   ```
   node1# crm_resource -r dmfGroup -M -H node1
   ```

5. Remove the resource location constraints generated by the administrative move commands executed above:

   ```
   node1# crm_resource -r dmfGroup -U
   ```

STONITH Resource Examples

This chapter discusses the following:

Overview of STONITH Resources

STONITH (“*shoot the other node in the head*”) node-level fencing is required in order to protect data integrity in case of failure:

- `l2network` for ia64 systems using L2 controllers, such as SGI Altix® 450 systems
- `sgi-ipmi` for x86-64 systems using baseboard management controller (BMC) and intelligent platform management interface (IPMI) network reset, such as SGI Altix XE systems

L2 STONITH Examples

This section discusses examples of the following:

For more information, see the *SGI L1 and L2 Controller Software User’s Guide* and the user guide or quick start for your system.

Creating the L2 STONITH Clone

To create the STONITH clone, do the following:

1. Select **Resources** in the left-hand navigation panel.
2. Click the **Add** button, select **Clone**, and click **OK**.
3. Enter the name of the clone (such as `stonith-l2network-set`) in the **ID** field.
4. Use the **target-role** of **Started** and the default options (2 maximum number of copies and 1 number of copies on a single node) and click **Forward**.
5. Select **OK** to add a **Primitive**. Add the STONITH primitive according to the steps described in "Creating the L2 STONITH Primitive " on page 112.

Creating the L2 STONITH Primitive

Required Fields for L2 STONITH

ID	*Unique ID such as* `stonith-l2network`
Class	**stonith**
Type	**l2network**

Instance Attributes for L2 STONITH

nodelist

Nodes to be acted upon, such as:

```
node1;128.162.245.170;;3 node2;128.162.245.170;;4
```

You must provide the following information for each node (each field is separated by a semicolon), and each node is separated by a space:

nodename`;`*L2_ipaddr*`;`*L2passwd*`;`*partition*

where the fields are:

- Node name (such as `node1`)
- L2 IP address (such as `128.162.245.170`)

- L2 password (an empty field indicates that the L2 has no password)
- Machine partition (such as `3`); use `0` for a nonpartitioned system, which is the most common circumstance

---

Note: The following command shows the partition ID on an SGI ia64 system:

```
# cat /proc/sgi_sn/partition_id
```

Operations for L2 STONITH

Monitor operation:

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

Probe monitor operation:

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `300s`

Start operation:

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

Testing the L2 STONITH Node-Level Fencing Service

To test the L2 STONITH node-level fencing service, do the following:

1. Enter the following command:

```
# stonith -t l2network -T reset nodelist="nodelist" machine_to_reset
```

For example, to reset `node1` (line breaks here shown for readability):

```
# stonith -t l2network -T reset \
nodelist="node1;128.162.245.170;;3 node2;128.162.245.170;;4" node1
** INFO: Initiating l2network-reset on node1 via L2 128.162.245.170, partition 3
```

2. Verify that the specified node was reset and was able to successfully complete a reboot.

IPMI STONITH Examples

This section discusses examples of the following:

Creating the IPMI STONITH Clone

To create the IPMI STONITH clone, do the following:

1. Select **Resources** in the left-hand navigation panel.
2. Click the **Add** button, select **Clone**, and click **OK**.
3. Enter the name of the clone (such as `stonith-sgi-ipmi-set`) in the **ID** field.
4. Use the **target-role** of **Started** and the default options (2 maximum number of copies and 1 number of copies on a single node) and click **Forward**.
5. Select **OK** to add a **Primitive**. Add the STONITH primitive according to the steps described in "Creating the L2 STONITH Primitive " on page 112.

Creating the IPMI STONITH Primitive

Required Fields for IPMI STONITH

ID	*Unique ID such as* `stonith-sgi-ipmi`
Class	**stonith**
Type	**external/sgi-ipmi**

Instance Attributes for IPMI STONITH

nodelist — *Nodes to be acted upon, such as:*

```
node1;admin;admin;supermicro;128.162.245.170
node2;admin;admin;supermicro;128.162.245.171
```

You must provide the following information for each node (each field is separated by a semicolon), and each node is separate by a space:

nodename`;`*userID*`;`*IPMIpasswd*`;`*BMCtype*`;`*IPMIipaddr1*`[;`*IPMIipaddrN*`]`

where the fields are:

- Node name (such as `node1`)
- User ID to use on the IPMI device (such as the default `admin`)
- IPMI device password (such as the default `admin`)
- BMC type of the IPMI device:
  - `intel` for Intel® BMC
  - `supermicro` for Supermicro® BMC
- IPMI device IP address (such as `128.162.245.170`)

Operations for IPMI STONITH

Monitor operation:

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

Probe monitor operation:

ID	*(Generated based on the primitive name and interval)*
name	**monitor**
Interval	**0**
Timeout	*Timeout, such as* `300s`

Start operation:

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

Testing the IPMI STONITH Node-level Fencing Service

To test the IPMI STONITH node-level fencing service, do the following:

1. Enter the following command:

```
# stonith -t external/sgi-ipmi -p "nodelist_value" -T reset node_to_be_reset
```

For example, to reset node1:

```
# stonith -t external/sgi-ipmi -p "node1;admin;admin;supermicro;128.162.245.170" -T reset node1
```

2. Verify that the specified node was reset and was able to successfully complete a reboot.

Enabling System Reset

To enable system reset (which was disabled for testing purposes in step 12 in Chapter 4, "Outline of the Configuration Procedure" on page 27), enter the following:

```
node1# crm_attribute -t crm_config -n stonith-enabled -v true
```

Note: After all of the STONITH resources are configured and tested, ensure that any constraints remaining in the cluster are appropriate for a production environment. To remove any remaining hostname constraint, enter the following:

```
node1# crm_resource -t group -r resourceGROUP -r -U
```

Administrative Tasks and Considerations

This chapter discusses various administrative tasks and considerations for a High Availability Extension (HAE) cluster:

Understanding CIFS and NFS in an HAE 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.

Using the DMF `run_daily_drive_report` Task

The `run_daily_drive_report` DMF task tells you about drives that need cleaning. It uses `tsreport` and `ts` log files that are on the local host. Therefore, it will only tell you about cleaning notifications that are in files on the local host. It does not make any attempt to copy the `ts` log files from the alternate node.

For example, suppose `machineA` was the DMF server from 00:01 (12:01 am) until 12:00 (noon), when a failover to `machineB` occurred. Sometime during the morning, `driveB` reported that it needed cleaning. If `run_daily_drive_report` ran at 12:30 (12:30 pm) and `driveB` had not been used in that half hour, `run_daily_drive_report` would not report that `driveB` needed cleaning.

To work around this problem, use the `tsreport` command on all DMF server nodes.

Reviewing the Log File

You will find information about HAE in the `/var/log/messages` log file.

You can control logging of debug messages by setting the `debug` parameter in the `logging` stanza of the `/etc/corosync/corosync.conf` file (`off` by default):

```
logging{
...
debug: on|off
...
 }
```

Clearing the Resource Primitive Failcount

To clear resource primitive failcounts, either reboot the nodes or set the failcount to `0` by entering the following on all nodes for each resource primitive:

```
# crm_failcount -r resourcePRIMITIVE -v 0
```

Cleaning Up the Resource State

After you resolve the cause of `action` error messages in the `crm_mon` output, you should enter the following to clear the resource state administratively from all nodes in the cluster:

`crm_resource -r` *`resourcePRIMITIVE`* `-C`

To clear the resource state from a single node, add the `-H` option:

`crm_resource -r` *`resourcePRIMITIVE`* `-C -H` *`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.

Manually Issuing a System Reset

To manually issue a system reset, do the following:

- For ia64 systems, where *nodelist_value* is the value for the `nodelist` attribute as described in "L2 STONITH Examples" on page 111:

  `stonith -t l2network -p "`*nodelist_value*`" -T reset` *node_to_be_reset*

  For example, to reset `node1`:

  # **`stonith -t l2network -p "node1;128.162.245.170;;3" -T reset node1`**

  In the above command, `128.162.245.170` is the IP address of the L2 that has `node1` configured as partition 3.

- For x86–64 systems, where *nodelist_value* is the value of the `nodelist` attribute in "IPMI STONITH Examples" on page 114:

  `stonith -t external/sgi-ipmi -p "`*nodelist_value*`" -T reset` *node_to_be_reset*

  For example, to reset `node1`:

`stonith -t external/sgi-ipmi -p "node1;admin;admin;supermicro;128.162.245.170" -T reset node1`

In the above command, `node1` has a BMC responding at IP address `128.162.245.170`. The BMC is a Supermicro and has been configured with usercode `admin` and password `admin`.

If you enter the above command on node1, it will reboot node1. If you execute the command from node2, it will execute the IPMI power-off and power-on commands via the BMC at `128.162.245.170`.

In general, the `external/sgi-ipmi` STONITH agent will execute the reboot command if it is run on the node that will be reset or it will execute the IPMI power-off and power-on commands via the first responsive BMC at one of the IP addresses provided in *nodelist_value*.

Managing HAE Control of a Resource Group

To remove HAE control for a resource group, enter the following:

```
# crm_resource -r resourceGROUP -m -p is-managed -v false
```

To return control of the resource group to HAE, enter the following:

```
# crm_resource -r resourceGROUP -m -p is-managed -v true
```

Stopping HAE

To stop HAE on the local node, enter the following:

```
# service openais stop
```

Note: Stopping `openais` will cause a failover if there are resources running on the node. Depending on how things are defined and whether the resource `stop` actions succeed, it might even cause the node to be reset.

Controlling the Number of Historical Files

Each time the cluster information base (CIB) is updated, a new version is created and the older version is saved. These files reside in `/var/lib/heartbeat/crm`. SGI

recommends that you keep the number of files manageable by using the following `crm_config` options:

```
pe-error-series-max
pe-input-series-max
pe-warn-series-max
```

For more information, see the Novell *High Availability Guide.*

Controlling the HAE Timeout

HAE sends messages among the nodes to verify that each node is available. You can use the `crm_config cluster-delay` option to set the time required for this communication. If set to a value that is too small, a node may be assumed to be failed when in fact the message has just not yet been delivered to other nodes in the cluster.

For more information, see the Novell *High Availability Guide.*

Performing a Rolling Upgrade in an HAE Environment

Assuming that you have a two-node production HAE environment in place and want to perform a rolling upgrade with minimal testing, use the procedures in the following sections:

- "CXFS NFS Edge-Serving HAE Rolling Upgrade" on page 124
- "DMF HAE Rolling Upgrade" on page 125

Note: Some software may not allow the rolling upgrade. Such a situation might require an extended outage window with all resources down and HAE turned off, which would permit more thorough testing (similar to that done during the initial installation).

CXFS NFS Edge-Serving HAE Rolling Upgrade

Do the following for a rolling upgrade in a CXFS NFS edge-serving HAE cluster:

1. Read the *SGI InfiniteStorage Software Platform* release note, the release notes for the products you intend to upgrade, and any late-breaking caveats on Supportfolio™ to see if there are any changes to this procedure.
2. Ensure that the resource groups are running on node1:

   ```
   # crm_resource -r ipalias-group-1 -M -H node1
   # crm_resource -r ipalias-group-2 -M -H node1
   ```

3. Set node2 (the node you intend to upgrade) to standby:

   ```
   # crm_standby -U node2 -v true
   ```

   This will shut down `cxfs_client` on node2 automatically.

4. Shut down `openais`:

   ```
   node2# chkconfig openais off
   node2# service openais stop
   ```

5. Upgrade the software on node2.
6. Set `cxfs_client` so that it will not restart automatically upon reboot:

   ```
   node2# chkconfig cxfs_client off
   ```

   Note: This step is required because the upgrade in step 5 automatically reset `cxfs_client` so that it will restart upon reboot, no matter what the prior setting.

7. Reboot node2.
8. Turn `openais` back on for node2:

   ```
   node2# chkconfig openais on
   node2# service openais start
   ```

9. Make node2 active again:

   ```
   # crm_standby -U node2 -v false
   ```

10. Move the resource groups from node1 to node2:

    Note: This command implicitly creates a location constraint with a score of `INFINITY` for node2, meaning that the group will remain on node2.

    ```
    node1# crm_resource -r ipalias-group-1 -M -H node2
    node1# crm_resource -r ipalias-group-2 -M -H node2
    ```

11. *(Optional)* Allow the resource groups to run on node2 for a period of time as a test.

12. Repeat steps 3 through 11 above but executed for node1.

13. *(Optional)* Move the appropriate resource groups from node2 back to node1:

    Note: This command implicitly creates a location constraint with a score of `INFINITY` for node1, meaning that the group will remain on node1.

    ```
    # crm_resource -r ipalias-group-1 -M -H node1
    # crm_resource -r ipalias-group-2 -M -H node1
    ```

14. Remove the implicit location constraints, since the cluster is now back to normal operational state:

    ```
    # crm_resource -r ipalias-group-1 -U
    # crm_resource -r ipalias-group-2 -U
    ```

DMF HAE Rolling Upgrade

Do the following for a rolling upgrade in a DMF HAE cluster:

1. Read the *SGI InfiniteStorage Software Platform* release note, the release notes for the products you intend to upgrade, and any late-breaking caveats on Supportfolio to see if there are any changes to this procedure.

2. Ensure that the resource groups are running on node1:

    ```
    # crm_resource -r dmfGroup -M -H node1
    ```

3. Remove node2 from the HAE cluster and CXFS cluster (if present) by turning the HAE and CXFS services off:

Note: It is possible that the node may be reset and reboot during this process.

```
node2# chkconfig openais off
node2# chkconfig cxfs off
node2# chkconfig cxfs_cluster off
node2# service openais stop
node2# service cxfs stop
node2# service cxfs_cluster stop
```

4. Upgrade the software on node2 and reboot.
5. Add node2 back into the CXFS cluster (if present) by turning the CXFS services on and restarting them:

```
node2# chkconfig cxfs_cluster on
node2# chkconfig cxfs on
node2# service cxfs_cluster start
node2# service cxfs start
```

6. Verify that node2 is fully back in the CXFS cluster with filesystems mounted.
7. Turn the HAE `openais` service on and restart it:

```
node2# chkconfig openais on
node2# service openais start
```

8. Move the resource groups from node1 to node2:

Note: This command implicitly creates a location constraint with a score of `INFINITY` for node2, meaning that the group will remain on node2.

```
# crm_resource -r dmfGroup -M -H node2
```

9. *(Optional)* Allow the resource groups to run on node2 for a period of time as a test.
10. Repeat steps 3 through 9 above but executed for node1.

11. *(Optional)* Move the appropriate resource groups from node2 back to node1:

Note: This command implicitly creates a location constraint with a score of `INFINITY` for node1, meaning that the group will remain on node1.

```
# crm_resource -r dmfGroup -M -H node1
```

12. Remove the implicit location constraints, since the cluster is now back to normal operational state:

```
# crm_resource -r dmfGroup -U
```

Troubleshooting

This section discusses the following:

For details about troubleshooting High Availability Extension (HAE), see the Novell *High Availability Guide*.

General HAE Troubleshooting

If you notice problems with HAE, do the following:

1. Watch the output from `crm_mon`.
2. If things do not seem to be responding correctly or if `crm_mon` lists an error, execute the following:

   ```
   # crm_verify  -LV
   ```

Note: You can run `crm_verify` with a multiple `-V` command-line arguments (such as `-VVV`) for more detail. If you run `crm_verify` before STONITH is enabled, you will see errors. Errors similar to the following may be ignored at this time and will go away after STONITH is configured (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
```

3. If there are any problems listed in the `crm_verify` output, they will contain the failed action and the host on which the action failed. To find the specific problem, try to match those events to messages in `/var/log/messages` or to other information you have about the cluster state.

Recovering from an Incomplete Failover

After an incomplete failover, in which one or more of the resource primitives are not started and the cluster can no longer provide high availability, you must do the following to restore functionality and high availability:

1. Disable HAE management from the resource group:

   # **crm_resource -r *resourceGROUP* -m -p is-managed -v false**

   Note: You must perform this action on the resource group, not on the individual resource primitives.

2. Determine which resource primitives have failcounts:

   # **crm_failcount -G -U *node* -r *resourcePRIMITIVE***

   Repeat for each resource primitive on each node.

3. Troubleshoot the failed resource operations. Examine the `/var/log/messages` system log and application logs around the time of the operation failures in order to deduce why they failed and deal with those causes.

4. Ensure that all of the individual resources are working properly according to the information in Chapter 7, "DMF HA Cluster Resource Examples" on page 59.

5. Remove the failcounts found in step 2:

   # **crm_failcount -D -U *node* -r *failed_resourcePRIMITIVE***

   Repeat this for each failed resource primitive on each node.

6. Remove error messages:

   # **crm_resource -C -H *node* -r *failed_resourcePRIMITIVE***

   Repeat this for each failed resource primitive on each node.

7. Reenable HAE management for the resource group:

   # **crm_resource -r *resourceGROUP* -m -d is-managed**

Clearing the Failcounts After a Severe Error

Under certain circumstances, a severe failure will cause the failcount for the resource primitives to be set to `INFINITY`. This means that the resource primitives cannot run on a specific node again until the failcount is cleared, which requires administrative action. See "Clearing the Resource Primitive Failcount" on page 120.

Error Messages in `/var/log/messages`

If you see errors in the `/var/log/messages` file, you should run the `crm_verify` command and look for corresponding messages.

`dmaudit` Error Detection

Files in the DMF-managed user filesystems that are out of synchronization with DMF database entries because of a system crash may cause errors to be found by `dmaudit`.

For example, when a user file is removed, the DMF daemon will immediately soft-delete the database entries corresponding to that file indicating at what time the file was removed. Should the machine crash shortly thereafter, the removal of the file within the filesystem might not have yet been updated on disk, and so the file will reappear when the filesystem is mounted on the alternate node. If `dmaudit` is then run, it will report that the filesystem and database are inconsistent because it found an existing file pointing at a soft-deleted database entry. You can minimize these

errors by using the `dirsync` mount option if potentially lower filesystem performance is not a concern.

DMF-managed user filesystems may be mounted with either the `sync` or `dirsync` mount option, depending on the desire for filesystem completeness upon system failure. Alternatively, the user applications accessing these filesystems are responsible for verifying file synchronization.

Caution: The `sync` and `dirsync` mount options have serious performance implications that may outweigh filesystem synchronization benefits.

DMF Logs are Incomplete

DMF logs may not be complete due to system failures. If you want to ensure that the DMF logs are absolutely complete upon system failure, despite any performance issues, you may optionally choose to mount the DMF spool filesystem with the `sync` mount option.

The DMF home, journals, and disk MSP filesystems do not require the `sync` mount option because DMF synchronizes data on those filesystems. The move and temporary filesystems do not require the `sync` mount option because DMF does not recover anything from them.

Caution: The `sync` mount option has serious performance implications that may outweigh filesystem synchronization benefits.

Using SGI Knowledgebase

If you encounter problems and have an SGI support contract, you can log on to Supportfolio and access the Knowledgebase tool to help find answers.

To log in to Supportfolio Online, see:

https://support.sgi.com/login

Then click on **Search the SGI Knowledgebase** and select the type of search you want to perform.

If you need further assistance, contact SGI Support.

Reporting Problems to SGI

If you need to report problems to SGI Support, do the following:

- Run the following command as `root` on every node in the cluster in order to gather system configuration information:

  # **`/usr/sbin/system_info_gather -A -o node.out`**

- Collect HAE cluster information by using the `hb_report` command:

`hb_report -f priortime destination_directory`

  where:

  – *priortime* specifies some time prior to when the problem began (specify *priortime* in `Date::Parse` Perl module format)

  – *destination_directory* is the absolute pathname of a nonexistent directory that will be created as a compressed `bunzip2` tarball in the format:

  destination_directory`.tar.bz2`

  For example, if run on June 2 2010 at 3:06 PM, the following will create a report starting from 1:00 am that day and place the output in `/tmp/hb_report.20100602-1506.tar.bz2`:

`hb_report -f 1am /tmp/hb_report.$(date +%Y%m%d-%H%M)`

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

- 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.

- Collect any other system log files that may contain information about HAE or the services included in the HA configuration (if not otherwise gathered by the above tools).

When you contact SGI Support, you will be provided with information on how and where to upload the collected information files for SGI analysis.

Appendix A

Differences Among FailSafe®, Heartbeat, and HAE

Table A-1 summarizes the differences among the following, for those readers who may be familiar with with the older products:

- FailSafe®
- Linux-HA Heartbeat
- SUSE Linux Enterprise High Availability Extension (HAE)

Note: These products do not work together and cannot form an HA cluster.

Table A-1 Differences Among FailSafe, Heartbeat, and HAE

Topic	FailSafe	Heartbeat	HAE
Operating system	IRIX	Can be built and run on most operating systems based on UNIX. The version of Heartbeat packaged by SGI is part of the ISSP media distribution and runs on the base OS for ISSP as defined in the ISSP release notes.	SLES 11
Terminology	node resource	node resource	node resource
Size of cluster	8 nodes	8+ nodes (Specific resource agents may have cluster size limitations. DMF can run on only 2 nodes in active/passive mode.)	16 nodes in active/passive mode for DMF, but 2 nodes recommended. 2 nodes for CXFS NFS edge-serving in active/active mode.
Node/member name	Hostname or private network address	Hostname and private network address	Hostname and private network address

Topic	FailSafe	Heartbeat	HAE
NFS lock failover	Supported	Not supported by the operating system	Supported in active/passive configurations
Network tiebreaker	A node that is participating in the cluster membership. FailSafe tries to include the tiebreaker node in the membership in case of a split cluster.	You can configure Heartbeat to use a variety of methods to provide tiebreaker functionality.	You can configure HAE to use a variety of methods to provide tiebreaker functionality.
Rolling upgrade	Supported	Supported	Supported
Configuration information storage	Information is stored in the cluster database. The cluster database is replicated on all nodes automatically and kept in synchronization.	The `/etc/ha.d/ha.cf` file contains bootstrap information and must be manually replicated across the cluster when changed. Other cluster configuration is stored in the cluster information base (CIB), which is a replicated database. You can use `cibadmin`(8) to query and update the CIB.	The `/etc/corosync/corosync.conf` file contains bootstrap information. Other cluster configuration is stored in the replicated CIB. You can use `cibadmin`(8) to query and update the CIB.
Making changes while the service is enabled	Depends upon the plug-in and the configuration device parameter.	Service parameters can be changed while a service is running. Depending on the service and parameter, a change may cause a `stop`/`start` or a trigger a `restart` action. SGI recommends that you do not make any changes that could stop or restart DMF and CXFS.	Service parameters can be changed while a service is running. Depending on the service and parameter, a change may cause a `stop`/`start` or a trigger a `restart` action. SGI recommends that you do not make any changes that could stop or restart DMF and CXFS.
Heartbeat interval and timeout	You can specify cluster membership heartbeat interval and timeout (in milliseconds).	Heartbeat provides a number of parameters to tune node status monitoring and failure actions.	HAE provides a number of parameters to tune node status monitoring and failure actions.

Topic	FailSafe	Heartbeat	HAE
Heartbeat networks	Allows multiple networks to be designated as heartbeat networks. You can choose a list of networks.	You can configure Heartbeat to communicate over one or more private or public networks.	You can configure HAE to communicate over one or more private or public networks.
Action scripts	Separate scripts named `start`, `stop`, `monitor`, `restart`, `exclusive`.	Open Cluster Framework (OCF) resource agent specification, which may support `start`, `monitor`, `stop`, and `restart` actions as well as other more-specialized actions.	OCF resource agent specification, which may support `start`, `monitor`, `stop`, and `restart` actions as well as other more-specialized actions.
Resource timeouts	Timeouts can be specified for each action (`start`, `stop`, `monitor`, `restart`, `exclusive`) and for each resource type independently.	Timeouts and failover actions are highly configurable.	Timeouts and failover actions are highly configurable.
Resource dependencies	Resource and resource type dependencies are supported and can be modified by the user.	Heartbeat provides great flexibility to configure resource dependencies.	HAE provides great flexibility to configure resource dependencies.
Failover policies	The ordered and round-robin failover policies are predefined. User-defined failover policies are supported.	Heartbeat provides great flexibility to configure resource failover policies.	HAE provides great flexibility to configure resource failover policies.

Glossary

This glossary lists terms and abbreviations used within this guide. For a more information, see the Novell *High Availability Guide*:

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

active/active mode

An HAE cluster in which multiple nodes are able to run disjoint sets of resources, with each node serving as a backup for another node's resources in case of node failure.

active/passive mode

An HAE cluster in which all of the resources run on one node and one or more other nodes are the standby in case the first node fails.

BMC

Baseboard management controller, a system controller used in resetting x86-64 systems, such as SGI Altix XE.

CIB

Cluster information base, used to define the HAE cluster.

clone

A resource that is active on more than one node.

CXFS

Clustered XFS.

CXFS NFS edge-serving

A configuration in which CXFS client nodes can export data with NFS.

DCP

Drive control program.

DMF

Data Migration Facility, a hierarchical storage management system for SGI environments.

DMF Manager

A web-based tool you can use to deal with day-to-day DMF operational issues and focus on work flow.

edge-serving

See *CXFS NFS edge-serving*.

fencing

The method that HAE uses to guarantee a known cluster state when communication to a node fails or actions on a node fail. (This differs from the concept of *fencing* in CXFS.)

HA

Highly available or *high availability*, in which resources fail over from one node to another without disrupting services for clients.

HAE fail policy

A parameter defined in the CIB that determines what happens when a resource fails.

HAE-managed filesystem

A filesystem that will be made highly available according to the instructions in this guide.

High Availability Extension (HAE)

Novell SUSE Linux Enterprise product for high availability.

IPMI

Intelligent Platform Management Interface, a system reset method for x86-64 systems, such as SGI Altix XE.

ISSP

InfiniteStorage Software Platform, an SGI software distribution.

LCP

library control program.

LSB

Linux Standard Base.

node1

In the examples in this guide, the initial host (which will later become a node in the HAE cluster) on which all of the filesystems will be mounted and on which all tape drives and libraries are accessible. See also *alternate node*.

NSM

Network Status Monitor.

node2

In the examples in this guide, the alternate host in the HAE cluster other than the first node (node1). See also *node1*.

OCF

Open Cluster Framework.

OpenVault

A tape mounting service used by DMF.

physvol

XVM physical volume.

primitive

Used to define a resource in the CIB.

resource

A service, associated with an IP address, that is managed by HAE.

resource agent

The software that allows a service 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

A concept in HAE 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*.

STONITH

Shoot the other node in the head, the facility that guarantees cluster state by fencing non-responsive or failing nodes.

TMF

Tape Management Facility, a tape mounting service used by DMF.

XFS

A filesystem implementation type for the Linux operating system. It defines the format that is used to store data on disks managed by the filesystem.

WSDL

Web Service Definition Language

XVM

Volume manager for XFS filesystems (local XVM).

Index

P

R

S

T

U

V

W

X

Y