Red Hat Enterprise Linux 6

Cluster Administration

Configuring and Managing the High Availability Add-On

Red Hat Enterprise Linux 6 Cluster Administration
Configuring and Managing the High Availability Add-On
Edition 0

1801 Varsity Drive
Raleigh, NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701

Configuring and Managing the High Availability Add-On describes the configuration and management of the High Availability Add-On for Red Hat Enterprise Linux 6.

Introduction **vii**
1. Document Conventions ... viii
1.1. Typographic Conventions ... viii
1.2. Pull-quote Conventions ... ix
1.3. Notes and Warnings ... x
2. Feedback ... x

1. Red Hat High Availability Add-On Configuration and Management Overview **1**
1.1. New and Changed Features ... 1
1.1.1. New and Changed Features for Red Hat Enterprise Linux 6.1 ... 1
1.1.2. New and Changed Features for Red Hat Enterprise Linux 6.2 ... 2
1.2. Configuration Basics ... 3
1.3. Setting Up Hardware ... 3
1.4. Installing Red Hat High Availability Add-On software ... 4
1.5. Configuring Red Hat High Availability Add-On Software ... 5

2. Before Configuring the Red Hat High Availability Add-On **7**
2.1. General Configuration Considerations ... 7
2.2. Compatible Hardware ... 8
2.3. Enabling IP Ports ... 9
2.3.1. Enabling IP Ports on Cluster Nodes ... 9
2.3.2. Enabling the IP Port for **luci** ... 9
2.4. Configuring **luci** with **`/etc/sysconfig/luci`** ... 10
2.5. Configuring ACPI For Use with Integrated Fence Devices ... 11
2.5.1. Disabling ACPI Soft-Off with **`chkconfig`** Management ... 12
2.5.2. Disabling ACPI Soft-Off with the BIOS ... 13
2.5.3. Disabling ACPI Completely in the **`grub.conf`** File ... 14
2.6. Considerations for Configuring HA Services ... 16
2.7. Configuration Validation ... 18
2.8. Considerations for **NetworkManager** ... 21
2.9. Considerations for Using Quorum Disk ... 21
2.10. Red Hat High Availability Add-On and SELinux ... 22
2.11. Multicast Addresses ... 22
2.12. Configuring the iptables Firewall to Allow Cluster Components ... 23
2.13. UDP Unicast Traffic ... 24
2.14. Considerations for **`ricci`** ... 24

3. Configuring Red Hat High Availability Add-On With Conga **25**
3.1. Configuration Tasks ... 25
3.2. Starting **luci** ... 26
3.3. Controlling Access to luci ... 27
3.4. Creating a Cluster ... 28
3.5. Global Cluster Properties ... 31
3.5.1. Configuring General Properties ... 31
3.5.2. Configuring Fence Daemon Properties ... 32
3.5.3. Network Configuration ... 32
3.5.4. Quorum Disk Configuration ... 33
3.5.5. Logging Configuration ... 34
3.6. Configuring Fence Devices ... 35
3.6.1. Creating a Fence Device ... 36
3.6.2. Modifying a Fence Device ... 36
3.6.3. Deleting a Fence Device ... 37
3.7. Configuring Fencing for Cluster Members ... 37
3.7.1. Configuring a Single Fence Device for a Node ... 37
3.7.2. Configuring a Backup Fence Device ... 38

3.7.3. Configuring a Node with Redundant Power 39
3.8. Configuring a Failover Domain 40
3.8.1. Adding a Failover Domain 42
3.8.2. Modifying a Failover Domain 43
3.8.3. Deleting a Failover Domain 43
3.9. Configuring Global Cluster Resources 43
3.10. Adding a Cluster Service to the Cluster 44

4. Managing Red Hat High Availability Add-On With Conga 47
4.1. Adding an Existing Cluster to the luci Interface 47
4.2. Managing Cluster Nodes 47
4.2.1. Rebooting a Cluster Node 47
4.2.2. Causing a Node to Leave or Join a Cluster 48
4.2.3. Adding a Member to a Running Cluster 48
4.2.4. Deleting a Member from a Cluster 49
4.3. Starting, Stopping, Restarting, and Deleting Clusters 50
4.4. Managing High-Availability Services 51

5. Configuring Red Hat High Availability Add-On With the ccs Command 53
5.1. Operational Overview 54
5.1.1. Creating the Cluster Configuration File on a Local System 54
5.1.2. Viewing the Current Cluster Configuration 55
5.1.3. Specifying ricci Passwords with the ccs Command 55
5.1.4. Modifying Cluster Configuration Components 55
5.2. Configuration Tasks 55
5.3. Starting **ricci** 56
5.4. Creating A Cluster 56
5.5. Configuring Fence Devices 58
5.6. Listing Fence Devices and Fence Device Options 60
5.7. Configuring Fencing for Cluster Members 61
5.7.1. Configuring a Single Power-Based Fence Device for a Node 61
5.7.2. Configuring a Single Storage-Based Fence Device for a Node 63
5.7.3. Configuring a Backup Fence Device 65
5.7.4. Configuring a Node with Redundant Power 68
5.7.5. Removing Fence Methods and Fence Instances 71
5.8. Configuring a Failover Domain 71
5.9. Configuring Global Cluster Resources 74
5.10. Adding a Cluster Service to the Cluster 75
5.11. Listing Available Cluster Services 77
5.12. Virtual Machine Resources 78
5.13. Configuring a Quorum Disk 79
5.14. Miscellaneous Cluster Configuration 80
5.14.1. Cluster Configuration Version 81
5.14.2. Multicast Configuration 81
5.14.3. Configuring a Two-Node Cluster 82
5.15. Propagating the Configuration File to the Cluster Nodes 82

6. Managing Red Hat High Availability Add-On With ccs 85
6.1. Managing Cluster Nodes 85
6.1.1. Causing a Node to Leave or Join a Cluster 85
6.1.2. Adding a Member to a Running Cluster 85
6.2. Starting and Stopping a Cluster 86
6.3. Diagnosing and Correcting Problems in a Cluster 86

7. Configuring Red Hat High Availability Add-On With Command Line Tools 87
7.1. Configuration Tasks 88

7.2. Creating a Basic Cluster Configuration File 88
7.3. Configuring Fencing 92
7.4. Configuring Failover Domains 97
7.5. Configuring HA Services 100
7.5.1. Adding Cluster Resources 101
7.5.2. Adding a Cluster Service to the Cluster 103
7.6. Verifying a Configuration 105

8. Managing Red Hat High Availability Add-On With Command Line Tools 109
8.1. Starting and Stopping the Cluster Software 110
8.1.1. Starting Cluster Software 110
8.1.2. Stopping Cluster Software 110
8.2. Deleting or Adding a Node 111
8.2.1. Deleting a Node from a Cluster 111
8.2.2. Adding a Node to a Cluster 114
8.2.3. Examples of Three-Node and Two-Node Configurations 117
8.3. Managing High-Availability Services 119
8.3.1. Displaying HA Service Status with **clustat** 119
8.3.2. Managing HA Services with **clusvcadm** 120
8.4. Updating a Configuration 122
8.4.1. Updating a Configuration Using **cman_tool version -r** 123
8.4.2. Updating a Configuration Using **scp** 125

9. Diagnosing and Correcting Problems in a Cluster 127
9.1. Cluster Does Not Form 127
9.2. Nodes Unable to Rejoin Cluster after Fence or Reboot 127
9.3. Cluster Services Hang 128
9.4. Cluster Service Will Not Start 128
9.5. Cluster-Controlled Services Fails to Migrate 129
9.6. Each Node in a Two-Node Cluster Reports Second Node Down 129
9.7. Nodes are Fenced on LUN Path Failure 129
9.8. Quorum Disk Does Not Appear as Cluster Member 129
9.9. Unusual Failover Behavior 129
9.10. Fencing Occurs at Random 130

10. SNMP Configuration with the Red Hat High Availability Add-On 131
10.1. SNMP and the Red Hat High Availability Add-On 131
10.2. Configuring SNMP with the Red Hat High Availability Add-On 131
10.3. Forwarding SNMP traps 132
10.4. SNMP Traps Produced by Red Hat High Availability Add-On 132

11. Clustered Samba Configuration 135
11.1. CTDB Overview 135
11.2. Required Packages 135
11.3. GFS2 Configuration 135
11.4. CTDB Configuration 137
11.5. Samba Configuration 139
11.6. Starting CTDB and Samba Services 140
11.7. Using the Clustered Samba Server 140

A. Fence Device Parameters 143

B. HA Resource Parameters 153

C. HA Resource Behavior 163
C.1. Parent, Child, and Sibling Relationships Among Resources 164
C.2. Sibling Start Ordering and Resource Child Ordering 164

C.2.1. Typed Child Resource Start and Stop Ordering 165
C.2.2. Non-typed Child Resource Start and Stop Ordering 167
C.3. Inheritance, the <resources> Block, and Reusing Resources 169
C.4. Failure Recovery and Independent Subtrees 170
C.5. Debugging and Testing Services and Resource Ordering 171

D. Cluster Service Resource Check and Failover Timeout 173
D.1. Modifying the Resource Status Check Interval 173
D.2. Enforcing Resource Timeouts 174

E. Command Line Tools Summary 175

F. Revision History 177

Index 181

Introduction

This document provides information about installing, configuring and managing Red Hat High Availability Add-On components. Red Hat High Availability Add-On components allow you to connect a group of computers (called *nodes* or *members*) to work together as a cluster. In this document, the use of the word *cluster* or *clusters* is used to refer to a group of computers running the Red Hat High Availability Add-On.

The audience of this document should have advanced working knowledge of Red Hat Enterprise Linux and understand the concepts of clusters, storage, and server computing.

This document is organized as follows:

- *Chapter 1, Red Hat High Availability Add-On Configuration and Management Overview*
- *Chapter 2, Before Configuring the Red Hat High Availability Add-On*
- *Chapter 3, Configuring Red Hat High Availability Add-On With **Conga***
- *Chapter 4, Managing Red Hat High Availability Add-On With **Conga***
- *Chapter 7, Configuring Red Hat High Availability Add-On With Command Line Tools*
- *Chapter 8, Managing Red Hat High Availability Add-On With Command Line Tools*
- *Chapter 9, Diagnosing and Correcting Problems in a Cluster*
- *Chapter 10, SNMP Configuration with the Red Hat High Availability Add-On*
- *Appendix A, Fence Device Parameters*
- *Appendix B, HA Resource Parameters*
- *Appendix C, HA Resource Behavior*
- *Appendix E, Command Line Tools Summary*
- *Appendix F, Revision History*

For more information about Red Hat Enterprise Linux 6, refer to the following resources:

- *Red Hat Enterprise Linux Installation Guide* — Provides information regarding installation of Red Hat Enterprise Linux 6.
- *Red Hat Enterprise Linux Deployment Guide* — Provides information regarding the deployment, configuration and administration of Red Hat Enterprise Linux 6.

For more information about the High Availability Add-On and related products for Red Hat Enterprise Linux 6, refer to the following resources:

- *High Availability Add-On Overview* — Provides a high-level overview of the Red Hat High Availability Add-On.
- *Logical Volume Manager Administration* — Provides a description of the Logical Volume Manager (LVM), including information on running LVM in a clustered environment.
- *Global File System 2: Configuration and Administration* — Provides information about installing, configuring, and maintaining Red Hat GFS2 (Red Hat Global File System 2), which is included in the Resilient Storage Add-On.

- *DM Multipath* — Provides information about using the Device-Mapper Multipath feature of Red Hat Enterprise Linux 6.
- *Load Balancer Administration* — Provides information on configuring high-performance systems and services with the Load Balancer Add-On, a set of integrated software components that provide Linux Virtual Servers (LVS) for balancing IP load across a set of real servers.
- *Release Notes* — Provides information about the current release of Red Hat products.

High Availability Add-On documentation and other Red Hat documents are available in HTML, PDF, and RPM versions on the Red Hat Enterprise Linux Documentation CD and online at *http://docs.redhat.com/docs/en-US/index.html*.

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[1] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keycaps and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to the first virtual terminal. Press **`Ctrl`**+**`Alt`**+**`F1`** to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

[1] https://fedorahosted.org/liberation-fonts/

File-related classes include **filesystem** for file systems, **file** for files, and **dir** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).

To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic or ***Proportional Bold Italic***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

To connect to a remote machine using ssh, type **ssh *username*@*domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.

The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.

To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books         Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests   Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **`mono-spaced roman`** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. Feedback

If you spot a typo, or if you have thought of a way to make this manual better, we would love to hear from you. Please submit a report in Bugzilla (*http://bugzilla.redhat.com/bugzilla/*) against the component **doc-Cluster_Administration**.

Be sure to mention the manual identifier:

```
Cluster_Administration(EN)-6 (2011-11-14T16:26)
```

By mentioning this manual's identifier, we know exactly which version of the guide you have.

If you have a suggestion for improving the documentation, try to be as specific as possible. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Red Hat High Availability Add-On Configuration and Management Overview

Red Hat High Availability Add-On allows you to connect a group of computers (called *nodes* or *members*) to work together as a cluster. You can use Red Hat High Availability Add-On to suit your clustering needs (for example, setting up a cluster for sharing files on a GFS2 file system or setting up service failover).

Note

For information on best practices for deploying and upgrading Red Hat Enterprise Linux clusters using the High Availability Add-On and Red Hat Global File System 2 (GFS2) refer to the article "Red Hat Enterprise Linux Cluster, High Availability, and GFS Deployment Best Practices" on Red Hat Customer Portal at . *https://access.redhat.com/kb/docs/DOC-40821*[1].

This chapter provides a summary of documentation features and updates that have been added to the Red Hat High Availability Add-On since the initial release of Red Hat Enterprise Linux 6, followed by an overview of configuring and managing the Red Hat High Availability Add-On.

1.1. New and Changed Features

This section lists new and changed features of the Red Hat High Availability Add-On documentation that have been added since the initial release of Red Hat Enterprise Linux 6.

1.1.1. New and Changed Features for Red Hat Enterprise Linux 6.1

Red Hat Enterprise Linux 6.1 includes the following documentation and feature updates and changes.

- As of the Red Hat Enterprise Linux 6.1 release and later, the Red Hat High Availability Add-On provides support for SNMP traps. For information on configuring SNMP traps with the Red Hat High Availability Add-On, refer to *Chapter 10, SNMP Configuration with the Red Hat High Availability Add-On*.
- As of the Red Hat Enterprise Linux 6.1 release and later, the Red Hat High Availability Add-On provides support for the **ccs** cluster configuration command. For information on the **ccs** command, refer to *Chapter 5, Configuring Red Hat High Availability Add-On With the **ccs** Command* and *Chapter 6, Managing Red Hat High Availability Add-On With **ccs***.
- The documentation for configuring and managing Red Hat High Availability Add-On software using Conga has been updated to reflect updated Conga screens and feature support.

[1] https://access.redhat.com/kb/docs/DOC-40821

- For the Red Hat Enterprise Linux 6.1 release and later, using **ricci** requires a password the first time you propagate updated cluster configuration from any particular node. For information on **ricci** refer to *Section 2.14, "Considerations for **ricci**"*.
- You can now specify a *Restart-Disable* failure policy for a service, indicating that the system should attempt to restart the service in place if it fails, but if restarting the service fails the service will be disabled instead of being moved to another host in the cluster. This feature is documented in *Section 3.10, "Adding a Cluster Service to the Cluster"* and *Appendix B, HA Resource Parameters*.
- You can now configure an independent subtree as non-critical, indicating that if the resource fails then only that resource is disabled. For information on this feature see *Section 3.10, "Adding a Cluster Service to the Cluster"* and *Section C.4, "Failure Recovery and Independent Subtrees"*.
- This document now includes the new chapter *Chapter 9, Diagnosing and Correcting Problems in a Cluster*.

In addition, small corrections and clarifications have been made throughout the document.

1.1.2. New and Changed Features for Red Hat Enterprise Linux 6.2

Red Hat Enterprise Linux 6.2 includes the following documentation and feature updates and changes.

- Red Hat Enterprise Linux now provides support for running Clustered Samba in an active/active configuration. For information on clustered Samba configuration, refer to *Chapter 11, Clustered Samba Configuration*.
- Although any user able to authenticate on the system that is hosting **luci** can log in to **luci**, as of Red Hat Enterprise Linux 6.2 only the root user on the system that is running **luci** can access any of the **luci** components until an administrator (the root user or a user with administrator permission) sets permissions for that user. For information on setting **luci** permissions for users, refer to *Section 3.3, "Controlling Access to luci"*.
- Red Hat High-Availability Add-On nodes can communicate with each other using the UDP unicast transport mechanism. For information on configuring UDP unicast, refer to *Section 2.13, "UDP Unicast Traffic"*.
- You can now configure some aspects of **luci**'s behavior by means of the **/etc/sysconfig/luci** file. For example, you can specifically configure the only IP address **luci** is being served at. For information on configuring the only IP address **luci** is being served at, refer to *Table 2.2, "Enabled IP Port on a Computer That Runs **luci**"*. For information on the **/etc/sysconfig/luci** file in general, refer to *Section 2.4, "Configuring **luci** with **/etc/sysconfig/luci**"*.
- The **ccs** command now includes the **--lsfenceopts** option, which prints a list of available fence devices, and the **--lsfenceopts** *fence_type* option, which prints each available fence type. For information on these options, refer to *Section 5.6, "Listing Fence Devices and Fence Device Options"*.
- The **ccs** command now includes the **--lsserviceopts** option, which prints a list of cluster services currently available for your cluster, and the **--lsserviceopts** *service_type* option, which prints a list of the options you can specify for a particular service type. For information on these options, refer to *Section 5.11, "Listing Available Cluster Services"*.
- The Red Hat Enterprise Linux 6.2 release provides support for the VMware (SOAP Interface) fence agent. For information on fence device parameters, refer to *Appendix A, Fence Device Parameters*.

- The Red Hat Enterprise Linux 6.2 release provides support for the RHEV-M REST API fence agent, against RHEV 3.0 and later. For information on fence device parameters, refer to *Appendix A, Fence Device Parameters*.
- As of the Red Hat Enterprise Linux 6.2 release, when you configure a virtual machine in a cluster with the **ccs** command you can use the **--addvm** option (rather than the **addservice** option). This ensures that the **vm** resource is defined directly under the **rm** configuration node in the cluster configuration file. For information on configuring virtual machine resources with the **ccs** command, refer to *Section 5.12, "Virtual Machine Resources"*.
- This document includes a new appendix, *Appendix D, Cluster Service Resource Check and Failover Timeout*. This appendix describes how **rgmanager** monitors the status of cluster resources, and how to modify the status check interval. The appendix also describes the **__enforce_timeouts** service parameter, which indicates that a timeout for an operation should cause a service to fail.
- This document includes a new section, *Section 2.12, "Configuring the iptables Firewall to Allow Cluster Components"*. This section shows the filtering you can use to allow multicast traffic through the **iptables** firewall for the various cluster components.

In addition, small corrections and clarifications have been made throughout the document.

1.2. Configuration Basics

To set up a cluster, you must connect the nodes to certain cluster hardware and configure the nodes into the cluster environment. Configuring and managing the Red Hat High Availability Add-On consists of the following basic steps:

1. Setting up hardware. Refer to *Section 1.3, "Setting Up Hardware"*.
2. Installing Red Hat High Availability Add-On software. Refer to *Section 1.4, "Installing Red Hat High Availability Add-On software"*.
3. Configuring Red Hat High Availability Add-On Software. Refer to *Section 1.5, "Configuring Red Hat High Availability Add-On Software"*.

1.3. Setting Up Hardware

Setting up hardware consists of connecting cluster nodes to other hardware required to run the Red Hat High Availability Add-On. The amount and type of hardware varies according to the purpose and availability requirements of the cluster. Typically, an enterprise-level cluster requires the following type of hardware (refer to *Figure 1.1, "Red Hat High Availability Add-On Hardware Overview"*). For considerations about hardware and other cluster configuration concerns, refer to *Chapter 2, Before Configuring the Red Hat High Availability Add-On* or check with an authorized Red Hat representative.

- High Availability Add-On nodes — Computers that are capable of running Red Hat Enterprise Linux 6 software, with at least 1GB of RAM.
- Ethernet switch or hub for public network — This is required for client access to the cluster.
- Ethernet switch or hub for private network — This is required for communication among the cluster nodes and other cluster hardware such as network power switches and Fibre Channel switches.
- Network power switch — A network power switch is recommended to perform fencing in an enterprise-level cluster.

- Fibre Channel switch — A Fibre Channel switch provides access to Fibre Channel storage. Other options are available for storage according to the type of storage interface; for example, iSCSI. A Fibre Channel switch can be configured to perform fencing.
- Storage — Some type of storage is required for a cluster. The type required depends on the purpose of the cluster.

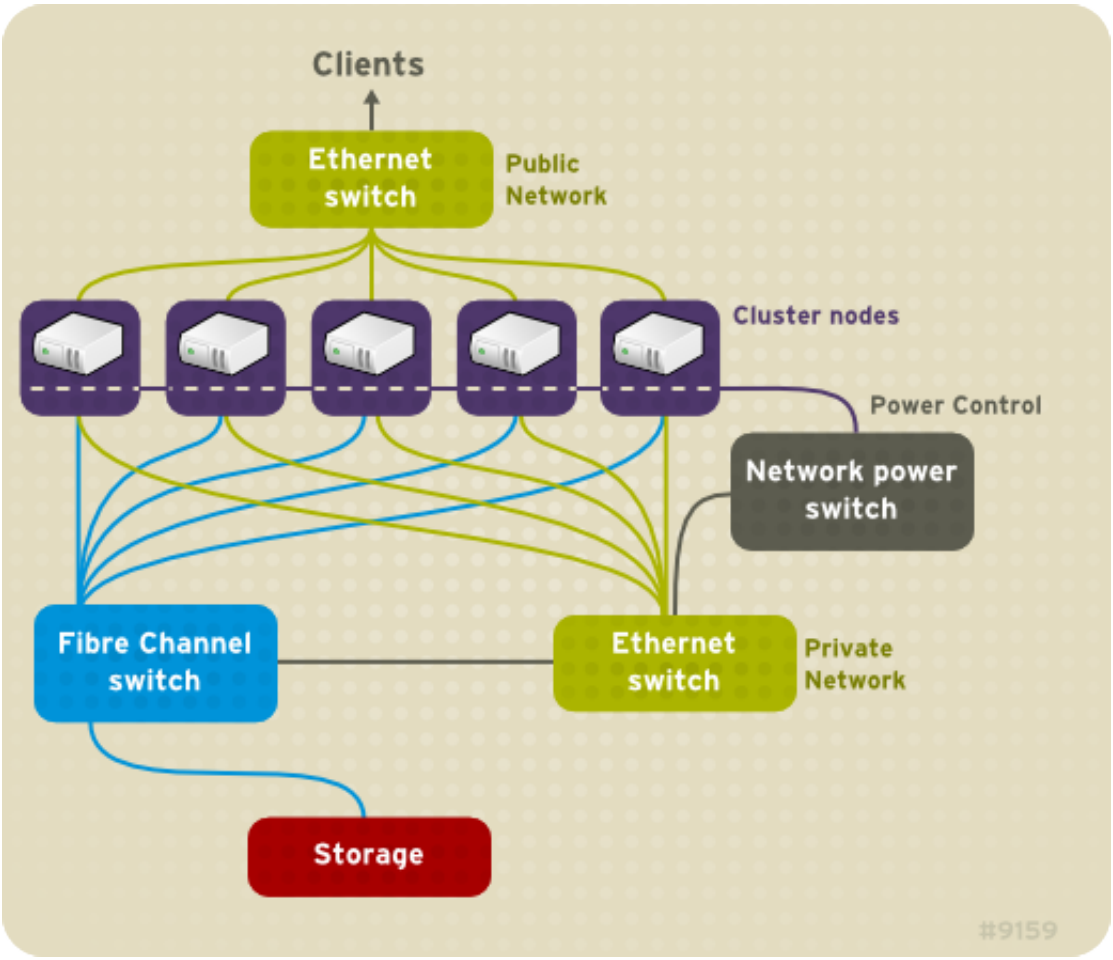

Figure 1.1. Red Hat High Availability Add-On Hardware Overview

1.4. Installing Red Hat High Availability Add-On software

To install Red Hat High Availability Add-On software, you must have entitlements for the software. If you are using the **Conga** configuration GUI, you can let it install the cluster software. If you are using other tools to configure the cluster, secure and install the software as you would with Red Hat Enterprise Linux software.

Upgrading Red Hat High Availability Add-On Software

It is possible to upgrade the cluster software on a given major release of Red Hat Enterprise Linux without taking the cluster out of production. Doing so requires disabling the cluster software on one host at a time, upgrading the software, and restarting the cluster software on that host.

1. Shut down all cluster services on a single cluster node. For instructions on stopping cluster software on a node, refer to *Section 8.1.2, "Stopping Cluster Software"*. It may be desirable to manually relocate cluster-managed services and virtual machines off of the host prior to stopping **`rgmanager`**.

2. Execute the **yum update** command to install the new RPMs. For example:

```
# yum update -y openais cman rgmanager lvm2-cluster gfs2-utils
```

3. Reboot the cluster node or restart the cluster services manually. For instructions on starting cluster software on a node, refer to *Section 8.1.1, "Starting Cluster Software"*.

1.5. Configuring Red Hat High Availability Add-On Software

Configuring Red Hat High Availability Add-On software consists of using configuration tools to specify the relationship among the cluster components. The following cluster configuration tools are available with Red Hat High Availability Add-On:

- **Conga** — This is a comprehensive user interface for installing, configuring, and managing Red Hat High Availability Add-On. Refer to *Chapter 3, Configuring Red Hat High Availability Add-On With **Conga*** and *Chapter 4, Managing Red Hat High Availability Add-On With **Conga*** for information about configuring and managing High Availability Add-On with **Conga**.
- The **ccs** command — This command configures and manages Red Hat High Availability Add-On. Refer to *Chapter 5, Configuring Red Hat High Availability Add-On With the **ccs** Command* and *Chapter 6, Managing Red Hat High Availability Add-On With **ccs*** for information about configuring and managing High Availability Add-On with the **ccs** command.
- Command-line tools — This is a set of command-line tools for configuring and managing Red Hat High Availability Add-On. Refer to *Chapter 7, Configuring Red Hat High Availability Add-On With Command Line Tools* and *Chapter 8, Managing Red Hat High Availability Add-On With Command Line Tools* for information about configuring and managing a cluster with command-line tools. Refer to *Appendix E, Command Line Tools Summary* for a summary of preferred command-line tools.

Note

system-config-cluster is not available in Red Hat Enterprise Linux 6.

Before Configuring the Red Hat High Availability Add-On

This chapter describes tasks to perform and considerations to make before installing and configuring the Red Hat High Availability Add-On, and consists of the following sections.

Important

Make sure that your deployment of Red Hat High Availability Add-On meets your needs and can be supported. Consult with an authorized Red Hat representative to verify your configuration prior to deployment. In addition, allow time for a configuration burn-in period to test failure modes.

- *Section 2.1, "General Configuration Considerations"*
- *Section 2.2, "Compatible Hardware"*
- *Section 2.3, "Enabling IP Ports"*
- *Section 2.4, "Configuring **luci** with `/etc/sysconfig/luci`"*
- *Section 2.5, "Configuring ACPI For Use with Integrated Fence Devices"*
- *Section 2.6, "Considerations for Configuring HA Services"*
- *Section 2.7, "Configuration Validation"*
- *Section 2.8, "Considerations for **NetworkManager**"*
- *Section 2.9, "Considerations for Using Quorum Disk"*
- *Section 2.10, "Red Hat High Availability Add-On and SELinux"*
- *Section 2.11, "Multicast Addresses"*
- *Section 2.12, "Configuring the iptables Firewall to Allow Cluster Components"*
- *Section 2.13, "UDP Unicast Traffic"*
- *Section 2.14, "Considerations for `ricci`"*

2.1. General Configuration Considerations

You can configure the Red Hat High Availability Add-On in a variety of ways to suit your needs. Take into account the following general considerations when you plan, configure, and implement your deployment.

Number of cluster nodes supported
: The maximum number of cluster nodes supported by the High Availability Add-On is 16.

Single site clusters
: Only single site clusters are fully supported at this time. Clusters spread across multiple physical locations are not formally supported. For more details and to discuss multi-site clusters, please speak to your Red Hat sales or support representative.

GFS2
: Although a GFS2 file system can be implemented in a standalone system or as part of a cluster configuration, Red Hat does not support the use of GFS2 as a single-node file system. Red Hat does support a number of high-performance single-node file systems that are optimized for single node, and thus have generally lower overhead than a cluster file system. Red Hat recommends using those file systems in preference to GFS2 in cases where only a single node needs to mount the file system. Red Hat will continue to support single-node GFS2 file systems for existing customers.

  When you configure a GFS2 file system as a cluster file system, you must ensure that all nodes in the cluster have access to the shared file system. Asymmetric cluster configurations in which some nodes have access to the file system and others do not are not supported.This does not require that all nodes actually mount the GFS2 file system itself.

No-single-point-of-failure hardware configuration
: Clusters can include a dual-controller RAID array, multiple bonded network channels, multiple paths between cluster members and storage, and redundant un-interruptible power supply (UPS) systems to ensure that no single failure results in application down time or loss of data.

  Alternatively, a low-cost cluster can be set up to provide less availability than a no-single-point-of-failure cluster. For example, you can set up a cluster with a single-controller RAID array and only a single Ethernet channel.

  Certain low-cost alternatives, such as host RAID controllers, software RAID without cluster support, and multi-initiator parallel SCSI configurations are not compatible or appropriate for use as shared cluster storage.

Data integrity assurance
: To ensure data integrity, only one node can run a cluster service and access cluster-service data at a time. The use of power switches in the cluster hardware configuration enables a node to power-cycle another node before restarting that node's HA services during a failover process. This prevents two nodes from simultaneously accessing the same data and corrupting it. *Fence devices* (hardware or software solutions that remotely power, shutdown, and reboot cluster nodes) are used to guarantee data integrity under all failure conditions.

Ethernet channel bonding
: Cluster quorum and node health is determined by communication of messages among cluster nodes via Ethernet. In addition, cluster nodes use Ethernet for a variety of other critical cluster functions (for example, fencing). With Ethernet channel bonding, multiple Ethernet interfaces are configured to behave as one, reducing the risk of a single-point-of-failure in the typical switched Ethernet connection among cluster nodes and other cluster hardware.

IPv4 and IPv6
: The High Availability Add-On supports both IPv4 and IPv6 Internet Protocols. Support of IPv6 in the High Availability Add-On is new for Red Hat Enterprise Linux 6.

2.2. Compatible Hardware

Before configuring Red Hat High Availability Add-On software, make sure that your cluster uses appropriate hardware (for example, supported fence devices, storage devices, and Fibre Channel switches). Refer to the hardware configuration guidelines at *http://www.redhat.com/cluster_suite/hardware/* for the most current hardware compatibility information.

2.3. Enabling IP Ports

Before deploying the Red Hat High Availability Add-On, you must enable certain IP ports on the cluster nodes and on computers that run **luci** (the **Conga** user interface server). The following sections identify the IP ports to be enabled:

- *Section 2.3.1, "Enabling IP Ports on Cluster Nodes"*
- *Section 2.3.2, "Enabling the IP Port for **luci**"*

2.3.1. Enabling IP Ports on Cluster Nodes

To allow Red Hat High Availability Add-On nodes to communicate with each other, you must enable the IP ports assigned to certain Red Hat High Availability Add-On components. *Table 2.1, "Enabled IP Ports on Red Hat High Availability Add-On Nodes"* lists the IP port numbers, their respective protocols, and the components to which the port numbers are assigned. At each cluster node, enable IP ports according to *Table 2.1, "Enabled IP Ports on Red Hat High Availability Add-On Nodes"*. You can use **`system-config-firewall`** to enable the IP ports.

Table 2.1. Enabled IP Ports on Red Hat High Availability Add-On Nodes

IP Port Number	Protocol	Component
5404, 5405	UDP	**`corosync/cman`** (Cluster Manager)
11111	TCP	**`ricci`** (propagates updated cluster information)
21064	TCP	**`dlm`** (Distributed Lock Manager)
16851	TCP	**`modclusterd`**

2.3.2. Enabling the IP Port for luci

To allow client computers to communicate with a computer that runs **luci** (the **Conga** user interface server), you must enable the IP port assigned to **luci**. At each computer that runs **luci**, enable the IP port according to *Table 2.2, "Enabled IP Port on a Computer That Runs **luci**"*.

Note

If a cluster node is running **luci**, port 11111 should already have been enabled.

Table 2.2. Enabled IP Port on a Computer That Runs **luci**

IP Port Number	Protocol	Component
8084	TCP	**luci** (**Conga** user interface server)

As of the Red Hat Enterprise Linux 6.1 release, which enabled configuration by means of the **`/etc/sysconfig/luci`** file, you can specifically configure the only IP address **luci** is being served at. You

can use this capability if your server infrastructure incorporates more than one network and you want to access **luci** from the internal network only. To do this, uncomment and edit the line in the file that specifies **host**. For example, to change the **host** setting in the file to 10.10.10.10, edit the **host** line as follows:

```
host = 10.10.10.10
```

For more information on the **/etc/sysconfig/luci** file, refer to *Section 2.4, "Configuring **luci** with **/etc/sysconfig/luci**"*.

2.4. Configuring luci with /etc/sysconfig/luci

As of the Red Hat Enterprise Linux 6.1 release, you can configure some aspects of **luci**'s behavior by means of the **/etc/sysconfig/luci** file. The parameters you can change with this file include auxiliary settings of the running environment used by the init script as well as server configuration. In addition, you can edit this file to modify some application configuration parameters. There are instructions within the file itself describing which configuration parameters you can change by editing this file.

In order to protect the intended format, you should not change the non-configuration lines of the **/etc/sysconfig/luci** file when you edit the file. Additionally, you should take care to follow the required syntax for this file, particularly for the **INITSCRIPT** section which does not allow for white spaces around the equal sign and requires that you use quotation marks to enclose strings containing white spaces.

The following example shows how to change the port at which **luci** is being served by editing the **/etc/sysconfig/luci** file.

1. Uncomment the following line in the **/etc/sysconfig/luci** file:

   ```
   #port = 4443
   ```

2. Replace 4443 with the desired port number, which must be higher than or equal to 1024 (not a privileged port). For example, you can edit that line of the file as follows to set the port at which **luci** is being served to 7084.

   ```
   port = 7084
   ```

3. Restart the **luci** service for the changes to take effect.

Important

When you modify a configuration parameter in the **/etc/sysconfig/luci** file to redefine a default value, you should take care to use the new value in place of the documented default value. For example, when you modify the port at which **luci** is being served, make sure that you specify the modified value when you enable an IP port for **luci**, as described in *Section 2.3.2, "Enabling the IP Port for **luci**"*.

Modified port and host parameters will automatically be reflected in the URL displayed when the **luci** service starts, as described in *Section 3.2, "Starting **luci**"*. You should use this URL to access **luci**.

For more complete information on the parameters you can configure with the **/etc/sysconfig/luci** file, refer to the documentation within the file itself.

2.5. Configuring ACPI For Use with Integrated Fence Devices

If your cluster uses integrated fence devices, you must configure ACPI (Advanced Configuration and Power Interface) to ensure immediate and complete fencing.

Note

For the most current information about integrated fence devices supported by Red Hat High Availability Add-On, refer to *http://www.redhat.com/cluster_suite/hardware/*[1].

If a cluster node is configured to be fenced by an integrated fence device, disable ACPI Soft-Off for that node. Disabling ACPI Soft-Off allows an integrated fence device to turn off a node immediately and completely rather than attempting a clean shutdown (for example, **shutdown -h now**). Otherwise, if ACPI Soft-Off is enabled, an integrated fence device can take four or more seconds to turn off a node (refer to note that follows). In addition, if ACPI Soft-Off is enabled and a node panics or freezes during shutdown, an integrated fence device may not be able to turn off the node. Under those circumstances, fencing is delayed or unsuccessful. Consequently, when a node is fenced with an integrated fence device and ACPI Soft-Off is enabled, a cluster recovers slowly or requires administrative intervention to recover.

[1] http://www.redhat.com/cluster_suite/hardware/

Note

The amount of time required to fence a node depends on the integrated fence device used. Some integrated fence devices perform the equivalent of pressing and holding the power button; therefore, the fence device turns off the node in four to five seconds. Other integrated fence devices perform the equivalent of pressing the power button momentarily, relying on the operating system to turn off the node; therefore, the fence device turns off the node in a time span much longer than four to five seconds.

To disable ACPI Soft-Off, use **chkconfig** management and verify that the node turns off immediately when fenced. The preferred way to disable ACPI Soft-Off is with **chkconfig** management: however, if that method is not satisfactory for your cluster, you can disable ACPI Soft-Off with one of the following alternate methods:

- Changing the BIOS setting to "instant-off" or an equivalent setting that turns off the node without delay

Note

Disabling ACPI Soft-Off with the BIOS may not be possible with some computers.

- Appending **acpi=off** to the kernel boot command line of the **/boot/grub/grub.conf** file

Important

This method completely disables ACPI; some computers do not boot correctly if ACPI is completely disabled. Use this method *only* if the other methods are not effective for your cluster.

The following sections provide procedures for the preferred method and alternate methods of disabling ACPI Soft-Off:

- *Section 2.5.1, "Disabling ACPI Soft-Off with **chkconfig** Management"* — Preferred method
- *Section 2.5.2, "Disabling ACPI Soft-Off with the BIOS"* — First alternate method
- *Section 2.5.3, "Disabling ACPI Completely in the **grub.conf** File"* — Second alternate method

2.5.1. Disabling ACPI Soft-Off with chkconfig Management

You can use **chkconfig** management to disable ACPI Soft-Off either by removing the ACPI daemon (**acpid**) from **chkconfig** management or by turning off **acpid**.

Note

This is the preferred method of disabling ACPI Soft-Off.

Disable ACPI Soft-Off with **chkconfig** management at each cluster node as follows:

1. Run either of the following commands:
   - **chkconfig --del acpid** — This command removes **acpid** from **chkconfig** management.

     — OR —

   - **chkconfig --level 2345 acpid off** — This command turns off **acpid**.
2. Reboot the node.
3. When the cluster is configured and running, verify that the node turns off immediately when fenced.

Note

You can fence the node with the **fence_node** command or **Conga**.

2.5.2. Disabling ACPI Soft-Off with the BIOS

The preferred method of disabling ACPI Soft-Off is with **chkconfig** management (*Section 2.5.1, "Disabling ACPI Soft-Off with **chkconfig** Management"*). However, if the preferred method is not effective for your cluster, follow the procedure in this section.

Note

Disabling ACPI Soft-Off with the BIOS may not be possible with some computers.

You can disable ACPI Soft-Off by configuring the BIOS of each cluster node as follows:

1. Reboot the node and start the **BIOS CMOS Setup Utility** program.
2. Navigate to the **Power** menu (or equivalent power management menu).
3. At the **Power** menu, set the **Soft-Off by PWR-BTTN** function (or equivalent) to **Instant-Off** (or the equivalent setting that turns off the node via the power button without delay). *Example 2.1, "**BIOS CMOS Setup Utility**: **Soft-Off by PWR-BTTN** set to **Instant-Off**"* shows a **Power** menu with **ACPI Function** set to **Enabled** and **Soft-Off by PWR-BTTN** set to **Instant-Off**.

Note

The equivalents to **ACPI Function**, **Soft-Off by PWR-BTTN**, and **Instant-Off** may vary among computers. However, the objective of this procedure is to configure the BIOS so that the computer is turned off via the power button without delay.

4. Exit the **BIOS CMOS Setup Utility** program, saving the BIOS configuration.
5. When the cluster is configured and running, verify that the node turns off immediately when fenced.

Note

You can fence the node with the **fence_node** command or **Conga**.

Example 2.1. **BIOS CMOS Setup Utility**: **Soft-Off by PWR-BTTN** set to **Instant-Off**

```
+---------------------------------------------|-------------------+
|    ACPI Function             [Enabled]      |    Item Help      |
|    ACPI Suspend Type         [S1(POS)]      |-------------------|
|  x Run VGABIOS if S3 Resume   Auto          |   Menu Level   *  |
|    Suspend Mode              [Disabled]     |                   |
|    HDD Power Down            [Disabled]     |                   |
|    Soft-Off by PWR-BTTN      [Instant-Off   |                   |
|    CPU THRM-Throttling       [50.0%]        |                   |
|    Wake-Up by PCI card       [Enabled]      |                   |
|    Power On by Ring          [Enabled]      |                   |
|    Wake Up On LAN            [Enabled]      |                   |
|  x USB KB Wake-Up From S3     Disabled      |                   |
|    Resume by Alarm           [Disabled]     |                   |
|  x  Date(of Month) Alarm       0            |                   |
|  x  Time(hh:mm:ss) Alarm       0 :  0 :     |                   |
|    POWER ON Function         [BUTTON ONLY   |                   |
|  x KB Power ON Password       Enter         |                   |
|  x Hot Key Power ON           Ctrl-F1       |                   |
|                                             |                   |
|                                             |                   |
+---------------------------------------------|-------------------+
```

This example shows **ACPI Function** set to **Enabled**, and **Soft-Off by PWR-BTTN** set to **Instant-Off**.

2.5.3. Disabling ACPI Completely in the grub.conf File

The preferred method of disabling ACPI Soft-Off is with **chkconfig** management (*Section 2.5.1, "Disabling ACPI Soft-Off with **chkconfig** Management"*). If the preferred method is not effective for your cluster, you can disable ACPI Soft-Off with the BIOS power management (*Section 2.5.2, "Disabling ACPI Soft-Off with the BIOS"*). If neither of those methods is effective for your cluster,

you can disable ACPI completely by appending **acpi=off** to the kernel boot command line in the **grub.conf** file.

Important

This method completely disables ACPI; some computers do not boot correctly if ACPI is completely disabled. Use this method *only* if the other methods are not effective for your cluster.

You can disable ACPI completely by editing the **grub.conf** file of each cluster node as follows:

1. Open **/boot/grub/grub.conf** with a text editor.
2. Append **acpi=off** to the kernel boot command line in **/boot/grub/grub.conf** (refer to *Example 2.2, "Kernel Boot Command Line with **acpi=off** Appended to It"*).
3. Reboot the node.
4. When the cluster is configured and running, verify that the node turns off immediately when fenced.

Note

You can fence the node with the **fence_node** command or **Conga**.

Example 2.2. Kernel Boot Command Line with **acpi=off** Appended to It

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/mapper/vg_doc01-lv_root
#          initrd /initrd-[generic-]version.img
#boot=/dev/hda
default=0
timeout=5
serial --unit=0 --speed=115200
terminal --timeout=5 serial console
title Red Hat Enterprise Linux Server (2.6.32-193.el6.x86_64)
        root (hd0,0)
        kernel /vmlinuz-2.6.32-193.el6.x86_64 ro root=/dev/mapper/vg_doc01-lv_root
 console=ttyS0,115200n8 acpi=off
        initrd /initramrs-2.6.32-131.0.15.el6.x86_64.img
```

In this example, **acpi=off** has been appended to the kernel boot command line — the line starting with "kernel /vmlinuz-2.6.32-193.el6.x86_64.img".

2.6. Considerations for Configuring HA Services

You can create a cluster to suit your needs for high availability by configuring HA (high-availability) services. The key component for HA service management in the Red Hat High Availability Add-On, **rgmanager**, implements cold failover for off-the-shelf applications. In the Red Hat High Availability Add-On, an application is configured with other cluster resources to form an HA service that can fail over from one cluster node to another with no apparent interruption to cluster clients. HA-service failover can occur if a cluster node fails or if a cluster system administrator moves the service from one cluster node to another (for example, for a planned outage of a cluster node).

To create an HA service, you must configure it in the cluster configuration file. An HA service comprises cluster *resources*. Cluster resources are building blocks that you create and manage in the cluster configuration file — for example, an IP address, an application initialization script, or a Red Hat GFS2 shared partition.

An HA service can run on only one cluster node at a time to maintain data integrity. You can specify failover priority in a failover domain. Specifying failover priority consists of assigning a priority level to each node in a failover domain. The priority level determines the failover order — determining which node that an HA service should fail over to. If you do not specify failover priority, an HA service can fail over to any node in its failover domain. Also, you can specify if an HA service is restricted to run only on nodes of its associated failover domain. (When associated with an unrestricted failover domain, an HA service can start on any cluster node in the event no member of the failover domain is available.)

Figure 2.1, "Web Server Cluster Service Example" shows an example of an HA service that is a web server named "content-webserver". It is running in cluster node B and is in a failover domain that consists of nodes A, B, and D. In addition, the failover domain is configured with a failover priority to fail over to node D before node A and to restrict failover to nodes only in that failover domain. The HA service comprises these cluster resources:

- IP address resource — IP address 10.10.10.201.
- An application resource named "httpd-content" — a web server application init script **/etc/init.d/httpd** (specifying **httpd**).
- A file system resource — Red Hat GFS2 named "gfs2-content-webserver".

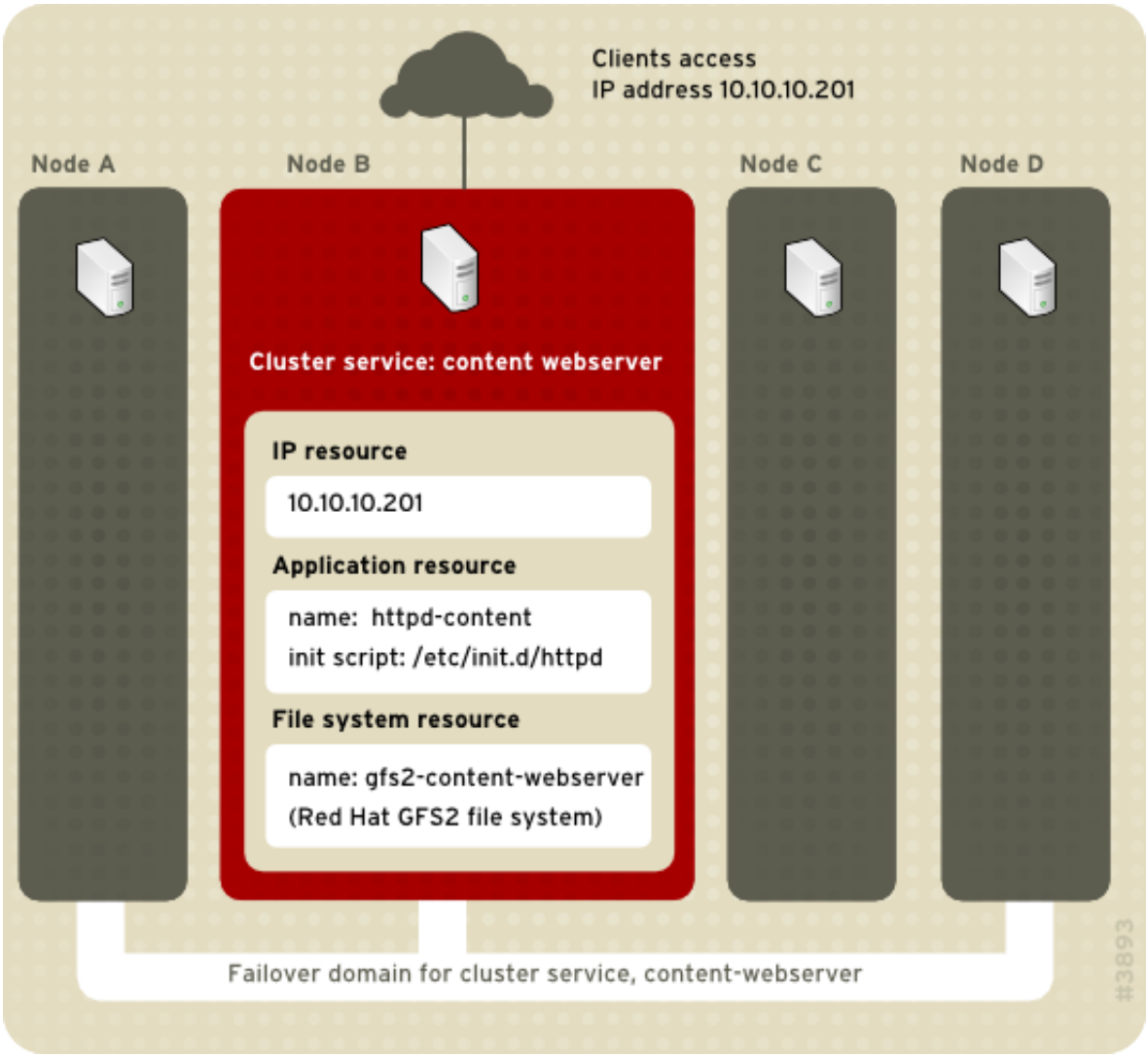

Figure 2.1. Web Server Cluster Service Example

Clients access the HA service through the IP address 10.10.10.201, enabling interaction with the web server application, httpd-content. The httpd-content application uses the gfs2-content-webserver file system. If node B were to fail, the content-webserver HA service would fail over to node D. If node D were not available or also failed, the service would fail over to node A. Failover would occur with minimal service interruption to the cluster clients. For example, in an HTTP service, certain state information may be lost (like session data). The HA service would be accessible from another cluster node via the same IP address as it was before failover.

Note

For more information about HA services and failover domains, refer to the *High Availability Add-On Overview*. For information about configuring failover domains, refer to *Chapter 3, Configuring Red Hat High Availability Add-On With **Conga***(using **Conga**) or *Chapter 7, Configuring Red Hat High Availability Add-On With Command Line Tools* (using command line utilities).

An HA service is a group of cluster resources configured into a coherent entity that provides specialized services to clients. An HA service is represented as a resource tree in the cluster

configuration file, **/etc/cluster/cluster.conf** (in each cluster node). In the cluster configuration file, each resource tree is an XML representation that specifies each resource, its attributes, and its relationship among other resources in the resource tree (parent, child, and sibling relationships).

Note

Because an HA service consists of resources organized into a hierarchical tree, a service is sometimes referred to as a *resource tree* or *resource group*. Both phrases are synonymous with *HA service*.

At the root of each resource tree is a special type of resource — a *service resource*. Other types of resources comprise the rest of a service, determining its characteristics. Configuring an HA service consists of creating a service resource, creating subordinate cluster resources, and organizing them into a coherent entity that conforms to hierarchical restrictions of the service.

There are two major considerations to take into account when configuring an HA service:

- The types of resources needed to create a service
- Parent, child, and sibling relationships among resources

The types of resources and the hierarchy of resources depend on the type of service you are configuring.

The types of cluster resources are listed in *Appendix B, HA Resource Parameters*. Information about parent, child, and sibling relationships among resources is described in *Appendix C, HA Resource Behavior*.

2.7. Configuration Validation

The cluster configuration is automatically validated according to the cluster schema at **/usr/share/cluster/cluster.rng** during startup time and when a configuration is reloaded. Also, you can validate a cluster configuration any time by using the **ccs_config_validate** command.

An annotated schema is available for viewing at **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** (for example **/usr/share/doc/cman-3.0.12/cluster_conf.html**).

Configuration validation checks for the following basic errors:

- XML validity — Checks that the configuration file is a valid XML file.
- Configuration options — Checks to make sure that options (XML elements and attributes) are valid.
- Option values — Checks that the options contain valid data (limited).

The following examples show a valid configuration and invalid configurations that illustrate the validation checks:

- Valid configuration — *Example 2.3, "**cluster.conf** Sample Configuration: Valid File"*
- Invalid XML — *Example 2.4, "**cluster.conf** Sample Configuration: Invalid XML"*

- Invalid option — *Example 2.5, "**cluster.conf** Sample Configuration: Invalid Option"*
- Invalid option value — *Example 2.6, "**cluster.conf** Sample Configuration: Invalid Option Value"*

Example 2.3. **cluster.conf** Sample Configuration: Valid File

```
<cluster name="mycluster" config_version="1">
  <logging debug="off"/>
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
   </fencedevices>
   <rm>
   </rm>
</cluster>
```

Example 2.4. **cluster.conf** Sample Configuration: Invalid XML

```
<cluster name="mycluster" config_version="1">
  <logging debug="off"/>
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
   </fencedevices>
   <rm>
   </rm>
<cluster>         <----------------INVALID
```

In this example, the last line of the configuration (annotated as "INVALID" here) is missing a slash — it is **<cluster>** instead of **</cluster>**.

Example 2.5. **cluster.conf** Sample Configuration: Invalid Option

```
<cluster name="mycluster" config_version="1">
  <loging debug="off"/>         <----------------INVALID
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
   </fencedevices>
   <rm>
   </rm>
<cluster>
```

In this example, the second line of the configuration (annotated as "INVALID" here) contains an invalid XML element — it is **loging** instead of **logging**.

Example 2.6. **cluster.conf** Sample Configuration: Invalid Option Value

```
<cluster name="mycluster" config_version="1">
  <loging debug="off"/>
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="-1">  <--------INVALID
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
   </fencedevices>
   <rm>
   </rm>
<cluster>
```

In this example, the fourth line of the configuration (annotated as "INVALID" here) contains an invalid value for the XML attribute, **nodeid** in the **clusternode** line for **node-01.example.com**.

> The value is a negative value ("-1") instead of a positive value ("1"). For the **nodeid** attribute, the value must be a positive value.

2.8. Considerations for NetworkManager

The use of **NetworkManager** is not supported on cluster nodes. If you have installed **NetworkManager** on your cluster nodes, you should either remove it or disable it.

Note

The **cman** service will not start if **NetworkManager** is either running or has been configured to run with the **chkconfig** command.

2.9. Considerations for Using Quorum Disk

Quorum Disk is a disk-based quorum daemon, **qdiskd**, that provides supplemental heuristics to determine node fitness. With heuristics you can determine factors that are important to the operation of the node in the event of a network partition. For example, in a four-node cluster with a 3:1 split, ordinarily, the three nodes automatically "win" because of the three-to-one majority. Under those circumstances, the one node is fenced. With **qdiskd** however, you can set up heuristics that allow the one node to win based on access to a critical resource (for example, a critical network path). If your cluster requires additional methods of determining node health, then you should configure **qdiskd** to meet those needs.

Note

Configuring **qdiskd** is not required unless you have special requirements for node health. An example of a special requirement is an "all-but-one" configuration. In an all-but-one configuration, **qdiskd** is configured to provide enough quorum votes to maintain quorum even though only one node is working.

Important

Overall, heuristics and other **qdiskd** parameters for your deployment depend on the site environment and special requirements needed. To understand the use of heuristics and other **qdiskd** parameters, refer to the qdisk(5) man page. If you require assistance understanding and using **qdiskd** for your site, contact an authorized Red Hat support representative.

If you need to use **qdiskd**, you should take into account the following considerations:

Cluster node votes
: When using Quorum Disk, each cluster node must have one vote.

CMAN membership timeout value
: The CMAN membership timeout value (the time a node needs to be unresponsive before CMAN considers that node to be dead, and not a member) should be at least two times that of the **qdiskd** membership timeout value. The reason is because the quorum daemon must detect failed nodes on its own, and can take much longer to do so than CMAN. The default value for CMAN membership timeout is 10 seconds. Other site-specific conditions may affect the relationship between the membership timeout values of CMAN and **qdiskd**. For assistance with adjusting the CMAN membership timeout value, contact an authorized Red Hat support representative.

Fencing
: To ensure reliable fencing when using **qdiskd**, use power fencing. While other types of fencing can be reliable for clusters not configured with **qdiskd**, they are not reliable for a cluster configured with **qdiskd**.

Maximum nodes
: A cluster configured with **qdiskd** supports a maximum of 16 nodes. The reason for the limit is because of scalability; increasing the node count increases the amount of synchronous I/O contention on the shared quorum disk device.

Quorum disk device
: A quorum disk device should be a shared block device with concurrent read/write access by all nodes in a cluster. The minimum size of the block device is 10 Megabytes. Examples of shared block devices that can be used by **qdiskd** are a multi-port SCSI RAID array, a Fibre Channel RAID SAN, or a RAID-configured iSCSI target. You can create a quorum disk device with **mkqdisk**, the Cluster Quorum Disk Utility. For information about using the utility refer to the mkqdisk(8) man page.

Note

Using JBOD as a quorum disk is not recommended. A JBOD cannot provide dependable performance and therefore may not allow a node to write to it quickly enough. If a node is unable to write to a quorum disk device quickly enough, the node is falsely evicted from a cluster.

2.10. Red Hat High Availability Add-On and SELinux

The High Availability Add-On for Red Hat Enterprise Linux 6 supports SELinux in the **enforcing** state with the SELinux policy type set to **targeted**.

For more information about SELinux, refer to *Deployment Guide* for Red Hat Enterprise Linux 6.

2.11. Multicast Addresses

Red Hat High Availability Add-On nodes communicate among each other using multicast addresses. Therefore, each network switch and associated networking equipment in the Red Hat High Availability Add-On must be configured to enable multicast addresses and support IGMP (Internet Group

Management Protocol). Ensure that each network switch and associated networking equipment in the Red Hat High Availability Add-On are capable of supporting multicast addresses and IGMP; if they are, ensure that multicast addressing and IGMP are enabled. Without multicast and IGMP, not all nodes can participate in a cluster, causing the cluster to fail; use UDP unicast in these environments, as described in *Section 2.13, "UDP Unicast Traffic"*.

Note

Procedures for configuring network switches and associated networking equipment vary according each product. Refer to the appropriate vendor documentation or other information about configuring network switches and associated networking equipment to enable multicast addresses and IGMP.

2.12. Configuring the iptables Firewall to Allow Cluster Components

You can use the following filtering to allow traffic through the **iptables** firewall for the various cluster components.

For **corosync**, use the following filtering. Port 5405 is used to receive multicast traffic.

```
iptables -I INPUT -p udp -m state --state NEW -m multiport --dports 5404,5405 -j ACCEPT
```

For **ricci**:

```
iptables -I INPUT -p tcp -m state --state NEW -m multiport --dports 11111 -j ACCEPT
```

For **modcluster**:

```
iptables -I INPUT -p tcp -m state --state NEW -m multiport --dports 16851 -j ACCEPT
```

For **luci**:

```
iptables -I INPUT -p tcp -m state --state NEW -m multiport --dports 8084 -j ACCEPT
```

For **DLM**:

```
iptables -I INPUT -p tcp -m state --state NEW -m multiport --dports 21064 -j ACCEPT
```

After executing these commands, run the following command.

```
service iptables save ; service iptables restart
```

In Red Hat Enterprise Linux 6, **rgmanager** does not access the network directly; **rgmanager** communication happens by means of **corosync** network transport. Enabling **corosync** allows **rgmanager** (or any **corosync** clients) to work automatically.

2.13. UDP Unicast Traffic

As of the Red Hat Enterprise Linux 6.2 release, Red Hat High-Availability Add-On nodes can communicate with each other using the UDP Unicast transport mechanism. It is recommended, however, that you use IP multicasting for the cluster network. UDP unicast is an alternative that can be used when IP multicasting is not available. For GFS2 deployments using UDP unicast is not recommended.

You can configure the Red Hat High-Availability Add-On to use UDP unicast by setting the **cman transport="udpu"** parameter in the **cluster.conf** configuration file. You can also specify Unicast from the **Network Configuration** page of the **Conga** user interface, as described in *Section 3.5.3, "Network Configuration"*.

2.14. Considerations for ricci

For Red Hat Enterprise Linux 6, **ricci** replaces **ccsd**. Therefore, it is necessary that **ricci** is running in each cluster node to be able to propagate updated cluster configuration whether it is via the **cman_tool version -r** command, the **ccs** command, or the **luci** user interface server. You can start **ricci** by using **service ricci start** or by enabling it to start at boot time via **chkconfig**. For information on enabling IP ports for **ricci**, refer to *Section 2.3.1, "Enabling IP Ports on Cluster Nodes"*.

For the Red Hat Enterprise Linux 6.1 release and later, using **ricci** requires a password the first time you propagate updated cluster configuration from any particular node. You set the **ricci** password as root after you install **ricci** on your system with the **passwd ricci** command, for user **ricci**.

Configuring Red Hat High Availability Add-On With Conga

This chapter describes how to configure Red Hat High Availability Add-On software using **Conga**. For information on using **Conga** to manage a running cluster, see *Chapter 4, Managing Red Hat High Availability Add-On With **Conga***.

Note

Conga is a graphical user interface that you can use to administer the Red Hat High Availability Add-On. Note, however, that in order to use this interface effectively you need to have a good and clear understanding of the underlying concepts. Learning about cluster configuration by exploring the available features in the user interface is not recommended, as it may result in a system that is not robust enough to keep all services running when components fail.

This chapter consists of the following sections:

- *Section 3.1, “Configuration Tasks”*
- *Section 3.2, “Starting **luci**”*
- *Section 3.3, “Controlling Access to luci”*
- *Section 3.4, “Creating a Cluster”*
- *Section 3.5, “Global Cluster Properties”*
- *Section 3.6, “Configuring Fence Devices”*
- *Section 3.7, “Configuring Fencing for Cluster Members”*
- *Section 3.8, “Configuring a Failover Domain”*
- *Section 3.9, “Configuring Global Cluster Resources”*
- *Section 3.10, “Adding a Cluster Service to the Cluster”*

3.1. Configuration Tasks

Configuring Red Hat High Availability Add-On software with **Conga** consists of the following steps:

1. Configuring and running the **Conga** configuration user interface — the **luci** server. Refer to *Section 3.2, “Starting **luci**”*.
2. Creating a cluster. Refer to *Section 3.4, “Creating a Cluster”*.
3. Configuring global cluster properties. Refer to *Section 3.5, “Global Cluster Properties”*.
4. Configuring fence devices. Refer to *Section 3.6, “Configuring Fence Devices”*.
5. Configuring fencing for cluster members. Refer to *Section 3.7, “Configuring Fencing for Cluster Members”*.

6. Creating failover domains. Refer to *Section 3.8, "Configuring a Failover Domain"*.
7. Creating resources. Refer to *Section 3.9, "Configuring Global Cluster Resources"*.
8. Creating cluster services. Refer to *Section 3.10, "Adding a Cluster Service to the Cluster"*.

3.2. Starting luci

Installing ricci

Using **luci** to configure a cluster requires that **ricci** be installed and running on the cluster nodes, as described in *Section 2.14, "Considerations for **ricci**"*. As noted in that section, using **ricci** requires a password which **luci** requires you to enter for each cluster node when you create a cluster, as described in *Section 3.4, "Creating a Cluster"*.

Before starting **luci**, ensure that the IP ports on your cluster nodes allow connections to port 11111 from the **luci** server on any nodes that **luci** will be communicating with. For information on enabling IP ports on cluster nodes, see *Section 2.3.1, "Enabling IP Ports on Cluster Nodes"*.

To administer Red Hat High Availability Add-On with **Conga**, install and run **luci** as follows:

1. Select a computer to host **luci** and install the **luci** software on that computer. For example:

```
# yum install luci
```

Note

Typically, a computer in a server cage or a data center hosts **luci**; however, a cluster computer can host **luci**.

2. Start **luci** using **service luci start**. For example:

```
# service luci start
Starting luci: generating https SSL certificates...  done
                                                           [  OK  ]

Please, point your web browser to https://nano-01:8084 to access luci
```

Note

As of Red Hat Enterprise Linux release 6.1, you can configure some aspects of **luci**'s behavior by means of the **/etc/sysconfig/luci** file, including the port and host parameters, as described in *Section 2.4, "Configuring **luci** with **/etc/sysconfig/luci**"*. Modified port and host parameters will automatically be reflected in the URL displayed when the **luci** service starts.

3. At a Web browser, place the URL of the **luci** server into the URL address box and click **Go** (or the equivalent). The URL syntax for the **luci** server is **https://*luci_server_hostname*:*luci_server_port***. The default value of *luci_server_port* is **8084**.

   The first time you access **luci**, a web browser specific prompt regarding the self-signed SSL certificate (of the **luci** server) is displayed. Upon acknowledging the dialog box or boxes, your Web browser displays the **luci** login page.

4. Although any user able to authenticate on the system that is hosting **luci** can log in to **luci**, as of Red Hat Enterprise Linux 6.2 only the root user on the system that is running **luci** can access any of the **luci** components until an administrator (the root user or a user with administrator permission) sets permissions for that user. For information on setting **luci** permissions for users, refer to *Section 3.3, "Controlling Access to luci"*.

   Logging in to **luci** displays the **luci Homebase** page, as shown in *Figure 3.1, "luci Homebase page"*.

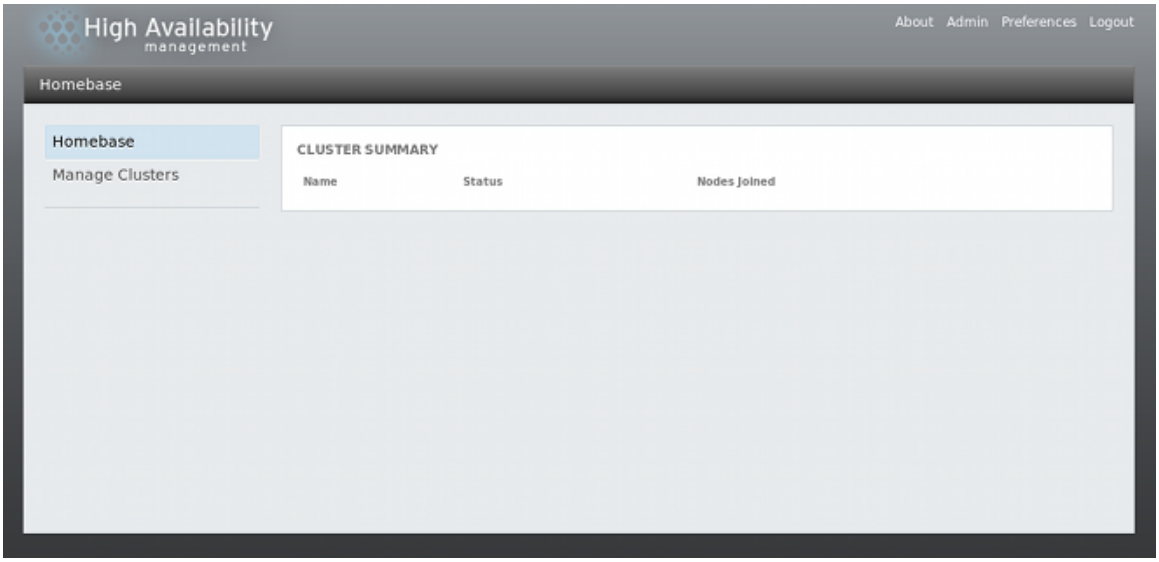

Figure 3.1. luci Homebase page

3.3. Controlling Access to luci

As of Red Hat Enterprise Linux 6.2, the root user or a user who has been granted **luci** administrator permissions on a system running **luci** can control access to the various **luci** components by setting permissions for the individual users on a system. To set the user permissions, log in to **luci** as **root**

and click the **Admin** selection in the upper right corner of the **luci** screen. This brings up the **USER PERMISSIONS** page, showing a dropdown menu from which you can select a user.

After selecting a user, you can check whether to allow the following permissions for that user:

Luci Administrator
Grants the user the same permissions as the root user, with full permissions on all clusters and the ability to set or remove permissions on all other users except root, whose permissions cannot be restricted.

Can Create Clusters
Allows the user to create new clusters, as described in *Section 3.4, "Creating a Cluster"*.

Can Import Existing Clusters
Allows the user to add an existing cluster to the **luci** interface, as described in *Section 4.1, "Adding an Existing Cluster to the luci Interface"*.

For each cluster that has been created or imported to **luci**, you can set the following permissions for the indicated user:

Can View This Cluster
Allows the user to view the specified cluster.

Can Change the Cluster Configuration
Allows the user to modify the configuration for the specified cluster, with the exception of adding and removing cluster nodes.

Can Enable, Disable, Relocate, and Migrate Service Groups
Allows the user to manage high-availability services, as described in *Section 4.4, "Managing High-Availability Services"*.

Can Stop, Start, and Reboot Cluster Nodes
Allows the user to manage the individual nodes of a cluster, as described in *Section 4.2, "Managing Cluster Nodes"*.

Can Add and Delete Nodes
Allows the user to add and delete nodes from a cluster, as described in *Section 3.4, "Creating a Cluster"*.

Can Remove This Cluster from Luci
Allows the user to remove a cluster from the **luci** interface, as described in *Section 4.3, "Starting, Stopping, Restarting, and Deleting Clusters"*.

Click **Submit** for the permissions to take affect, or click **Reset** to return to the initial values.

3.4. Creating a Cluster

Creating a cluster with **luci** consists of naming a cluster, adding cluster nodes to the cluster, entering the **ricci** passwords for each node, and submitting the request to create a cluster. If the node information and passwords are correct, **Conga** automatically installs software into the cluster nodes (if the appropriate software packages are not currently installed) and starts the cluster. Create a cluster as follows:

1. Click **Manage Clusters** from the menu on the left side of the **luci Homebase** page. The **Clusters** screen appears, as shown in *Figure 3.2, "luci cluster management page"*.

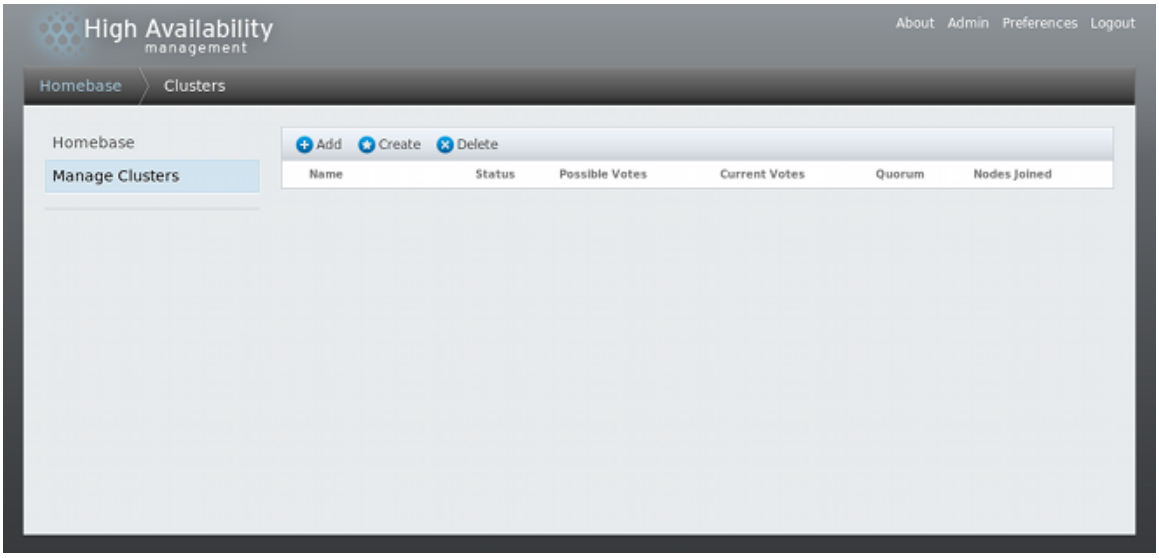

Figure 3.2. luci cluster management page

2. Click **Create**. The **Create New Cluster** dialog box appears, as shown in *Figure 3.3, "luci cluster creation dialog box"*.

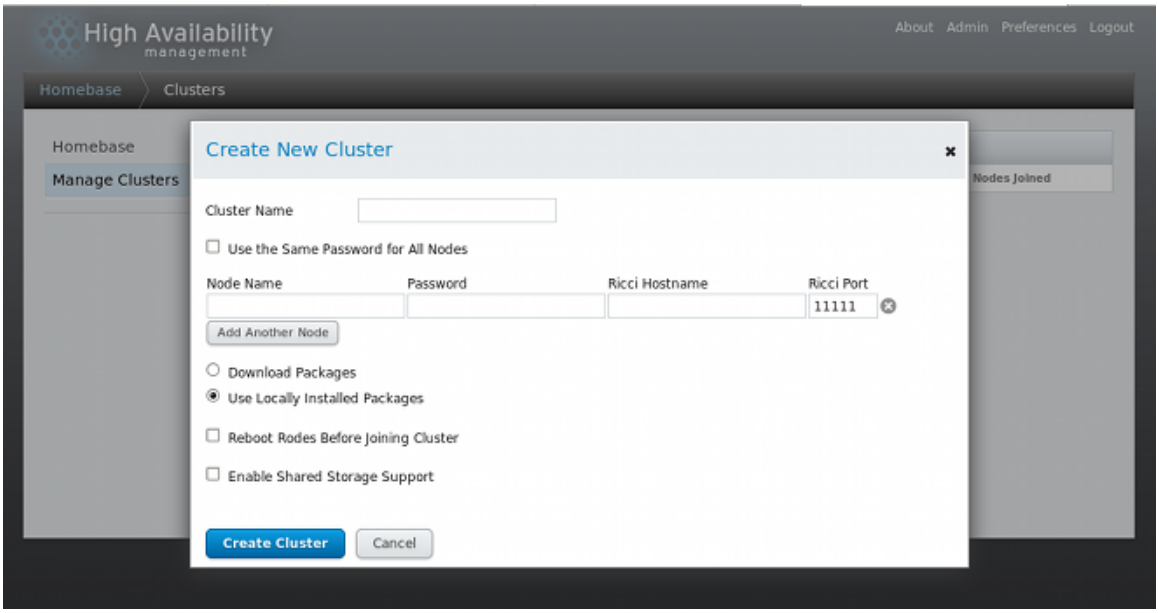

Figure 3.3. luci cluster creation dialog box

3. Enter the following parameters on the **Create New Cluster** dialog box, as necessary:

   - At the **Cluster Name** text box, enter a cluster name. The cluster name cannot exceed 15 characters.
   - If each node in the cluster has the same **ricci** password, you can check **Use the same password for all nodes** to autofill the **password** field as you add nodes.
   - Enter the node name for a node in the cluster in the **Node Name** column and enter the **ricci** password for the node in the **Password** column.
   - If your system is configured with a dedicated private network that is used only for cluster traffic, you may want to configure **luci** to communicate with **ricci** on an address that is different from

the address to which the cluster node name resolves. You can do this by entering that address as the **Ricci Hostname**.

- If you are using a different port for the **ricci** agent than the default of 11111, you can change that parameter.
- Click on **Add Another Node** and enter the node name and **ricci** password for each additional node in the cluster.
- If you do not want to upgrade the cluster software packages that are already installed on the nodes when you create the cluster, leave the **Use locally installed packages** option selected. If you want to upgrade all cluster software packages, select the **Download Packages** option.

Note

Whether you select the **Use locally installed packages** or the **Download Packages** option, if any of the base cluster components are missing (**`cman`**, **`rgmanager`**, **`modcluster`** and all their dependencies), they will be installed. If they cannot be installed, the node creation will fail.

- Select **Reboot nodes before joining cluster** if desired.
- Select **Enable shared storage support** if clustered storage is required; this downloads the packages that support clustered storage and enables clustered LVM. You should select this only when you have access to the Resilient Storage Add-On or the Scalable File System Add-On.

4. Click **Create Cluster**. Clicking **Create Cluster** causes the following actions:

   a. If you have selected **Download Packages**, the cluster software packages are downloaded onto the nodes.

   b. Cluster software is installed onto the nodes (or it is verified that the appropriate software packages are installed).

   c. The cluster configuration file is updated and propagated to each node in the cluster.

   d. The added nodes join the cluster.

   A message is displayed indicating that the cluster is being created. When the cluster is ready, the display shows the status of the newly created cluster, as shown in *Figure 3.4, "Cluster node display"*. Note that if **ricci** is not running on any of the nodes, the cluster creation will fail.

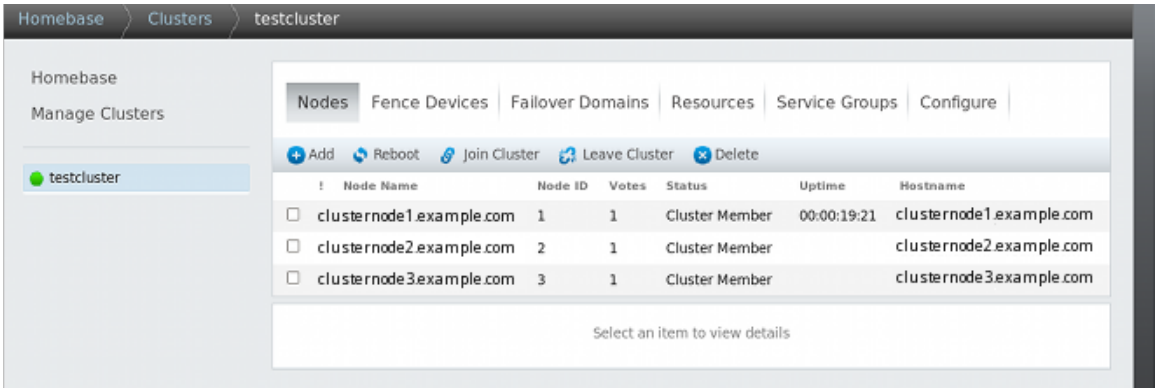

Figure 3.4. Cluster node display

5. After clicking **Create Cluster** to create the cluster, you can add or delete nodes from the cluster by clicking the **Add** or **Delete** function from the menu at the top of the cluster node display page. Unless you are deleting an entire cluster, nodes must be stopped before being deleted. For information on deleting a node from an existing cluster that is currently in operation, see *Section 4.2.4, "Deleting a Member from a Cluster"*.

Note

Removing a cluster node from the cluster is a destructive operation that cannot be undone.

3.5. Global Cluster Properties

When you select a cluster to configure, a cluster-specific page is displayed. The page provides an interface for configuring cluster-wide properties. You can configure cluster-wide properties by clicking on **Configure** along the top of the cluster display. This yields a tabbed interface which provides the following tabs: **General**, **Fence Daemon**, **Network**, **QDisk** and **Logging**. To configure the parameters in those tabs, follow the steps in the following sections. If you do not need to configure parameters in a tab, skip the section for that tab.

3.5.1. Configuring General Properties

Clicking on the **General** tab displays the **General Properties** page, which provides an interface for modifying the configuration version.

- The **Cluster Name** text box displays the cluster name; it does not accept a cluster name change. The only way to change the name of a cluster is to create a new cluster configuration with the new name.
- The **Configuration Version** value is set to **1** at the time of cluster creation and is automatically incremented each time you modify your cluster configuration. However, if you need to set it to another value, you can specify it at the **Configuration Version** text box.

If you have changed the **Configuration Version** value, click **Apply** for this change to take effect.

3.5.2. Configuring Fence Daemon Properties

Clicking on the **Fence Daemon** tab displays the **Fence Daemon Properties** page, which provides an interface for configuring **Post Fail Delay** and **Post Join Delay**. The values you configure for these parameters are general fencing properties for the cluster. To configure specific fence devices for the nodes of the cluster, use the **Fence Devices** menu item of the cluster display, as described in *Section 3.6, "Configuring Fence Devices"*.

- The **Post Fail Delay** parameter is the number of seconds the fence daemon (**fenced**) waits before fencing a node (a member of the fence domain) after the node has failed. The **Post Fail Delay** default value is **0**. Its value may be varied to suit cluster and network performance.
- The **Post Join Delay** parameter is the number of seconds the fence daemon (**fenced**) waits before fencing a node after the node joins the fence domain. The **Post Join Delay** default value is **3**. A typical setting for **Post Join Delay** is between 20 and 30 seconds, but can vary according to cluster and network performance.

Enter the values required and click **Apply** for changes to take effect.

Note

For more information about **Post Join Delay** and **Post Fail Delay**, refer to the fenced(8) man page.

3.5.3. Network Configuration

Clicking on the **Network** tab displays the **Network Configuration** page, which provides an interface for configuring the network transport type.

You can use this tab to select one of the following options:

- **UDP Multicast and Let Cluster Choose the Multicast Address**

  This is the default setting. With this option selected, the Red Hat High Availability Add-On software creates a multicast address based on the cluster ID. It generates the lower 16 bits of the address and appends them to the upper portion of the address according to whether the IP protocol is IPv4 or IPv6:

  - For IPv4 — The address formed is 239.192. plus the lower 16 bits generated by Red Hat High Availability Add-On software.
  - For IPv6 — The address formed is FF15:: plus the lower 16 bits generated by Red Hat High Availability Add-On software.

Note

The cluster ID is a unique identifier that **cman** generates for each cluster. To view the cluster ID, run the **cman_tool status** command on a cluster node.

- **UDP Multicast and Specify the Multicast Address Manually**

If you need to use a specific multicast address, select this option enter a multicast address into the **Multicast Address** text box.

If you do specify a multicast address, you should use the 239.192.x.x series (or FF15:: for IPv6) that **cman** uses. Otherwise, using a multicast address outside that range may cause unpredictable results. For example, using 224.0.0.x (which is "All hosts on the network") may not be routed correctly, or even routed at all by some hardware.

Note

If you specify a multicast address, make sure that you check the configuration of routers that cluster packets pass through. Some routers may take a long time to learn addresses, seriously impacting cluster performance.

- **Unicast**

  As of the Red Hat Enterprise Linux 6.2 release, Red Hat High-Availability Add-On nodes can communicate each other using the UDP Unicast transport mechanism. It is recommended, however, that you use IP multicasting for the cluster network. UDP unicast is an alternative that can be used when IP multicasting is not available. For GFS2 deployments using UDP unicast is not recommended.

Click **Apply**. When changing the transport type, a cluster restart is necessary for the changes to take effect.

3.5.4. Quorum Disk Configuration

Clicking on the **QDisk** tab displays the **Quorum Disk Configuration** page, which provides an interface for configuring quorum disk parameters to configure if you need to use a quorum disk.

Important

Quorum disk parameters and heuristics depend on the site environment and the special requirements needed. To understand the use of quorum disk parameters and heuristics, refer to the qdisk(5) man page. If you require assistance understanding and using quorum disk, contact an authorized Red Hat support representative.

The **Do Not Use a Quorum Disk** parameter is enabled by default. If you need to use a quorum disk, click **Use a Quorum Disk**, enter quorum disk parameters, click **Apply**, and restart the cluster for the changes to take effect.

Table 3.1, "Quorum-Disk Parameters" describes the quorum disk parameters.

Table 3.1. Quorum-Disk Parameters

Parameter	Description
Specify Physical Device: By Device Label	Specifies the quorum disk label created by the **mkqdisk** utility. If this field is used, the quorum daemon reads the **/proc/partitions** file and checks for qdisk signatures on every block device found, comparing the label against the specified label. This is useful in configurations where the quorum device name differs among nodes.
Heuristics	**Path to Program** — The program used to determine if this heuristic is available. This can be anything that can be executed by **/bin/sh -c**. A return value of 0 indicates success; anything else indicates failure. This field is required. **Interval** — The frequency (in seconds) at which the heuristic is polled. The default interval for every heuristic is 2 seconds. **Score** — The weight of this heuristic. Be careful when determining scores for heuristics. The default score for each heuristic is 1. **TKO** — The number of consecutive failures required before this heuristic is declared unavailable.
Minimum Total Score	The minimum score for a node to be considered "alive". If omitted or set to 0, the default function, **floor((*n*+1)/2)**, is used, where *n* is the sum of the heuristics scores. The **Minimum Total Score** value must never exceed the sum of the heuristic scores; otherwise, the quorum disk cannot be available.

Note

Clicking **Apply** on the **QDisk Configuration** tab propagates changes to the cluster configuration file (**/etc/cluster/cluster.conf**) in each cluster node. However, for the quorum disk to operate, you must restart the cluster (refer to *Section 4.3, "Starting, Stopping, Restarting, and Deleting Clusters"*).

3.5.5. Logging Configuration

Clicking on the **Logging** tab displays the **Logging Configuration** page, which provides an interface for configuring logging settings.

You can configure the following settings for global logging configuration:

- Checking **Log Debugging Messages** enables debugging messages in the log file.
- Checking **Log Messages to Syslog** enables messages to **syslog**. You can select the **Syslog Message Facility** and the **Syslog Message Priority**. The **Syslog Message Priority** setting indicates that messages at the selected level and higher are sent to **syslog**.
- Checking **Log Messages to Log File** enables messages to the log file. You can specify the **Log File Path** name. The **logfile message priority** setting indicates that messages at the selected level and higher are written to the log file.

You can override the global logging settings for specific daemons by selecting one of the daemons listed beneath the **Daemon-specific Logging Overrides** heading at the bottom of the **Logging Configuration** page. After selecting the daemon, you can check whether to log the debugging

messages for that particular daemon. You can also specify the **syslog** and log file settings for that daemon.

Click **Apply** for the logging configuration changes you have specified to take effect.

3.6. Configuring Fence Devices

Configuring fence devices consists of creating, updating, and deleting fence devices for the cluster. You must configure the fence devices in a cluster before you can configure fencing for the nodes in the cluster.

Creating a fence device consists of selecting a fence device type and entering parameters for that fence device (for example, name, IP address, login, and password). Updating a fence device consists of selecting an existing fence device and changing parameters for that fence device. Deleting a fence device consists of selecting an existing fence device and deleting it.

This section provides procedures for the following tasks:

- Creating fence devices — Refer to *Section 3.6.1, "Creating a Fence Device"*. Once you have created and named a fence device, you can configure the fence devices for each node in the cluster, as described in *Section 3.7, "Configuring Fencing for Cluster Members"*.
- Updating fence devices — Refer to *Section 3.6.2, "Modifying a Fence Device"*.
- Deleting fence devices — Refer to *Section 3.6.3, "Deleting a Fence Device"*.

From the cluster-specific page, you can configure fence devices for that cluster by clicking on **Fence Devices** along the top of the cluster display. This displays the fence devices for the cluster and displays the menu items for fence device configuration: **Add** and **Delete**. This is the starting point of each procedure described in the following sections.

Note

If this is an initial cluster configuration, no fence devices have been created, and therefore none are displayed.

Figure 3.5, "luci fence devices configuration page" shows the fence devices configuration screen before any fence devices have been created.

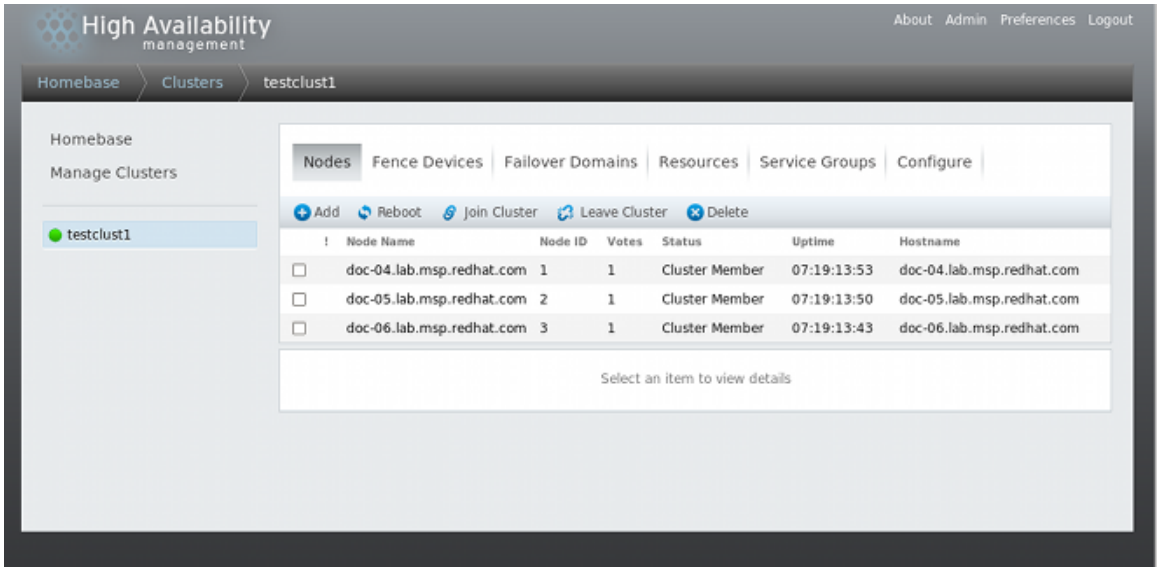

Figure 3.5. luci fence devices configuration page

3.6.1. Creating a Fence Device

To create a fence device, follow these steps:

1. From the **Fence Devices** configuration page, click **Add**. Clicking **Add** displays the **Add Fence Device (Instance)** dialog box. From this dialog box, select the type of fence device to configure.

2. Specify the information in the **Add Fence Device (Instance)** dialog box according to the type of fence device. Refer to *Appendix A, Fence Device Parameters* for more information about fence device parameters. In some cases you will need to specify additional node-specific parameters for the fence device when you configure fencing for the individual nodes, as described in *Section 3.7, "Configuring Fencing for Cluster Members"*.

3. Click **Submit**.

After the fence device has been added, it appears on the **Fence Devices** configuration page.

3.6.2. Modifying a Fence Device

To modify a fence device, follow these steps:

1. From the **Fence Devices** configuration page, click on the name of the fence device to modify. This displays the dialog box for that fence device, with the values that have been configured for the device.

2. To modify the fence device, enter changes to the parameters displayed. Refer to *Appendix A, Fence Device Parameters* for more information.

3. Click **Apply** and wait for the configuration to be updated.

3.6.3. Deleting a Fence Device

Note

Fence devices that are in use cannot be deleted. To delete a fence device that a node is currently using, first update the node fence configuration for any node using the device and then delete the device.

To delete a fence device, follow these steps:

1. From the **Fence Devices** configuration page, check the box to the left of the fence device or devices to select the devices to delete.
2. Click **Delete** and wait for the configuration to be updated. A message appears indicating which devices are being deleted.

When the configuration has been updated, the deleted fence device no longer appears in the display.

3.7. Configuring Fencing for Cluster Members

Once you have completed the initial steps of creating a cluster and creating fence devices, you need to configure fencing for the cluster nodes. To configure fencing for the nodes after creating a new cluster and configuring the fencing devices for the cluster, follow the steps in this section. Note that you must configure fencing for each node in the cluster.

The following sections provide procedures for configuring a single fence device for a node, configuring a node with a backup fence device, and configuring a node with redundant power supplies:

- *Section 3.7.1, “Configuring a Single Fence Device for a Node”*
- *Section 3.7.2, “Configuring a Backup Fence Device”*
- *Section 3.7.3, “Configuring a Node with Redundant Power”*

3.7.1. Configuring a Single Fence Device for a Node

Use the following procedure to configure a node with a single fence device.

1. From the cluster-specific page, you can configure fencing for the nodes in the cluster by clicking on **Nodes** along the top of the cluster display. This displays the nodes that constitute the cluster. This is also the default page that appears when you click on the cluster name beneath **Manage Clusters** from the menu on the left side of the **luci Homebase** page.
2. Click on a node name. Clicking a link for a node causes a page to be displayed for that link showing how that node is configured.

   The node-specific page displays any services that are currently running on the node, as well as any failover domains of which this node is a member. You can modify an existing failover domain by clicking on its name. For information on configuring failover domains, see *Section 3.8, “Configuring a Failover Domain”*.
3. On the node-specific page, under **Fence Devices**, click **Add Fence Method**. This displays the **Add Fence Method to Node** dialog box.

4. Enter a **Method Name** for the fencing method that you are configuring for this node. This is an arbitrary name that will be used by Red Hat High Availability Add-On; it is not the same as the DNS name for the device.

5. Click **Submit**. This displays the node-specific screen that now displays the method you have just added under **Fence Devices**.

6. Configure a fence instance for this method by clicking the **Add Fence Instance** button that appears beneath the fence method. This displays the **Add Fence Device (Instance)** drop-down menu from which you can select a fence device you have previously configured, as described in *Section 3.6.1, "Creating a Fence Device"*.

7. Select a fence device for this method. If this fence device requires that you configure node-specific parameters, the display shows the parameters to configure. For information on fencing parameters, refer to *Appendix A, Fence Device Parameters*.

Note

For non-power fence methods (that is, SAN/storage fencing), **Unfencing** is selected by default on the node-specific parameters display. This ensures that a fenced node's access to storage is not re-enabled until the node has been rebooted. For information on unfencing a node, refer to the **fence_node**(8) man page.

8. Click **Submit**. This returns you to the node-specific screen with the fence method and fence instance displayed.

3.7.2. Configuring a Backup Fence Device

You can define multiple fencing methods for a node. If fencing fails using the first method, the system will attempt to fence the node using the second method, followed by any additional methods you have configured.

Use the following procedure to configure a backup fence device for a node.

1. Use the procedure provided in *Section 3.7.1, "Configuring a Single Fence Device for a Node"* to configure the primary fencing method for a node.

2. Beneath the display of the primary method you defined, click **Add Fence Method**.

3. Enter a name for the backup fencing method that you are configuring for this node and click **Submit**. This displays the node-specific screen that now displays the method you have just added, below the primary fence method.

4. Configure a fence instance for this method by clicking **Add Fence Instance**. This displays a drop-down menu from which you can select a fence device you have previously configured, as described in *Section 3.6.1, "Creating a Fence Device"*.

5. Select a fence device for this method. If this fence device requires that you configure node-specific parameters, the display shows the parameters to configure. For information on fencing parameters, refer to *Appendix A, Fence Device Parameters*.

6. Click **Submit**. This returns you to the node-specific screen with the fence method and fence instance displayed.

You can continue to add fencing methods as needed. You can rearrange the order of fencing methods that will be used for this node by clicking on **Move Up** and **Move Down**.

3.7.3. Configuring a Node with Redundant Power

If your cluster is configured with redundant power supplies for your nodes, you must be sure to configure fencing so that your nodes fully shut down when they need to be fenced. If you configure each power supply as a separate fence method, each power supply will be fenced separately; the second power supply will allow the system to continue running when the first power supply is fenced and the system will not be fenced at all. To configure a system with dual power supplies, you must configure your fence devices so that both power supplies are shut off and the system is taken completely down. When configuring your system using **Conga**, this requires that you configure two instances within a single fencing method.

To configure fencing for a node with dual power supplies, follow the steps in this section.

1. Before you can configure fencing for a node with redundant power, you must configure each of the power switches as a fence device for the cluster. For information on configuring fence devices, see *Section 3.6, "Configuring Fence Devices"*.

2. From the cluster-specific page, click on **Nodes** along the top of the cluster display. This displays the nodes that constitute the cluster. This is also the default page that appears when you click on the cluster name beneath **Manage Clusters** from the menu on the left side of the **luci Homebase** page.

3. Click on a node name. Clicking a link for a node causes a page to be displayed for that link showing how that node is configured.

4. On the node-specific page, click **Add Fence Method**.

5. Enter a name for the fencing method that you are configuring for this node.

6. Click **Submit**. This displays the node-specific screen that now displays the method you have just added under **Fence Devices**.

7. Configure the first power supply as a fence instance for this method by clicking **Add Fence Instance**. This displays a drop-down menu from which you can select one of the power fencing devices you have previously configured, as described in *Section 3.6.1, "Creating a Fence Device"*.

8. Select one of the power fence devices for this method and enter the appropriate parameters for this device.

9. Click **Submit**. This returns you to the node-specific screen with the fence method and fence instance displayed.

10. Under the same fence method for which you have configured the first power fencing device, click **Add Fence Instance**. This displays a drop-down menu from which you can select the second power fencing devices you have previously configured, as described in *Section 3.6.1, "Creating a Fence Device"*.

11. Select the second of the power fence devices for this method and enter the appropriate parameters for this device.

12. Click **Submit**. This returns you to the node-specific screen with the fence methods and fence instances displayed, showing that each device will power the system off in sequence and power the system on in sequence. This is shown in *Figure 3.6, "Dual-Power Fencing Configuration"*.

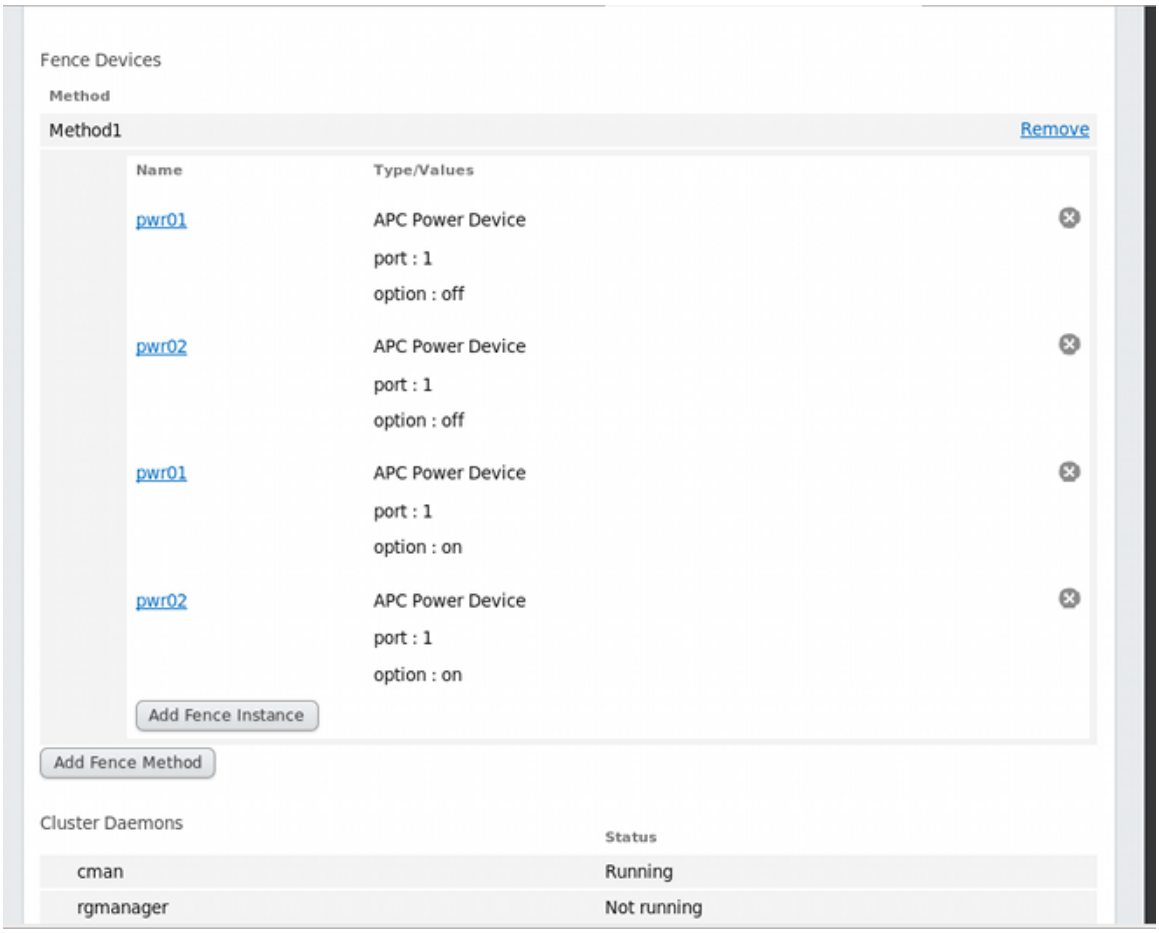

Figure 3.6. Dual-Power Fencing Configuration

3.8. Configuring a Failover Domain

A failover domain is a named subset of cluster nodes that are eligible to run a cluster service in the event of a node failure. A failover domain can have the following characteristics:

- Unrestricted — Allows you to specify that a subset of members are preferred, but that a cluster service assigned to this domain can run on any available member.
- Restricted — Allows you to restrict the members that can run a particular cluster service. If none of the members in a restricted failover domain are available, the cluster service cannot be started (either manually or by the cluster software).
- Unordered — When a cluster service is assigned to an unordered failover domain, the member on which the cluster service runs is chosen from the available failover domain members with no priority ordering.
- Ordered — Allows you to specify a preference order among the members of a failover domain. The member at the top of the list is the most preferred, followed by the second member in the list, and so on.
- Failback — Allows you to specify whether a service in the failover domain should fail back to the node that it was originally running on before that node failed. Configuring this characteristic is useful

in circumstances where a node repeatedly fails and is part of an ordered failover domain. In that circumstance, if a node is the preferred node in a failover domain, it is possible for a service to fail over and fail back repeatedly between the preferred node and another node, causing severe impact on performance.

Note

The failback characteristic is applicable only if ordered failover is configured.

Note

Changing a failover domain configuration has no effect on currently running services.

Note

Failover domains are *not* required for operation.

By default, failover domains are unrestricted and unordered.

In a cluster with several members, using a restricted failover domain can minimize the work to set up the cluster to run a cluster service (such as **`httpd`**), which requires you to set up the configuration identically on all members that run the cluster service. Instead of setting up the entire cluster to run the cluster service, you can set up only the members in the restricted failover domain that you associate with the cluster service.

Note

To configure a preferred member, you can create an unrestricted failover domain comprising only one cluster member. Doing that causes a cluster service to run on that cluster member primarily (the preferred member), but allows the cluster service to fail over to any of the other members.

The following sections describe adding, modifying, and deleting a failover domain:

- *Section 3.8.1, “Adding a Failover Domain”*
- *Section 3.8.2, “Modifying a Failover Domain”*
- *Section 3.8.3, “Deleting a Failover Domain”*

3.8.1. Adding a Failover Domain

To add a failover domain, follow the steps in this section.

1. From the cluster-specific page, you can configure failover domains for that cluster by clicking on **Failover Domains** along the top of the cluster display. This displays the failover domains that have been configured for this cluster.

2. Click **Add**. Clicking **Add** causes the display of the **Add Failover Domain to Cluster** dialog box, as shown in *Figure 3.7, "luci failover domain configuration dialog box"*.

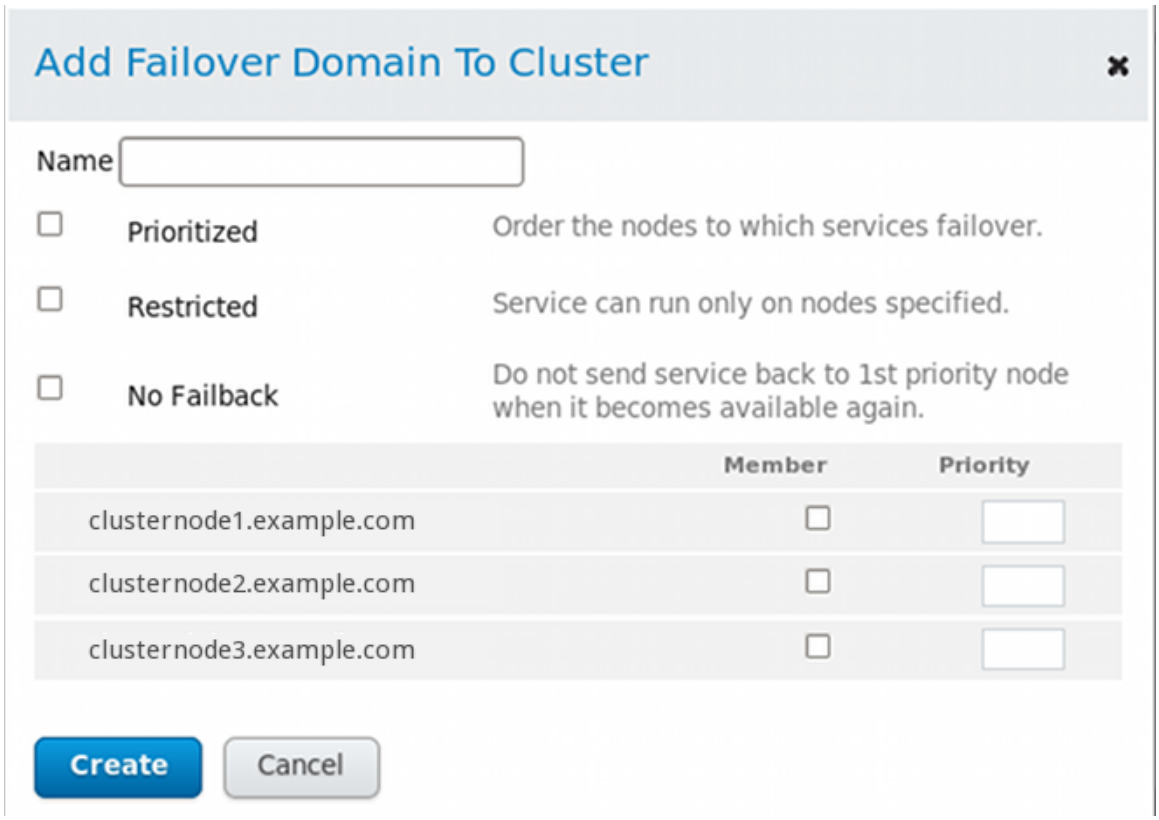

Figure 3.7. luci failover domain configuration dialog box

3. In the **Add Failover Domain to Cluster** dialog box, specify a failover domain name at the **Name** text box.

Note

The name should be descriptive enough to distinguish its purpose relative to other names used in your cluster.

4. To enable setting failover priority of the members in the failover domain, click the **Prioritized** checkbox. With **Prioritized** checked, you can set the priority value, **Priority**, for each node selected as members of the failover domain.

5. To restrict failover to members in this failover domain, click the **Restricted** checkbox. With **Restricted** checked, services assigned to this failover domain fail over only to nodes in this failover domain.

6. To specify that a node does not fail back in this failover domain, click the **No Failback** checkbox. With **No Failback** checked, if a service fails over from a preferred node, the service does not fail back to the original node once it has recovered.

7. Configure members for this failover domain. Click the **Member** checkbox for each node that is to be a member of the failover domain. If **Prioritized** is checked, set the priority in the **Priority** text box for each member of the failover domain.

8. Click **Create**. This displays the **Failover Domains** page with the newly-created failover domain displayed. A message indicates that the new domain is being created. Refresh the page for an updated status.

3.8.2. Modifying a Failover Domain

To modify a failover domain, follow the steps in this section.

1. From the cluster-specific page, you can configure Failover Domains for that cluster by clicking on **Failover Domains** along the top of the cluster display. This displays the failover domains that have been configured for this cluster.

2. Click on the name of a failover domain. This displays the configuration page for that failover domain.

3. To modify the **Prioritized**, **Restricted**, or **No Failback** properties for the failover domain, click or unclick the checkbox next to the property and click **Update Properties**.

4. To modify the failover domain membership, click or unclick the checkbox next to the cluster member. If the failover domain is prioritized, you can also modify the priority setting for the cluster member. Click **Update Settings**.

3.8.3. Deleting a Failover Domain

To delete a failover domain, follow the steps in this section.

1. From the cluster-specific page, you can configure Failover Domains for that cluster by clicking on **Failover Domains** along the top of the cluster display. This displays the failover domains that have been configured for this cluster.

2. Select the checkbox for the failover domain to delete.

3. Click on **Delete**.

3.9. Configuring Global Cluster Resources

You can configure global resources that can be used by any service running in the cluster, and you can configure resources that are available only to a specific service.

To add a global cluster resource, follow the steps in this section. You can add a resource that is local to a particular service when you configure the service, as described in *Section 3.10, "Adding a Cluster Service to the Cluster"*.

1. From the cluster-specific page, you can add resources to that cluster by clicking on **Resources** along the top of the cluster display. This displays the resources that have been configured for that cluster.

2. Click **Add**. This displays the **Add Resource to Cluster** drop-down menu.

3. Click the drop-down box under **Add Resource to Cluster** and select the type of resource to configure.
4. Enter the resource parameters for the resource you are adding. *Appendix B, HA Resource Parameters* describes resource parameters.
5. Click **Submit**. Clicking **Submit** returns to the resources page that displays the display of **Resources**, which displays the added resource (and other resources).

To modify an existing resource, perform the following steps.

1. From the **luci Resources** page, click on the name of the resource to modify. This displays the parameters for that resource.
2. Edit the resource parameters.
3. Click **Apply**.

To delete an existing resource, perform the following steps.

1. From the **luci Resources** page, click the checkbox for any resources to delete.
2. Click **Delete**.

3.10. Adding a Cluster Service to the Cluster

To add a cluster service to the cluster, follow the steps in this section.

1. From the cluster-specific page, you can add services to that cluster by clicking on **Service Groups** along the top of the cluster display. This displays the services that have been configured for that cluster. (From the **Service Groups** page, you can also start, restart, and disable a service, as described in *Section 4.4, "Managing High-Availability Services"*.)
2. Click **Add**. This displays the **Add Service to Cluster** dialog box.
3. On the **Add Service to Cluster** dialog box, at the **Service Name** text box, type the name of the service.

Note

Use a descriptive name that clearly distinguishes the service from other services in the cluster.

4. Check the **Automatically Start This Service** checkbox if you want the service to start automatically when a cluster is started and running. If the checkbox is *not* checked, the service must be started manually any time the cluster comes up from the stopped state.
5. Check the **Run Exclusive** checkbox to set a policy wherein the service only runs on nodes that have no other services running on them.

6. If you have configured failover domains for the cluster, you can use the drop-down menu of the **Failover domain** parameter to select a failover domain for this service. For information on configuring failover domains, see *Section 3.8, "Configuring a Failover Domain"*.

7. Use the **Recovery Policy** drop-down box to select a recovery policy for the service. The options are to **Relocate**, **Restart**, **Restart-Disable**, or **Disable** the service.

   Selecting the **Restart** option indicates that the system should attempt to restart the failed service before relocating the service. Selecting the **Restart-Disable** option indicates that the system should attempt to restart the service in place if it fails, but if restarting the service fails the service will be disabled instead of being moved to another host in the cluster.

   If you select **Restart** or **Restart-Disable** as the recovery policy for the service, you can specify the maximum number of restart failures before relocating or disabling the service, and you can specify the length of time in seconds after which to forget a restart.

8. To add a resource to the service, click **Add resource**. Clicking **Add resource** causes the display of the **Add Resource To Service** drop-down box that allows you to add an existing global resource or to add a new resource that is available *only* to this service.

   - To add an existing global resource, click on the name of the existing resource from the **Add Resource To Service** drop-down box. This displays the resource and its parameters on the **Service Groups** page for the service you are configuring. For information on adding or modifying global resources, see *Section 3.9, "Configuring Global Cluster Resources"*).

   - To add a new resource that is available only to this service, select the type of resource to configure from the **Add Resource To Service** drop-down box and enter the resource parameters for the resource you are adding. *Appendix B, HA Resource Parameters* describes resource parameters.

   - When adding a resource to a service, whether it is an existing global resource or a resource available only to this service, you can specify whether the resource is an **Independent Subtree** or a **Non-Critical Resource**.

     If you specify that a resource is an independent subtree, then if that resource fails only that resource is restarted (rather than the entire service) before the system attempting normal recovery. You can specify the maximum number of restarts to attempt for that resource on a node before implementing the recovery policy for the service. You can also specify the length of time in seconds after which the system will implement the recovery policy for the service.

     If you specify that the resource is a non-critical resource, then if that resource fails only that resource is restarted, and if the resource continues to fail then only that resource is disabled, rather than the entire service. You can specify the maximum number of restarts to attempt for that resource on a node before disabling that resource. You can also specify the length of time in seconds after which the system will disable that resource.

9. If you want to add child resources to the resource you are defining, click **Add a child resource**. Clicking **Add a child resource** causes the display of the **Add Resource To Service** drop-down box, from which you can add an existing global resource or add a new resource that is available only to this service. You can continue adding children resources to the resource to suit your requirements.

Note

If you are adding a Samba-service resource, add it directly to the service, *not* as a child of another resource.

10. When you have completed adding resources to the service, and have completed adding children resources to resources, click **Submit**. Clicking **Submit** returns to the **Service Groups** page displaying the added service (and other services).

Note

To verify the existence of the IP service resource used in a cluster service, you use the **/sbin/ip addr list** command on a cluster node. The following output shows the **/sbin/ip addr list** command executed on a node running a cluster service:

```
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1356 qdisc pfifo_fast qlen 1000
    link/ether 00:05:5d:9a:d8:91 brd ff:ff:ff:ff:ff:ff
    inet 10.11.4.31/22 brd 10.11.7.255 scope global eth0
    inet6 fe80::205:5dff:fe9a:d891/64 scope link
    inet 10.11.4.240/22 scope global secondary eth0
       valid_lft forever preferred_lft forever
```

To modify an existing service, perform the following steps.

1. From the **Service Groups** dialog box, click on the name of the service to modify. This displays the parameters and resources that have been configured for that service.
2. Edit the service parameters.
3. Click **Submit**.

To delete an existing service, perform the following steps.

1. From the **luci Service Groups** page, click the checkbox for any services to delete.
2. Click **Delete**.

Managing Red Hat High Availability Add-On With Conga

This chapter describes various administrative tasks for managing Red Hat High Availability Add-On and consists of the following sections:

- *Section 4.1, “Adding an Existing Cluster to the luci Interface”*
- *Section 4.2, “Managing Cluster Nodes”*
- *Section 4.3, “Starting, Stopping, Restarting, and Deleting Clusters”*
- *Section 4.4, “Managing High-Availability Services”*

4.1. Adding an Existing Cluster to the luci Interface

If you have previously created a High Availability Add-On cluster you can easily add the cluster to the **luci** interface so that you can manage the cluster with **Conga**.

To add an existing cluster to the **luci** interface, follow these steps:

1. Click **Manage Clusters** from the menu on the left side of the **luci Homebase** page. The **Clusters** screen appears.
2. Click **Add**. The **Add Existing Cluster** screen appears.
3. Enter the node hostname and **ricci** password for any of the nodes in the existing cluster. Since each node in the cluster contains all of the configuration information for the cluster, this should provide enough information to add the cluster to the **luci** interface.
4. Click **Connect**. The **Add Existing Cluster** screen then displays the cluster name and the remaining nodes in the cluster.
5. Enter the individual **ricci** passwords for each node in the cluster, or enter one password and select **Use same password for all nodes**.
6. Click **Add Cluster**. The previously-configured cluster now displays on the **Manage Clusters** screen.

4.2. Managing Cluster Nodes

This section documents how to perform the following node-management functions through the **luci** server component of **Conga**:

- *Section 4.2.1, “Rebooting a Cluster Node”*
- *Section 4.2.2, “Causing a Node to Leave or Join a Cluster”*
- *Section 4.2.3, “Adding a Member to a Running Cluster”*
- *Section 4.2.4, “Deleting a Member from a Cluster”*

4.2.1. Rebooting a Cluster Node

To reboot a node in a cluster, perform the following steps:

1. From the cluster-specific page, click on **Nodes** along the top of the cluster display. This displays the nodes that constitute the cluster. This is also the default page that appears when you click on the cluster name beneath **Manage Clusters** from the menu on the left side of the **luci Homebase** page.
2. Select the node to reboot by clicking the checkbox for that node.
3. Select the **Reboot** function from the menu at the top of the page. This causes the selected node to reboot and a message appears at the top of the page indicating that the node is being rebooted.
4. Refresh the page to see the updated status of the node.

It is also possible to reboot more than one node at a time by selecting all of the nodes that you wish to reboot before clicking on **Reboot**.

4.2.2. Causing a Node to Leave or Join a Cluster

You can use the **luci** server component of **Conga** to cause a node to leave an active cluster by stopping all cluster services on the node. You can also use the **luci** server component of **Conga** to cause a node that has left a cluster to rejoin the cluster.

Causing a node to leave a cluster does not remove the cluster configuration information from that node, and the node still appears in the cluster node display with a status of **`Not a cluster member`**. For information on deleting the node entirely from the cluster configuration, see *Section 4.2.4, “Deleting a Member from a Cluster”*.

To cause a node to leave a cluster, perform the following steps. This shuts down the cluster software in the node. Making a node leave a cluster prevents the node from automatically joining the cluster when it is rebooted.

1. From the cluster-specific page, click on **Nodes** along the top of the cluster display. This displays the nodes that constitute the cluster. This is also the default page that appears when you click on the cluster name beneath **Manage Clusters** from the menu on the left side of the **luci Homebase** page.
2. Select the node you want to leave the cluster by clicking the checkbox for that node.
3. Select the **Leave Cluster** function from the menu at the top of the page. This causes a message to appear at the top of the page indicating that the node is being stopped.
4. Refresh the page to see the updated status of the node.

It is also possible to cause more than one node at a time to leave the cluster by selecting all of the nodes to leave the cluster before clicking on **Leave Cluster**.

To cause a node to rejoin a cluster, select any nodes you want to have rejoin the cluster by clicking the checkbox for those nodes and selecting **Join Cluster**. This makes the selected nodes join the cluster, and allows the selected nodes to join the cluster when they are rebooted.

4.2.3. Adding a Member to a Running Cluster

To add a member to a running cluster, follow the steps in this section.

1. From the cluster-specific page, click **Nodes** along the top of the cluster display. This displays the nodes that constitute the cluster. This is also the default page that appears when you click on the cluster name beneath **Manage Clusters** from the menu on the left side of the **luci Homebase** page.

2. Click **Add**. Clicking **Add** causes the display of the **Add Nodes To Cluster** dialog box.

3. Enter the node name in the **Node Hostname** text box; enter the **ricci** password in the **Password** text box. If you are using a different port for the **ricci** agent than the default of 11111, change this parameter to the port you are using.

4. Check the **Enable Shared Storage Support** checkbox if clustered storage is required to download the packages that support clustered storage and enable clustered LVM; you should select this only when you have access to the Resilient Storage Add-On or the Scalable File System Add-On.

5. If you want to add more nodes, click **Add Another Node** and enter the node name and password for the each additional node.

6. Click **Add Nodes**. Clicking **Add Nodes** causes the following actions:

   a. If you have selected **Download Packages**, the cluster software packages are downloaded onto the nodes.

   b. Cluster software is installed onto the nodes (or it is verified that the appropriate software packages are installed).

   c. The cluster configuration file is updated and propagated to each node in the cluster — including the added node.

   d. The added node joins the cluster.

   The **Nodes** page appears with a message indicating that the node is being added to the cluster. Refresh the page to update the status.

7. When the process of adding a node is complete, click on the node name for the newly-added node to configure fencing for this node, as described in *Section 3.6, “Configuring Fence Devices”*.

4.2.4. Deleting a Member from a Cluster

To delete a member from an existing cluster that is currently in operation, follow the steps in this section. Note that nodes must be stopped before being deleted unless you are deleting all of the nodes in the cluster at once.

1. From the cluster-specific page, click **Nodes** along the top of the cluster display. This displays the nodes that constitute the cluster. This is also the default page that appears when you click on the cluster name beneath **Manage Clusters** from the menu on the left side of the **luci Homebase** page.

Note

To allow services running on a node to fail over when the node is deleted, skip the next step.

2. Disable or relocate each service that is running on the node to be deleted. For information on disabling and relocating services, see *Section 4.4, "Managing High-Availability Services"*.
3. Select the node or nodes to delete.
4. Click **Delete**. The **Nodes** page indicates that the node is being removed. Refresh the page to see the current status.

Important

Removing a cluster node from the cluster is a destructive operation that cannot be undone.

4.3. Starting, Stopping, Restarting, and Deleting Clusters

You can start, stop, and restart a cluster by performing these actions on the individual nodes in the cluster. From the cluster-specific page, click on **Nodes** along the top of the cluster display. This displays the nodes that constitute the cluster.

To stop a cluster, perform the following steps. This shuts down the cluster software in the nodes, but does not remove the cluster configuration information from the nodes and the nodes still appear in the cluster node display with a status of **`Not a cluster member`**.

1. Select all of the nodes in the cluster by clicking on the checkbox next to each node.
2. Select the **Leave Cluster** function from the menu at the top of the page. This causes a message to appear at the top of the page indicating that each node is being stopped.
3. Refresh the page to see the updated status of the nodes.

To start a cluster, perform the following steps:

1. Select all of the nodes in the cluster by clicking on the checkbox next to each node.
2. Select the **Join Cluster** function from the menu at the top of the page.
3. Refresh the page to see the updated status of the nodes.

To restart a running cluster, first stop all of the nodes in cluster, then start all of the nodes in the cluster, as described above.

To delete a cluster entirely from the **luci** interface, perform the following steps. This removes the cluster configuration information from the nodes themselves as well as removing them from the cluster display.

Important

Deleting a cluster is a destructive operation that cannot be undone. To restore a cluster after you have deleted it requires that you recreate and redefine the cluster from scratch.

1. Select all of the nodes in the cluster by clicking on the checkbox next to each node.
2. Select the **Delete** function from the menu at the top of the page.

4.4. Managing High-Availability Services

In addition to adding and modifying a service, as described in *Section 3.10, "Adding a Cluster Service to the Cluster"*, you can perform the following management functions for high-availability services through the **luci** server component of **Conga**:

- Start a service
- Restart a service
- Disable a service
- Delete a service
- Relocate a service

From the cluster-specific page, you can manage services for that cluster by clicking on **Service Groups** along the top of the cluster display. This displays the services that have been configured for that cluster.

- **Starting a service** — To start any services that are not currently running, select any services you want to start by clicking the checkbox for that service and clicking **Start**.
- **Restarting a service** — To restart any services that are currently running, select any services you want to restart by clicking the checkbox for that service and clicking **Restart**.
- **Disabling a service** — To disable any service that is currently running, select any services you want to disable by clicking the checkbox for that service and clicking **Disable**.
- **Deleting a service** — To delete any services that are not currently running, select any services you want to disable by clicking the checkbox for that service and clicking **Delete**.
- **Relocating a service** — To relocate a running service, click on the name of the service in the services display. This causes the services configuration page for the service to be displayed, with a display indicating on which node the service is currently running.

  From the **Start on node...** drop-down box, select the node on which you want to relocate the service, and click on the **Start** icon. A message appears at the top of the screen indicating that the service is being started. You may need to refresh the screen to see the new display indicating that the service is running on the node you have selected.

Note

You can also start, restart, disable or delete an individual service by clicking on the name of the service on the **Services** page. This displays the service configuration page. At the top right corner of the service configuration page are the same icons for **Start**, **Restart**, **Disable**, and **Delete**.

Configuring Red Hat High Availability Add-On With the ccs Command

As of the Red Hat Enterprise Linux 6.1 release and later, the Red Hat High Availability Add-On provides support for the **ccs** cluster configuration command. The **ccs** command allows an administrator to create, modify and view the **cluster.conf** cluster configuration file. You can use the **ccs** command to configure a cluster configuration file on a local file system or on a remote node. Using the **ccs** command, an administrator can also start and stop the cluster services on one or all of the nodes in a configured cluster.

This chapter describes how to configure the Red Hat High Availability Add-On cluster configuration file using the **ccs** command. For information on using the **ccs** command to manage a running cluster, see *Chapter 6, Managing Red Hat High Availability Add-On With **ccs***.

This chapter consists of the following sections:

- *Section 5.1, “Operational Overview”*
- *Section 5.2, “Configuration Tasks”*
- *Section 5.3, “Starting **ricci**”*
- *Section 5.4, “Creating A Cluster”*
- *Section 5.5, “Configuring Fence Devices”*
- *Section 5.7, “Configuring Fencing for Cluster Members”*
- *Section 5.8, “Configuring a Failover Domain”*
- *Section 5.9, “Configuring Global Cluster Resources”*
- *Section 5.10, “Adding a Cluster Service to the Cluster”*
- *Section 5.13, “Configuring a Quorum Disk”*
- *Section 5.14, “Miscellaneous Cluster Configuration”*
- *Section 5.15, “Propagating the Configuration File to the Cluster Nodes”*

Important

Make sure that your deployment of High Availability Add-On meets your needs and can be supported. Consult with an authorized Red Hat representative to verify your configuration prior to deployment. In addition, allow time for a configuration burn-in period to test failure modes.

Important

This chapter references commonly used **cluster.conf** elements and attributes. For a comprehensive list and description of **cluster.conf** elements and attributes, refer to the cluster schema at **/usr/share/cluster/cluster.rng**, and the annotated schema at **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** (for example **/usr/share/doc/cman-3.0.12/cluster_conf.html**).

5.1. Operational Overview

This section describes the following general operational aspects of using the **ccs** command to configure a cluster:

- *Section 5.1.1, "Creating the Cluster Configuration File on a Local System"*
- *Section 5.1.2, "Viewing the Current Cluster Configuration"*
- *Section 5.1.3, "Specifying ricci Passwords with the ccs Command"*
- *Section 5.1.4, "Modifying Cluster Configuration Components"*

5.1.1. Creating the Cluster Configuration File on a Local System

Using the **ccs** command, you can create a cluster configuration file on a cluster node, or you can create a cluster configuration file on a local file system and then send that file to a host in a cluster. This allows you to work on a file from a local machine, where you can maintain it under version control or otherwise tag the file according to your needs. Using the **ccs** command does not require root privilege.

When you create and edit a cluster configuration file on a cluster node with the **ccs** command, you use the **-h** option to specify the name of the host. This creates and edits the **cluster.conf** file on the host:

```
ccs -h host [options]
```

To create and edit a cluster configuration file on a local system, use the **-f** option of the **ccs** command to specify the name of the configuration file when you perform a cluster operation. You can name this file anything you want.

```
ccs -f file [options]
```

After you have created the file locally you can send it to a cluster node using the **--setconf** option of the **ccs** command. On a host machine in a cluster, the file you send will be named **cluster.conf** and it will be placed in the **/etc/cluster** directory.

```
ccs -h host -f file --setconf
```

For information on using the **--setconf** option of the **ccs** command, see *Section 5.15, "Propagating the Configuration File to the Cluster Nodes"*.

5.1.2. Viewing the Current Cluster Configuration

This chapter describes how to create a cluster configuration file. If at any time you want to print the current file for a cluster, use the following command, specifying a node in the cluster as the host:

```
ccs -h host --getconf
```

If you are creating your cluster configuration file on a local system you can specify the **-f** option instead of the **-h** option, as described in *Section 5.1.1, "Creating the Cluster Configuration File on a Local System"*.

5.1.3. Specifying ricci Passwords with the ccs Command

Executing **ccs** commands that distribute copies of the **cluster.conf** file to the nodes of a cluster requires that **ricci** be installed and running on the cluster nodes, as described in *Section 2.14, "Considerations for **ricci**"*. Using **ricci** requires a password the first time you interact with **ricci** from any specific machine.

If you have not entered a password for an instance of **ricci** on a particular machine from the machine you are using, you will be prompted for that password when the **ccs** command requires it. Alternately, you can use the **-p** option to specify a **ricci** password on the command line.

```
ccs -h host -p password --sync --activate
```

When you propagate the **cluster.conf** file to all of the nodes in the cluster with the **--sync** option of the **ccs** command and you specify a **ricci** password for the command, the **ccs** command will use that password for each node in the cluster. If you need to set different passwords for **ricci** on individual nodes, you can use the **--setconf** option with the **-p** option to distribute the configuration file to one node at a time.

5.1.4. Modifying Cluster Configuration Components

You use the **ccs** command to configure cluster components and their attributes in the cluster configuration file. After you have added a cluster component to the file, in order to modify the attributes of that component you must remove the component you have defined and add the component again, with the modified attributes. Information on how to do this with each component is provided in the individual sections of this chapter.

The attributes of the **cman** cluster component provide an exception to this procedure for modifying cluster components. To modify these attributes, you execute the **--setcman** option of the **ccs** command, specifying the new attributes.

5.2. Configuration Tasks

Configuring Red Hat High Availability Add-On software with the **ccs** consists of the following steps:

1. Ensuring that **ricci** is running on all nodes in the cluster. Refer to *Section 5.3, "Starting **ricci**"*.
2. Creating a cluster. Refer to *Section 5.4, "Creating A Cluster"*.
3. Configuring fence devices. Refer to *Section 5.5, "Configuring Fence Devices"*.
4. Configuring fencing for cluster members. Refer to *Section 5.7, "Configuring Fencing for Cluster Members"*.

5. Creating failover domains. Refer to *Section 5.8, "Configuring a Failover Domain"*.
6. Creating resources. Refer to *Section 5.9, "Configuring Global Cluster Resources"*.
7. Creating cluster services. Refer to *Section 5.10, "Adding a Cluster Service to the Cluster"*.
8. Configuring a quorum disk, if necessary. Refer to *Section 5.13, "Configuring a Quorum Disk"*.
9. Configuring global cluster properties. Refer to *Section 5.14, "Miscellaneous Cluster Configuration"*.
10. Propagating the cluster configuration file to all of the cluster nodes. Refer to *Section 5.15, "Propagating the Configuration File to the Cluster Nodes"*.

5.3. Starting ricci

In order to create and distribute cluster configuration files on the nodes of the cluster, the **ricci** service must be running on each node. Before starting **ricci**, you should ensure that you have configured your system as follows:

1. The IP ports on your cluster nodes should be enabled for **ricci**. For information on enabling IP ports on cluster nodes, see *Section 2.3.1, "Enabling IP Ports on Cluster Nodes"*.
2. The **ricci** service is installed on all nodes in the cluster and assigned a **ricci** password, as described in *Section 2.14, "Considerations for **ricci**"*.

After **ricci** has been installed and configured on each node, start the **ricci** service on each node:

```
# service ricci start
Starting ricci:                                            [  OK  ]
```

5.4. Creating A Cluster

This section describes how to create, modify, and delete a skeleton cluster configuration with the **ccs** command without fencing, failover domains, and HA services. Subsequent sections describe how to configure those parts of the configuration.

To create a skeleton cluster configuration file, first create and name the cluster and then add the nodes to the cluster, as in the following procedure:

1. Create a cluster configuration file on one of the nodes in the cluster by executing the **ccs** command using the **-h** parameter to specify the node on which to create the file and the **createcluster** option to specify a name for the cluster:

   ```
   ccs -h host --createcluster clustername
   ```

   For example, the following command creates a configuration file on **node-01.example.com** named **mycluster**:

   ```
   ccs -h node-01.example.com --createcluster mycluster
   ```

   The cluster name cannot exceed 15 characters.

If a **cluster.conf** file already exists on the host that you specify, executing this command will replace that existing file.

If you want to create a cluster configuration file on your local system you can specify the **-f** option instead of the **-h** option. For information on creating the file locally, see *Section 5.1.1, "Creating the Cluster Configuration File on a Local System"*.

2. To configure the nodes that the cluster contains, execute the following command for each node in the cluster:

```
ccs -h host --addnode node
```

For example, the following three commands add the nodes **node-01.example.com**, **node-02.example.com**, and **node-03.example.com** to the configuration file on **node-01.example.com**:

```
ccs -h node-01.example.com --addnode node-01.example.com
ccs -h node-01.example.com --addnode node-02.example.com
ccs -h node-01.example.com --addnode node-03.example.com
```

To view a list of the nodes that have been configured for a cluster, execute the following command:

```
ccs -h host --lsnodes
```

*Example 5.1, "**cluster.conf** File After Adding Three Nodes"* shows a **cluster.conf** configuration file after you have created the cluster **mycluster** that contains the nodes **node-01.example.com**, **node-02.example.com**, and **node-03.example.com**.

Example 5.1. **cluster.conf** File After Adding Three Nodes

```
<cluster name="mycluster" config_version="2">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
   </fencedevices>
   <rm>
   </rm>
</cluster>
```

When you add a node to the cluster, you can specify the number of votes the node contributes to determine whether there is a quorum. To set the number of votes for a cluster node, use the following command:

```
ccs -h host --addnode host --votes votes
```

When you add a node, the **ccs** assigns the node a unique integer that is used as the node identifier. If you want to specify the node identifier manually when creating a node, use the following command:

```
ccs -h host --addnode host --nodeide nodeid
```

To remove a node from a cluster, execute the following command:

```
ccs -h host --rmnode node
```

When you have finished configuring all of the components of your cluster, you will need to sync the cluster configuration file to all of the nodes, as described in *Section 5.15, "Propagating the Configuration File to the Cluster Nodes"*.

5.5. Configuring Fence Devices

Configuring fence devices consists of creating, updating, and deleting fence devices for the cluster. You must create and name the fence devices in a cluster before you can configure fencing for the nodes in the cluster. For information on configuring fencing for the individual nodes in the cluster, see *Section 5.7, "Configuring Fencing for Cluster Members"*.

Before configuring your fence devices, you may want to modify some of the fence daemon properties for your system from the default values. The values you configure for the fence daemon are general values for the cluster. The general fencing properties for the cluster you may want to modify are summarized as follows:

- The **post_fail_delay** attribute is the number of seconds the fence daemon (**fenced**) waits before fencing a node (a member of the fence domain) after the node has failed. The **post_fail_delay** default value is **0**. Its value may be varied to suit cluster and network performance.

  To configure a value for the **post_fail_delay** attribute, execute the following command:

  ```
  ccs -h host --setfencedaemon post_fail_delay=value
  ```

- The **post-join_delay** attribute is the number of seconds the fence daemon (**fenced**) waits before fencing a node after the node joins the fence domain. The **post_join_delay** default value is **3**. A typical setting for **post_join_delay** is between 20 and 30 seconds, but can vary according to cluster and network performance.

  To configure a value for the **post_join** attribute, execute the following command:

  ```
  ```

```
ccs -h host --setfencedaemon post_join_delay=value
```

Note

For more information about the **post_join_delay** and **post_fail_delay** attributes as well as the additional fence daemon properties you can modify, refer to the fenced(8) man page and refer to the cluster schema at **/usr/share/cluster/cluster.rng**, and the annotated schema at **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html**.

To configure a fence device for a cluster, execute the following command:

```
ccs -h host --addfencedev devicename [fencedeviceoptions]
```

For example, to configure an APC fence device in the configuration file on the cluster node **node1** named **myfence** with an IP address of **apc_ip_example**, a login of **login_example**, and a password of **password_example**, execute the following command:

```
ccs -h node1 --addfencedev myfence agent=fence_apc ipaddr=apc_ip_example login=login_example
 passwd=password_example
```

The following example shows the **fencedevices** section of the **cluster.conf** configuration file after you have added this APC fence device:

```
<fencedevices>
      <fencedevice agent="fence_apc" ipaddr="apc_ip_example" login="login_example"
 name="myfence" passwd="password_example"/>
</fencedevices>
```

When configuring fence devices for a cluster, you may find it useful to see a listing of available devices for your cluster and the options available for each device. You may also find it useful to see a listing of fence devices currently configured for your cluster. For information on using the **ccs** command to print a list of available fence devices and options or to print a list of fence devices currently configured for your cluster, refer to *Section 5.6, "Listing Fence Devices and Fence Device Options"*.

To remove a fence device from your cluster configuration, execute the following command:

```
ccs -h host --rmfencedev fence_device_name
```

For example, to remove a fence device that you have named **myfence** from the cluster configuration file on cluster node **node1**, execute the following command:

```
ccs -h node1 --rmfencedev myfence
```

If you need to modify the attributes of a fence device you have already configured, you must first remove that fence device then add it again with the modified attributes.

Note that when you have finished configuring all of the components of your cluster, you will need to sync the cluster configuration file to all of the nodes, as described in *Section 5.15, "Propagating the Configuration File to the Cluster Nodes"*.

5.6. Listing Fence Devices and Fence Device Options

You can use the **ccs** command to print a list of available fence devices and to print a list of options for each available fence type. You can also use the **ccs** command to print a list of fence devices currently configured for your cluster.

To print a list of fence devices currently available for your cluster, execute the following command:

```
ccs -h host --lsfenceopts
```

For example, the following command lists the fence devices available on the cluster node **node1**, showing sample output.

```
[root@ask-03 ~]# ccs -h node1 --lsfenceopts
fence_rps10 - RPS10 Serial Switch
fence_vixel - No description available
fence_egenera - No description available
fence_xcat - No description available
fence_na - Node Assassin
fence_apc - Fence agent for APC over telnet/ssh
fence_apc_snmp - Fence agent for APC over SNMP
fence_bladecenter - Fence agent for IBM BladeCenter
fence_bladecenter_snmp - Fence agent for IBM BladeCenter over SNMP
fence_cisco_mds - Fence agent for Cisco MDS
fence_cisco_ucs - Fence agent for Cisco UCS
fence_drac5 - Fence agent for Dell DRAC CMC/5
fence_eps - Fence agent for ePowerSwitch
fence_ibmblade - Fence agent for IBM BladeCenter over SNMP
fence_ifmib - Fence agent for IF MIB
fence_ilo - Fence agent for HP iLO
fence_ilo_mp - Fence agent for HP iLO MP
fence_intelmodular - Fence agent for Intel Modular
fence_ipmilan - Fence agent for IPMI over LAN
fence_kdump - Fence agent for use with kdump
fence_rhevm - Fence agent for RHEV-M REST API
fence_rsa - Fence agent for IBM RSA
fence_sanbox2 - Fence agent for QLogic SANBox2 FC switches
fence_scsi - fence agent for SCSI-3 persistent reservations
fence_virsh - Fence agent for virsh
fence_virt - Fence agent for virtual machines
fence_vmware - Fence agent for VMware
fence_vmware_soap - Fence agent for VMware over SOAP API
fence_wti - Fence agent for WTI
fence_xvm - Fence agent for virtual machines
```

To print a list of the options you can specify for a particular fence type, execute the following command:

```
ccs -h host --lsfenceopts fence_type
```

For example, the following command lists the fence options for the **fence_wti** fence agent.

```
[root@ask-03 ~]# ccs -h node1 --lsfenceopts fence_wti
fence_wti - Fence agent for WTI
  Required Options:
  Optional Options:
    option: No description available
    action: Fencing Action
    ipaddr: IP Address or Hostname
    login: Login Name
    passwd: Login password or passphrase
    passwd_script: Script to retrieve password
    cmd_prompt: Force command prompt
    secure: SSH connection
    identity_file: Identity file for ssh
    port: Physical plug number or name of virtual machine
    inet4_only: Forces agent to use IPv4 addresses only
    inet6_only: Forces agent to use IPv6 addresses only
    ipport: TCP port to use for connection with device
    verbose: Verbose mode
    debug: Write debug information to given file
    version: Display version information and exit
    help: Display help and exit
    separator: Separator for CSV created by operation list
    power_timeout: Test X seconds for status change after ON/OFF
    shell_timeout: Wait X seconds for cmd prompt after issuing command
    login_timeout: Wait X seconds for cmd prompt after login
    power_wait: Wait X seconds after issuing ON/OFF
    delay: Wait X seconds before fencing is started
    retry_on: Count of attempts to retry power on
```

To print a list of fence devices currently configured for your cluster, execute the following command:

```
ccs -h host --lsfencedev
```

5.7. Configuring Fencing for Cluster Members

Once you have completed the initial steps of creating a cluster and creating fence devices, you need to configure fencing for the cluster nodes. To configure fencing for the nodes after creating a new cluster and configuring the fencing devices for the cluster, follow the steps in this section. Note that you must configure fencing for each node in the cluster.

This section documents the following procedures:

- *Section 5.7.1, “Configuring a Single Power-Based Fence Device for a Node”*
- *Section 5.7.2, “Configuring a Single Storage-Based Fence Device for a Node”*
- *Section 5.7.3, “Configuring a Backup Fence Device”*
- *Section 5.7.4, “Configuring a Node with Redundant Power”*
- *Section 5.7.5, “Removing Fence Methods and Fence Instances”*

5.7.1. Configuring a Single Power-Based Fence Device for a Node

Use the following procedure to configure a node with a single power-based fence device that uses a fence device named **apc**, which uses the **fence_apc** fencing agent.

1. Add a fence method for the node, providing a name for the fence method.

```

```

```
ccs -h host --addmethod method node
```

For example, to configure a fence method named **APC** for the node **node-01.example.com** in the configuration file on the cluster node **node-01.example.com**, execute the following command:

```
ccs -h node01.example.com --addmethod APC node01.example.com
```

2. Add a fence instance for the method. You must specify the fence device to use for the node, the node this instance applies to, the name of the method, and any options for this method that are specific to this node:

```
ccs -h host --addfenceinst fencedevicename node method [options]
```

For example, to configure a fence instance in the configuration file on the cluster node **node-01.example.com** that uses the APC switch power port 1 on the fence device named **apc** to fence cluster node **node-01.example.com** using the method named **APC**, execute the following command:

```
ccs -h node01.example.com --addfenceinst apc node01.example.com APC port=1
```

You will need to add a fence method for each node in the cluster. The following commands configure a fence method for each node with the method name **APC**. The device for the fence method specifies **apc** as the device name, which is a device previously configured with the **--addfencedev** option, as described in *Section 5.5, "Configuring Fence Devices"*. Each node is configured with a unique APC switch power port number: The port number for **node-01.example.com** is **1**, the port number for **node-02.example.com** is **2**, and the port number for **node-03.example.com** is **3**.

```
ccs -h node01.example.com --addmethod APC node01.example.com
ccs -h node01.example.com --addmethod APC node02.example.com
ccs -h node01.example.com --addmethod APC node03.example.com
ccs -h node01.example.com --addfenceinst apc node01.example.com APC port=1
ccs -h node01.example.com --addfenceinst apc node02.example.com APC port=2
ccs -h node01.example.com --addfenceinst apc node03.example.com APC port=3
```

*Example 5.2, "**cluster.conf** After Adding Power-Based Fence Methods "* shows a **cluster.conf** configuration file after you have added these fencing methods and instances to each node in the cluster.

Example 5.2. **cluster.conf** After Adding Power-Based Fence Methods

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC">
              <device name="apc" port="1"/>
             </method>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
```

```
          <fence>
             <method name="APC">
               <device name="apc" port="2"/>
             </method>
          </fence>
      </clusternode>
      <clusternode name="node-03.example.com" nodeid="3">
          <fence>
             <method name="APC">
               <device name="apc" port="3"/>
             </method>
          </fence>
      </clusternode>
    </clusternodes>
    <fencedevices>
          <fencedevice agent="fence_apc" ipaddr="apc_ip_example" login="login_example"
 name="apc" passwd="password_example"/>
    </fencedevices>
    <rm>
    </rm>
</cluster>
```

Note that when you have finished configuring all of the components of your cluster, you will need to sync the cluster configuration file to all of the nodes, as described in *Section 5.15, "Propagating the Configuration File to the Cluster Nodes"*.

5.7.2. Configuring a Single Storage-Based Fence Device for a Node

When using non-power fencing methods (that is, SAN/storage fencing) to fence a node, you must configure *unfencing* for the fence device. This ensures that a fenced node is not re-enabled until the node has been rebooted. When you configure unfencing for a node, you specify a device that mirrors the corresponding fence device you have configured for the node with the notable addition of the explicit action of **on** or **enable**.

For more information about unfencing a node, refer to the **fence_node**(8) man page.

Use the following procedure to configure a node with a single storage-based fence device that uses a fence device named **sanswitch1**, which uses the **fence_sanbox2** fencing agent.

1. Add a fence method for the node, providing a name for the fence method.

   ```
   ccs -h host --addmethod method node
   ```

   For example, to configure a fence method named **SAN** for the node **node-01.example.com** in the configuration file on the cluster node **node-01.example.com**, execute the following command:

   ```
   ccs -h node01.example.com --addmethod SAN  node01.example.com
   ```

2. Add a fence instance for the method. You must specify the fence device to use for the node, the node this instance applies to, the name of the method, and any options for this method that are specific to this node:

   ```
   ```

```
ccs -h host --addfenceinst fencedevicename node method [options]
```

For example, to configure a fence instance in the configuration file on the cluster node **node-01.example.com** that uses the SAN switch power port 11 on the fence device named **sanswitch1** to fence cluster node **node-01.example.com** using the method named **SAN**, execute the following command:

```
ccs -h node01.example.com --addfenceinst sanswitch1 node01.example.com SAN port=11
```

3. To configure unfencing for the storage based fence device on this node, execute the following command:

```
ccs -h host --addunfence fencedevicename node action=on|off
```

You will need to add a fence method for each node in the cluster. The following commands configure a fence method for each node with the method name **SAN**. The device for the fence method specifies **sanswitch** as the device name, which is a device previously configured with the --addfencedev option, as described in *Section 5.5, "Configuring Fence Devices"*. Each node is configured with a unique SAN physical port number: The port number for **node-01.example.com** is **11**, the port number for **node-02.example.com** is **12**, and the port number for **node-03.example.com** is **13**.

```
ccs -h node01.example.com --addmethod SAN node01.example.com
ccs -h node01.example.com --addmethod SAN node02.example.com
ccs -h node01.example.com --addmethod SAN node03.example.com
ccs -h node01.example.com --addfenceinst sanswitch1 node01.example.com SAN port=11
ccs -h node01.example.com --addfenceinst sanswitch1 node02.example.com SAN port=12
ccs -h node01.example.com --addfenceinst sanswitch1 node03.example.com SAN port=13
ccs -h node01.example.com --addunfence sanswitch1 node01.example.com port=11 action=on
ccs -h node01.example.com --addunfence sanswitch1 node02.example.com port=12 action=on
ccs -h node01.example.com --addunfence sanswitch1 node03.example.com port=13 action=on
```

*Example 5.3, "**cluster.conf** After Adding Storage-Based Fence Methods "* shows a **cluster.conf** configuration file after you have added fencing methods, fencing instances, and unfencing to each node in the cluster.

Example 5.3. **cluster.conf** After Adding Storage-Based Fence Methods

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="SAN">
       <device name="sanswitch1" port="11"/>
            </method>
         </fence>
         <unfence>
             <device name="sanswitch1" port="11" action="on"/>
         </unfence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="SAN">
       <device name="sanswitch1" port="12"/>
            </method>
```

```
          </fence>
          <unfence>
              <device name="sanswitch1" port="12" action="on"/>
          </unfence>
      </clusternode>
      <clusternode name="node-03.example.com" nodeid="3">
          <fence>
             <method name="SAN">
        <device name="sanswitch1" port="13"/>
             </method>
          </fence>
          <unfence>
              <device name="sanswitch1" port="13" action="on"/>
          </unfence>
      </clusternode>
   </clusternodes>
   <fencedevices>
        <fencedevice agent="fence_sanbox2" ipaddr="san_ip_example"
login="login_example" name="sanswitch1" passwd="password_example"/>
   </fencedevices>
   <rm>
   </rm>
</cluster>
```

Note that when you have finished configuring all of the components of your cluster, you will need to sync the cluster configuration file to all of the nodes, as described in *Section 5.15, "Propagating the Configuration File to the Cluster Nodes"*.

5.7.3. Configuring a Backup Fence Device

You can define multiple fencing methods for a node. If fencing fails using the first method, the system will attempt to fence the node using the second method, followed by any additional methods you have configured. To configure a backup fencing method for a node, you configure two methods for a node, configuring a fence instance for each method.

Note

The order in which the system will use the fencing methods you have configured follows their order in the cluster configuration file. The first method you configure with the **ccs** command is the primary fencing method, and the second method you configure is the backup fencing method. To change the order, you can remove the primary fencing method from the configuration file, then add that method back.

Note that at any time you can print a list of fence methods and instances currently configured for a node by executing the following command. If you do not specify a node, this command will list the fence methods and instances currently configured for all nodes.

```
ccs -h host --lsfenceinst [node]
```

Use the following procedure to configure a node with a primary fencing method that uses a fence device named **apc**, which uses the **fence_apc** fencing agent, and a backup fencing device that uses a fence device named **sanswitch1**, which uses the **fence_sanbox2** fencing agent. Since the

sanswitch1 device is a storage-based fencing agent, you will need to configure unfencing for that device as well.

1. Add a primary fence method for the node, providing a name for the fence method.

   ```
   ccs -h host --addmethod method node
   ```

   For example, to configure a fence method named **APC** as the primary method for the node **node-01.example.com** in the configuration file on the cluster node **node-01.example.com**, execute the following command:

   ```
   ccs -h node01.example.com --addmethod APC node01.example.com
   ```

2. Add a fence instance for the primary method. You must specify the fence device to use for the node, the node this instance applies to, the name of the method, and any options for this method that are specific to this node:

   ```
   ccs -h host --addfenceinst fencedevicename node method [options]
   ```

   For example, to configure a fence instance in the configuration file on the cluster node **node-01.example.com** that uses the APC switch power port 1 on the fence device named **apc** to fence cluster node **node-01.example.com** using the method named **APC**, execute the following command:

   ```
   ccs -h node01.example.com --addfenceinst apc node01.example.com APC port=1
   ```

3. Add a backup fence method for the node, providing a name for the fence method.

   ```
   ccs -h host --addmethod method node
   ```

   For example, to configure a backup fence method named **SAN** for the node **node-01.example.com** in the configuration file on the cluster node **node-01.example.com**, execute the following command:

   ```
   ccs -h node01.example.com --addmethod SAN  node01.example.com
   ```

4. Add a fence instance for the backup method. You must specify the fence device to use for the node, the node this instance applies to, the name of the method, and any options for this method that are specific to this node:

   ```
   ccs -h host --addfenceinst fencedevicename node method [options]
   ```

   For example, to configure a fence instance in the configuration file on the cluster node **node-01.example.com** that uses the SAN switch power port 11 on the fence device named **sanswitch1** to fence cluster node **node-01.example.com** using the method named **SAN**, execute the following command:

```
ccs -h node01.example.com --addfenceinst sanswitch1 node01.example.com SAN port=11
```

5. Since the **sanswitch1** device is a storage-based device, you must configure unfencing for this device.

```
ccs -h node01.example.com --addunfence sanswitch1 node01.example.com port=11 action=on
```

You can continue to add fencing methods as needed.

This procedure configures a fence device and a backup fence device for one node in the cluster. You will need to configure fencing for the other nodes in the cluster as well.

*Example 5.4, "**cluster.conf** After Adding Backup Fence Methods"* shows a **cluster.conf** configuration file after you have added a power-based primary fencing method and a storage-based backup fencing method to each node in the cluster.

Example 5.4. **cluster.conf** After Adding Backup Fence Methods

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC">
              <device name="apc" port="1"/>
             </method>
            <method name="SAN">
       <device name="sanswitch1" port="11"/>
             </method>
         </fence>
         <unfence>
             <device name="sanswitch1" port="11" action="on"/>
         </unfence
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="APC">
              <device name="apc" port="2"/>
            </method>
            <method name="SAN">
       <device name="sanswitch1" port="12"/>
             </method>
         </fence>
         <unfence>
             <device name="sanswitch1" port="12" action="on"/>
         </unfence
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
            <method name="APC">
              <device name="apc" port="3"/>
            </method>
            <method name="SAN">
       <device name="sanswitch1" port="13"/>
             </method>
         </fence>
         <unfence>
             <device name="sanswitch1" port="13" action="on"/>
```

```
        </unfence
    </clusternode>
  </clusternodes>
  <fencedevices>
      <fencedevice agent="fence_apc" ipaddr="apc_ip_example" login="login_example"
 name="apc" passwd="password_example"/>
      <fencedevice agent="fence_sanbox2" ipaddr="san_ip_example" login="login_example"
 name="sanswitch1" passwd="password_example"/>
  </fencedevices>
  <rm>
  </rm>
</cluster>
```

Note that when you have finished configuring all of the components of your cluster, you will need to sync the cluster configuration file to all of the nodes, as described in *Section 5.15, "Propagating the Configuration File to the Cluster Nodes"*.

Note

The order in which the system will use the fencing methods you have configured follows their order in the cluster configuration file. The first method you configure is the primary fencing method, and the second method you configure is the backup fencing method. To change the order, you can remove the primary fencing method from the configuration file, then add that method back.

5.7.4. Configuring a Node with Redundant Power

If your cluster is configured with redundant power supplies for your nodes, you must be sure to configure fencing so that your nodes fully shut down when they need to be fenced. If you configure each power supply as a separate fence method, each power supply will be fenced separately; the second power supply will allow the system to continue running when the first power supply is fenced and the system will not be fenced at all. To configure a system with dual power supplies, you must configure your fence devices so that both power supplies are shut off and the system is taken completely down. This requires that you configure two instances within a single fencing method, and that for each instance you configure both fence devices with an **`action`** attribute of **`off`** before configuring each of the devices with an **`action`** attribute of **`on`**.

To configure fencing for a node with dual power supplies, follow the steps in this section.

1. Before you can configure fencing for a node with redundant power, you must configure each of the power switches as a fence device for the cluster. For information on configuring fence devices, see *Section 5.5, "Configuring Fence Devices"*.

   To print a list of fence devices currently configured for your cluster, execute the following command:

   ```
   ccs -h host --lsfencedev
   ```

2. Add a fence method for the node, providing a name for the fence method.

```
ccs -h host --addmethod method node
```

For example, to configure a fence method named **APC-dual** for the node **node-01.example.com** in the configuration file on the cluster node **node-01.example.com**, execute the following command:

```
ccs -h node01.example.com --addmethod APC-dual node01.example.com
```

3. Add a fence instance for the first power supply to the fence method. You must specify the fence device to use for the node, the node this instance applies to, the name of the method, and any options for this method that are specific to this node. At this point you configure the **action** attribute as **off**.

```
ccs -h host --addfenceinst fencedevicename node method [options] action=off
```

For example, to configure a fence instance in the configuration file on the cluster node **node-01.example.com** that uses the APC switch power port 1 on the fence device named **apc1** to fence cluster node **node-01.example.com** using the method named **APC-dual**, and setting the **action** attribute to **off**, execute the following command:

```
ccs -h node01.example.com --addfenceinst apc1 node01.example.com APC-dual port=1
 action=off
```

4. Add a fence instance for the second power supply to the fence method. You must specify the fence device to use for the node, the node this instance applies to, the name of the method, and any options for this method that are specific to this node. At this point you configure the **action** attribute as **off** for this instance as well:

```
ccs -h host --addfenceinst fencedevicename node method [options] action=off
```

For example, to configure a second fence instance in the configuration file on the cluster node **node-01.example.com** that uses the APC switch power port 1 on the fence device named **apc2** to fence cluster node **node-01.example.com** using the same method as you specified for the first instance named **APC-dual**, and setting the **action** attribute to **off**, execute the following command:

```
ccs -h node01.example.com --addfenceinst apc2 node01.example.com APC-dual port=1
 action=off
```

5. At this point, add another fence instance for the first power supply to the fence method, configuring the **action** attribute as **on**. You must specify the fence device to use for the node, the node this instance applies to, the name of the method, and any options for this method that are specific to this node, and specifying the **action** attribute as **on**:

```
ccs -h host --addfenceinst fencedevicename node method [options] action=on
```

For example, to configure a fence instance in the configuration file on the cluster node **node-01.example.com** that uses the APC switch power port 1 on the fence device named **apc1** to fence cluster node **node-01.example.com** using the method named **APC-dual**, and setting the **action** attribute to **on**, execute the following command:

```
ccs -h node01.example.com --addfenceinst apc1 node01.example.com APC-dual port=1
 action=on
```

6. Add another fence instance for second power supply to the fence method, specifying the **action** attribute as **on** for this instance. You must specify the fence device to use for the node, the node this instance applies to, the name of the method, and any options for this method that are specific to this node as well as the **action** attribute of **on**.

```
ccs -h host --addfenceinst fencedevicename node method [options] action=on
```

For example, to configure a second fence instance in the configuration file on the cluster node **node-01.example.com** that uses the APC switch power port 1 on the fence device named **apc2** to fence cluster node **node-01.example.com** using the same method as you specified for the first instance named **APC-dual** and setting the **action** attribute to **on**, execute the following command:

```
ccs -h node01.example.com --addfenceinst apc2 node01.example.com APC-dual port=1
 action=on
```

*Example 5.5, "**cluster.conf** After Adding Dual-Power Fencing "* shows a **cluster.conf** configuration file after you have added fencing for two power supplies for each node in a cluster.

Example 5.5. **cluster.conf** After Adding Dual-Power Fencing

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC-dual">
              <device name="apc1" port="1"action="off"/>
              <device name="apc2" port="1"action="off"/>
              <device name="apc1" port="1"action="on"/>
              <device name="apc2" port="1"action="on"/>
             </method>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="APC-dual">
              <device name="apc1" port="2"action="off"/>
              <device name="apc2" port="2"action="off"/>
              <device name="apc1" port="2"action="on"/>
              <device name="apc2" port="2"action="on"/>
            </method>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
```

```
            <method name="APC-dual">
              <device name="apc1" port="3"action="off"/>
              <device name="apc2" port="3"action="off"/>
              <device name="apc1" port="3"action="on"/>
              <device name="apc2" port="3"action="on"/>
            </method>
          </fence>
      </clusternode>
    </clusternodes>
    <fencedevices>
        <fencedevice agent="fence_apc" ipaddr="apc_ip_example" login="login_example"
 name="apc1" passwd="password_example"/>
        <fencedevice agent="fence_apc" ipaddr="apc_ip_example" login="login_example"
 name="apc2" passwd="password_example"/>
    </fencedevices>
    <rm>
    </rm>
</cluster>
```

Note that when you have finished configuring all of the components of your cluster, you will need to sync the cluster configuration file to all of the nodes, as described in *Section 5.15, "Propagating the Configuration File to the Cluster Nodes"*.

5.7.5. Removing Fence Methods and Fence Instances

To remove a fence method from your cluster configuration, execute the following command:

```
ccs -h host --rmmethod method node
```

For example, to remove a fence method that you have named **APC** that you have configured for **node01.example.com** from the cluster configuration file on cluster node **node01.example.com**, execute the following command:

```
ccs -h node01.example.com  --rmmethod APC node01.example.com
```

To remove all fence instances of a fence device from a fence method, execute the following command:

```
ccs -h host --rmfenceinst fencedevicename node method
```

For example, to remove all instances of the fence device named **apc1** from the method named **APC-dual** configured for **node01.example.com** from the cluster configuration file on cluster node **node01.example.com**, execute the following command:

```
ccs -h node01.example.com --rmfenceinst apc1 node01.example.com APC-dual
```

5.8. Configuring a Failover Domain

A failover domain is a named subset of cluster nodes that are eligible to run a cluster service in the event of a node failure. A failover domain can have the following characteristics:

- Unrestricted — Allows you to specify that a subset of members are preferred, but that a cluster service assigned to this domain can run on any available member.
- Restricted — Allows you to restrict the members that can run a particular cluster service. If none of the members in a restricted failover domain are available, the cluster service cannot be started (either manually or by the cluster software).
- Unordered — When a cluster service is assigned to an unordered failover domain, the member on which the cluster service runs is chosen from the available failover domain members with no priority ordering.
- Ordered — Allows you to specify a preference order among the members of a failover domain. The member at the top of the list is the most preferred, followed by the second member in the list, and so on.
- Failback — Allows you to specify whether a service in the failover domain should fail back to the node that it was originally running on before that node failed. Configuring this characteristic is useful in circumstances where a node repeatedly fails and is part of an ordered failover domain. In that circumstance, if a node is the preferred node in a failover domain, it is possible for a service to fail over and fail back repeatedly between the preferred node and another node, causing severe impact on performance.

Note

The failback characteristic is applicable only if ordered failover is configured.

Note

Changing a failover domain configuration has no effect on currently running services.

Note

Failover domains are *not* required for operation.

By default, failover domains are unrestricted and unordered.

In a cluster with several members, using a restricted failover domain can minimize the work to set up the cluster to run a cluster service (such as **`httpd`**), which requires you to set up the configuration identically on all members that run the cluster service. Instead of setting up the entire cluster to run the cluster service, you can set up only the members in the restricted failover domain that you associate with the cluster service.

Note

To configure a preferred member, you can create an unrestricted failover domain comprising only one cluster member. Doing that causes a cluster service to run on that cluster member primarily (the preferred member), but allows the cluster service to fail over to any of the other members.

To configure a failover domain, perform the following procedure:

1. To add a failover domain, execute the following command:

```
ccs -h host --addfailoverdomain name [restricted] [ordered] [nofailback]
```

Note

The name should be descriptive enough to distinguish its purpose relative to other names used in your cluster.

For example, the following command configures a failover domain named **example_pri** on **node-01.example.com** that is unrestricted, ordered, and allows failback:

```
ccs -h node-01.example.com --addfailoverdomain example_pri ordered
```

2. To add a node to a failover domain, execute the following command:

```
ccs -h host --addfailoverdomainnode failoverdomain node priority
```

For example, to configure the failover domain **example_pri** in the configuration file on **node-01.example.com** so that it contains **node-01.example.com** with a priority of 1, **node-02.example.com** with a priority of 2, and **node-03.example.com** with a priority of 3, execute the following commands:

```
ccs -h node-01.example.com --addfailoverdomainnode example_pri node-01.example.com 1
ccs -h node-01.example.com --addfailoverdomainnode example_pri node-02.example.com 2
ccs -h node-01.example.com --addfailoverdomainnode example_pri node-03.example.com 3
```

You can list all of the failover domains and failover domain nodes configured in a cluster with the following command:

```
ccs -h host --lsfailoverdomain
```

To remove a failover domain, execute the following command:

```
ccs -h host --rmfailoverdomain name
```

To remove a node from a failover domain, execute the following command:

```
ccs -h host --rmfailoverdomainnode failoverdomain node
```

Note that when you have finished configuring all of the components of your cluster, you will need to sync the cluster configuration file to all of the nodes, as described in *Section 5.15, "Propagating the Configuration File to the Cluster Nodes"*.

5.9. Configuring Global Cluster Resources

You can configure two types of resources:

- Global — Resources that are available to any service in the cluster.
- Service-specific — Resources that are available to only one service.

To see a list of currently configured resources and services in the cluster, execute the following command:

```
ccs -h host --lsservices
```

To add a global cluster resource, execute the following command. You can add a resource that is local to a particular service when you configure the service, as described in *Section 5.10, "Adding a Cluster Service to the Cluster"*.

```
ccs -h host --addresource resourcetype [resource options]
```

For example, the following command adds a global file system resource to the cluster configuration file on **node01.example.com**. The name of the resource is **web_fs**, the file system device is **/dev/sdd2**, the file system mountpoint is **/var/www**, and the file system type is **ext3**.

```
ccs -h node01.example.com --addresource fs name=web_fs device=/dev/sdd2 mountpoint=/var/www
 fstype=ext3
```

For information about the available resource types and resource options, see *Appendix B, HA Resource Parameters*.

To remove a global resource, execute the following command:

```
ccs -h host --rmresource resourcetype [resource options]
```

If you need to modify the parameters of an existing global resource, you can remove the resource and configure it again.

Note that when you have finished configuring all of the components of your cluster, you will need to sync the cluster configuration file to all of the nodes, as described in *Section 5.15, "Propagating the Configuration File to the Cluster Nodes"*.

5.10. Adding a Cluster Service to the Cluster

To configure a cluster service in a cluster, perform the following steps:

1. Add a service to the cluster with the following command:

```
ccs -h host --addservice servicename [service options]
```

Note

Use a descriptive name that clearly distinguishes the service from other services in the cluster.

When you add a service to the cluster configuration, you configure the following attributes:

- **autostart** — Specifies whether to autostart the service when the cluster starts. Use "1" to enable and "0" to disable; the default is enabled.
- **domain** — Specifies a failover domain (if required).
- **exclusive** — Specifies a policy wherein the service only runs on nodes that have no other services running on them.
- **recovery** — Specifies a recovery policy for the service. The options are to relocate, restart, disable, or restart-disable the service. For information on service recovery policies, refer to *Table B.18, "Service"*.

For example, to add a service to the configuration file on the cluster node **node-01.example.com** named **example_apache** that uses the failover domain **example_pri**, and that has recovery policy of **relocate**, execute the following command:

```
ccs -h node-01.example.com --addservice example_apache domain=example_pri
 recovery=relocate
```

When configuring services for a cluster, you may find it useful to see a listing of available services for your cluster and the options available for each service. For information on using the **ccs** command to print a list of available services and their options, refer to *Section 5.11, "Listing Available Cluster Services"*.

2. Add resources to the service with the following command:

```
ccs -h host --addsubservice servicename subservice [service options]
```

Depending on the type of resources you want to use, populate the service with global or service-specific resources. To add a global resource, use the **--addsubservice** option of the **ccs** to add a resource. For example, to add the global file system resource named **web_fs** to the service named **example_apache** on the cluster configuration file on **node-01.example.com**, execute the following command:

```
ccs -h node01.example.com --addsubservice example_apache fs ref=web_fs
```

To add a service-specific resource to the service, you need to specify all of the service options. For example, if you had not previously defined **web_fs** as a global service, you could add it as a service-specific resource with the following command:

```
ccs -h node01.example.com --addsubservice example_apache fs name=web_fs device=/dev/sdd2
 mountpoint=/var/www fstype=ext3
```

3. To add a child service to the service, you also use the **--addsubservice** option of the **ccs** command, specifying the service options.

   If you need to add services within a tree structure of dependencies, use a colon (":") to separate elements and brackets to identify subservices of the same type. The following example adds a third **nfsclient** service as a subservice of an **nfsclient** service which is in itself a subservice of an **nfsclient** service which is a subservice of a service named **service_a**:

```
ccs -h node01.example.com --addsubservice service_a nfsclient[1]:nfsclient[2]:nfsclient
```

Note

If you are adding a Samba-service resource, add it directly to the service, *not* as a child of another resource.

Note

To verify the existence of the IP service resource used in a cluster service, you must use the **/sbin/ip addr list** command on a cluster node. The following output shows the **/sbin/ip addr list** command executed on a node running a cluster service:

```
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1356 qdisc pfifo_fast qlen 1000
    link/ether 00:05:5d:9a:d8:91 brd ff:ff:ff:ff:ff:ff
    inet 10.11.4.31/22 brd 10.11.7.255 scope global eth0
    inet6 fe80::205:5dff:fe9a:d891/64 scope link
    inet 10.11.4.240/22 scope global secondary eth0
       valid_lft forever preferred_lft forever
```

To remove a service and all of its subservices, execute the following command:

```
ccs -h host --rmservice servicename
```

To remove a subservice, execute the following command:

```
ccs -h host --rmsubservice servicename subservice [service options]
```

Note that when you have finished configuring all of the components of your cluster, you will need to sync the cluster configuration file to all of the nodes, as described in *Section 5.15, "Propagating the Configuration File to the Cluster Nodes"*.

5.11. Listing Available Cluster Services

You can use the **ccs** command to print a list of services that are available for a cluster. You can also use the **ccs** command to print a list of the options you can specify for a particular service type.

To print a list of cluster services currently available for your cluster, execute the following command:

```
ccs -h host --lsserviceopts
```

For example, the following command lists the cluster services available on the cluster node **node1**, showing sample output.

```
[root@ask-03 ~]# ccs -h node1 --lsserviceopts
service - Defines a service (resource group).
ASEHAagent - Sybase ASE Failover Instance
SAPDatabase - SAP database resource agent
SAPInstance - SAP instance resource agent
apache - Defines an Apache web server
clusterfs - Defines a cluster file system mount.
fs - Defines a file system mount.
ip - This is an IP address.
lvm - LVM Failover script
mysql - Defines a MySQL database server
named - Defines an instance of named server
netfs - Defines an NFS/CIFS file system mount.
nfsclient - Defines an NFS client.
nfsexport - This defines an NFS export.
nfsserver - This defines an NFS server resource.
openldap - Defines an Open LDAP server
oracledb - Oracle 10g Failover Instance
orainstance - Oracle 10g Failover Instance
oralistener - Oracle 10g Listener Instance
postgres-8 - Defines a PostgreSQL server
samba - Dynamic smbd/nmbd resource agent
script - LSB-compliant init script as a clustered resource.
tomcat-6 - Defines a Tomcat server
vm - Defines a Virtual Machine
action - Overrides resource action timings for a resource instance.
```

To print a list of the options you can specify for a particular service type, execute the following command:

```
ccs -h host --lsserviceopts service_type
```

For example, the following command lists the service options for the **vm** service.

```
[root@ask-03 ~]# ccs -f node1 --lsserviceopts vm
vm - Defines a Virtual Machine
  Required Options:
    name: Name
  Optional Options:
    domain: Cluster failover Domain
    autostart: Automatic start after quorum formation
    exclusive: Exclusive resource group
    recovery: Failure recovery policy
    migration_mapping: memberhost:targethost,memberhost:targethost ..
    use_virsh: If set to 1, vm.sh will use the virsh command to manage virtual machines
 instead of xm. This is required when using non-Xen virtual machines (e.g. qemu / KVM).
    xmlfile: Full path to libvirt XML file describing the domain.
    migrate: Migration type (live or pause, default = live).
    path: Path to virtual machine configuration files.
    snapshot: Path to the snapshot directory where the virtual machine image will be stored.
    depend: Top-level service this depends on, in service:name format.
    depend_mode: Service dependency mode (soft or hard).
    max_restarts: Maximum restarts for this service.
    restart_expire_time: Restart expiration time; amount of time before a restart is
 forgotten.
    status_program: Additional status check program
    hypervisor: Hypervisor
    hypervisor_uri: Hypervisor URI (normally automatic).
    migration_uri: Migration URI (normally automatic).
    __independent_subtree: Treat this and all children as an independent subtree.
    __enforce_timeouts: Consider a timeout for operations as fatal.
    __max_failures: Maximum number of failures before returning a failure to a status check.
    __failure_expire_time: Amount of time before a failure is forgotten.
    __max_restarts: Maximum number restarts for an independent subtree before giving up.
    __restart_expire_time: Amount of time before a failure is forgotten for an independent
 subtree.
```

5.12. Virtual Machine Resources

Virtual machine resources are configured differently than other cluster resources. In particular, they are not grouped into service definitions. As of the Red Hat Enterprise Linux 6.2 release, when you configure a virtual machine in a cluster with the **ccs** command you can use the **--addvm** (rather than the **addservice** option). This ensures that the **vm** resource is defined directly under the **rm** configuration node in the cluster configuration file.

A virtual machine resource requires at least a **name** and a **path** attribute. The **name** attribute should match the name of the **libvirt** domain and the **path** attribute should specify the directory where the shared virtual machine definitions are stored.

Note

The **path** attribute in the cluster configuration file is a path specification or a directory name, not a path to an individual file.

If virtual machine definitions are stored on a shared directory named **/mnt/vm_defs**, the following command will define a virtual machine named **guest1**:

```
# ccs -h node1.example.com --addvm guest1 path=/mnt/vm_defs
```

Running this command adds the following line to the **rm** configuration node in the **cluster.conf** file:

```
<vm name="guest1" path="/mnt/vm_defs"/>
```

5.13. Configuring a Quorum Disk

Important

Quorum-disk parameters and heuristics depend on the site environment and the special requirements needed. To understand the use of quorum-disk parameters and heuristics, refer to the qdisk(5) man page. If you require assistance understanding and using quorum disk, contact an authorized Red Hat support representative.

Use the following command to configure your system for using a quorum disk:

```
ccs -h host --setquorumd [quorumd options]
```

Table 5.1, "Quorum Disk Options" summarizes the meaning of the quorum disk options you may need to set. For a complete list of quorum disk parameters, refer to the cluster schema at **/usr/share/cluster/cluster.rng**, and the annotated schema at **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html**.

Table 5.1. Quorum Disk Options

Parameter	Description
interval	The frequency of read/write cycles, in seconds.
votes	The number of votes the quorum daemon advertises to **cman** when it has a high enough score.
tko	The number of cycles a node must miss to be declared dead.
min_score	The minimum score for a node to be considered "alive". If omitted or set to 0, the default function, **floor((*n*+1)/2)**, is used, where *n* is the sum of the heuristics scores. The **Minimum Score** value must never exceed the sum of the heuristic scores; otherwise, the quorum disk cannot be available.
device	The storage device the quorum daemon uses. The device must be the same on all nodes.
label	Specifies the quorum disk label created by the **mkqdisk** utility. If this field contains an entry, the label overrides the **Device** field. If this field is used, the quorum daemon reads **/proc/partitions** and checks for qdisk signatures on every block device found, comparing the label against the specified label. This is useful in configurations where the quorum device name differs among nodes.

Use the following command to configure the heuristics for a quorum disk:

```
ccs -h host --addheuristic [heuristic options]
```

Table 5.2, "Quorum Disk Heuristics" summarizes the meaning of the quorum disk heuristics you may need to set.

Table 5.2. Quorum Disk Heuristics

Parameter	Description
program	The path to the program used to determine if this heuristic is available. This can be anything that can be executed by **/bin/sh -c**. A return value of 0 indicates success; anything else indicates failure. This parameter is required to use a quorum disk.
interval	The frequency (in seconds) at which the heuristic is polled. The default interval for every heuristic is 2 seconds.
score	The weight of this heuristic. Be careful when determining scores for heuristics. The default score for each heuristic is 1.
tko	The number of consecutive failures required before this heuristic is declared unavailable.

To see a list of the quorum disk options and heuristics that are configured on a system, you can execute the following command:

```
ccs -h host --lsquorum
```

To remove a heuristic specified by a heuristic option, you can execute the following command:

```
ccs -h host rmheuristic [heuristic options]
```

Note that when you have finished configuring all of the components of your cluster, you will need to sync the cluster configuration file to all of the nodes, as described in *Section 5.15, "Propagating the Configuration File to the Cluster Nodes"*.

Note

Syncing and activating propagates and activates the updated cluster configuration file. However, for the quorum disk to operate, you must restart the cluster (refer to *Section 6.2, "Starting and Stopping a Cluster"*).

5.14. Miscellaneous Cluster Configuration

This section describes using the **ccs** command to configure the following:

- *Section 5.14.1, "Cluster Configuration Version"*
- *Section 5.14.2, "Multicast Configuration"*
- *Section 5.14.3, "Configuring a Two-Node Cluster"*

You can also use the **ccs** command to set advanced cluster configuration parameters, including **totem** options, **dlm** options, **rm** options and **cman** options. For information on setting these parameters see the **ccs**(8) man page and the annotated cluster configuration file schema at **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html**.

To view a list of the miscellaneous cluster attributes that have been configured for a cluster, execute the following command:

```
ccs -h host --lsmisc
```

5.14.1. Cluster Configuration Version

A cluster configuration file includes a cluster configuration version value. The configuration version value is set to **1** by default when you create a cluster configuration file and it is automatically incremented each time you modify your cluster configuration. However, if you need to set it to another value, you can specify it with the following command:

```
ccs -h host --setversion n
```

You can get the current configuration version value on an existing cluster configuration file with the following command:

```
ccs -h host --getversion
```

To increment the current configuration version value by 1 in the cluster configuration file on every node in the cluster, execute the following command:

```
ccs -h host --incversion
```

5.14.2. Multicast Configuration

If you do not specify a multicast address in the cluster configuration file, the Red Hat High Availability Add-On software creates one based on the cluster ID. It generates the lower 16 bits of the address and appends them to the upper portion of the address according to whether the IP protocol is IPv4 or IPv6:

- For IPv4 — The address formed is 239.192. plus the lower 16 bits generated by Red Hat High Availability Add-On software.
- For IPv6 — The address formed is FF15:: plus the lower 16 bits generated by Red Hat High Availability Add-On software.

Note

The cluster ID is a unique identifier that **cman** generates for each cluster. To view the cluster ID, run the **cman_tool status** command on a cluster node.

You can manually specify a multicast address in the cluster configuration file with the following command:

```
ccs -h host --setmulticast multicastaddress
```

If you specify a multicast address, you should use the 239.192.x.x series (or FF15:: for IPv6) that **cman** uses. Otherwise, using a multicast address outside that range may cause unpredictable results. For example, using 224.0.0.x (which is "All hosts on the network") may not be routed correctly, or even routed at all by some hardware.

Note

If you specify a multicast address, make sure that you check the configuration of routers that cluster packets pass through. Some routers may take a long time to learn addresses, seriously impacting cluster performance.

To remove a multicast address from a configuration file, use the **--setmulticast** option of the **ccs** but do not specify a multicast address:

```
ccs -h host --setmulticast
```

5.14.3. Configuring a Two-Node Cluster

If you are configuring a two-node cluster, you can execute the following command to allow a single node to maintain quorum (for example, if one node fails):

```
ccs -h host --setcman two_node=1 expected_votes=1
```

5.15. Propagating the Configuration File to the Cluster Nodes

After you have created or edited a cluster configuration file on one of the nodes in the cluster, you need to propagate that same file to all of the cluster nodes and activate the configuration.

Use the following command to propagate and activate a cluster configuration file:

```
ccs -h host --sync --activate
```

To verify that all of the nodes specified in the hosts cluster configuration file have the identical cluster configuration file, execute the following command:

```
ccs -h host --checkconf
```

If you have created or edited a configuration file on a local node, use the following command to send that file to one of the nodes in the cluster:

```
ccs -f file -h host --setconf
```

To verify that all of the nodes specified in the local file have the identical cluster configuration file, execute the following command:

```
ccs -f file --checkconf
```

Managing Red Hat High Availability Add-On With ccs

This chapter describes various administrative tasks for managing the Red Hat High Availability Add-On by means of the **ccs** command, which is supported as of the Red Hat Enterprise Linux 6.1 release and later. This chapter consists of the following sections:

- *Section 6.1, "Managing Cluster Nodes"*
- *Section 6.2, "Starting and Stopping a Cluster"*
- *Section 6.3, "Diagnosing and Correcting Problems in a Cluster"*

6.1. Managing Cluster Nodes

This section documents how to perform the following node-management functions with the **ccs** command:

- *Section 6.1.1, "Causing a Node to Leave or Join a Cluster"*
- *Section 6.1.2, "Adding a Member to a Running Cluster"*

6.1.1. Causing a Node to Leave or Join a Cluster

You can use the **ccs** command to cause a node to leave a cluster by stopping cluster services on that node. Causing a node to leave a cluster does not remove the cluster configuration information from that node. Making a node leave a cluster prevents the node from automatically joining the cluster when it is rebooted.

To cause a node to leave a cluster, execute the following command, which stops cluster services on the node specified with the **-h** option:

```
ccs -h host --stop
```

When you stop cluster services on a node, any service that is running on that node will fail over.

To delete a node entirely from the cluster configuration, use the **--rmnode** option of the **ccs** command, as described in *Section 5.4, "Creating A Cluster"*.

To cause a node to rejoin a cluster execute the following command, which starts cluster services on the node specified with the **-h** option:

```
ccs -h host --start
```

6.1.2. Adding a Member to a Running Cluster

To add a member to a running cluster, add a node to the cluster as described in *Section 5.4, "Creating A Cluster"*. After updating the configuration file, propagate the file to all nodes in the cluster and be

sure to activate the new cluster configuration file, as described in *Section 5.15, "Propagating the Configuration File to the Cluster Nodes"*.

6.2. Starting and Stopping a Cluster

You can use the **ccs** command to stop a cluster by using the following command to stop cluster services on all nodes in the cluster:

```
ccs -h host --stopall
```

You can use the **ccs** command to start a cluster that is not running by using the following command to start cluster services on all nodes in the cluster:

```
ccs -h host --startall
```

6.3. Diagnosing and Correcting Problems in a Cluster

For information about diagnosing and correcting problems in a cluster, see *Chapter 9, Diagnosing and Correcting Problems in a Cluster*. There are a few simple checks that you can perform with the **ccs** command, however.

To verify that all of the nodes specified in the host's cluster configuration file have identical cluster configuration files, execute the following command:

```
ccs -h host --checkconf
```

If you have created or edited a configuration file on a local node, you can verify that all of the nodes specified in the local file have identical cluster configuration files with the following command:

```
ccs -f file --checkconf
```

Configuring Red Hat High Availability Add-On With Command Line Tools

This chapter describes how to configure Red Hat High Availability Add-On software by directly editing the cluster configuration file (**/etc/cluster/cluster.conf**) and using command-line tools. The chapter provides procedures about building a configuration file one section at a time, starting with a sample file provided in the chapter. As an alternative to starting with a sample file provided here, you could copy a skeleton configuration file from the **cluster.conf** man page. However, doing so would not necessarily align with information provided in subsequent procedures in this chapter. There are other ways to create and configure a cluster configuration file; this chapter provides procedures about building a configuration file one section at a time. Also, keep in mind that this is just a starting point for developing a configuration file to suit your clustering needs.

This chapter consists of the following sections:

- *Section 7.1, "Configuration Tasks"*
- *Section 7.2, "Creating a Basic Cluster Configuration File"*
- *Section 7.3, "Configuring Fencing"*
- *Section 7.4, "Configuring Failover Domains"*
- *Section 7.5, "Configuring HA Services"*
- *Section 7.6, "Verifying a Configuration"*

Important

Make sure that your deployment of High Availability Add-On meets your needs and can be supported. Consult with an authorized Red Hat representative to verify your configuration prior to deployment. In addition, allow time for a configuration burn-in period to test failure modes.

Important

This chapter references commonly used **cluster.conf** elements and attributes. For a comprehensive list and description of **cluster.conf** elements and attributes, refer to the cluster schema at **/usr/share/cluster/cluster.rng**, and the annotated schema at **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** (for example **/usr/share/doc/cman-3.0.12/cluster_conf.html**).

Important

Certain procedure in this chapter call for using the **`cman_tool version -r`** command to propagate a cluster configuration throughout a cluster. Using that command requires that **`ricci`** is running. Using **ricci** requires a password the first time you interact with **ricci** from any specific machine. For information on the **`ricci`** service, refer to *Section 2.14, "Considerations for `ricci`"*.

Note

Procedures in this chapter may include specific commands for some of the command-line tools listed in *Appendix E, Command Line Tools Summary*. For more information about all commands and variables, refer to the man page for each command-line tool.

7.1. Configuration Tasks

Configuring Red Hat High Availability Add-On software with command-line tools consists of the following steps:

1. Creating a cluster. Refer to *Section 7.2, "Creating a Basic Cluster Configuration File"*.
2. Configuring fencing. Refer to *Section 7.3, "Configuring Fencing"*.
3. Configuring failover domains. Refer to *Section 7.4, "Configuring Failover Domains"*.
4. Configuring HA services. Refer to *Section 7.5, "Configuring HA Services"*.
5. Verifying a configuration. Refer to *Section 7.6, "Verifying a Configuration"*.

7.2. Creating a Basic Cluster Configuration File

Provided that cluster hardware, Red Hat Enterprise Linux, and High Availability Add-On software are installed, you can create a cluster configuration file (**`/etc/cluster/cluster.conf`**) and start running the High Availability Add-On. As a starting point only, this section describes how to create a skeleton cluster configuration file without fencing, failover domains, and HA services. Subsequent sections describe how to configure those parts of the configuration file.

Important

This is just an interim step to create a cluster configuration file; the resultant file does not have any fencing and is not considered to be a supported configuration.

The following steps describe how to create and configure a skeleton cluster configuration file. Ultimately, the configuration file for your cluster will vary according to the number of nodes, the type of fencing, the type and number of HA services, and other site-specific requirements.

1. At any node in the cluster, create **/etc/cluster/cluster.conf**, using the template of the example in *Example 7.1, "**cluster.conf** Sample: Basic Configuration"*.

2. **(Optional)** If you are configuring a two-node cluster, you can add the following line to the configuration file to allow a single node to maintain quorum (for example, if one node fails):

   ```
   <cman two_node="1" expected_votes="1"/>
   ```

   Refer to *Example 7.2, "**cluster.conf** Sample: Basic Two-Node Configuration"*.

3. Specify the cluster name and the configuration version number using the **cluster** attributes: **name** and **config_version** (refer to *Example 7.1, "**cluster.conf** Sample: Basic Configuration"* or *Example 7.2, "**cluster.conf** Sample: Basic Two-Node Configuration"*).

4. In the **clusternodes** section, specify the node name and the node ID of each node using the **clusternode** attributes: **name** and **nodeid**.

5. Save **/etc/cluster/cluster.conf**.

6. Validate the file with against the cluster schema (**cluster.rng**) by running the **ccs_config_validate** command. For example:

   ```
   [root@example-01 ~]# ccs_config_validate
   Configuration validates
   ```

7. Propagate the configuration file to **/etc/cluster/** in each cluster node. For example, you could propagate the file to other cluster nodes using the **scp** command.

   Note

   Propagating the cluster configuration file this way is necessary the first time a cluster is created. Once a cluster is installed and running, the cluster configuration file can be propagated using the **cman_tool version -r**. It is possible to use the **scp** command to propagate an updated configuration file; however, the cluster software must be stopped on all nodes while using the **scp** command.In addition, you should run the **ccs_config_validate** if you propagate an updated configuration file via the **scp**.

   Note

   While there are other elements and attributes present in the sample configuration file (for example, **fence** and **fencedevices**), there is no need to populate them now. Subsequent procedures in this chapter provide information about specifying other elements and attributes.

8. Start the cluster. At each cluster node run the following command:

   ```
   service cman start
   ```

For example:

```
[root@example-01 ~]# service cman start
Starting cluster:
   Checking Network Manager...                             [  OK  ]
   Global setup...                                         [  OK  ]
   Loading kernel modules...                               [  OK  ]
   Mounting configfs...                                    [  OK  ]
   Starting cman...                                        [  OK  ]
   Waiting for quorum...                                   [  OK  ]
   Starting fenced...                                      [  OK  ]
   Starting dlm_controld...                                [  OK  ]
   Starting gfs_controld...                                [  OK  ]
   Unfencing self...                                       [  OK  ]
   Joining fence domain...                                 [  OK  ]
```

9. At any cluster node, run **cman_tool nodes** to verify that the nodes are functioning as members in the cluster (signified as "M" in the status column, "Sts"). For example:

```
[root@example-01 ~]# cman_tool nodes
Node  Sts   Inc   Joined               Name
   1   M    548   2010-09-28 10:52:21  node-01.example.com
   2   M    548   2010-09-28 10:52:21  node-02.example.com
   3   M    544   2010-09-28 10:52:21  node-03.example.com
```

10. If the cluster is running, proceed to *Section 7.3, "Configuring Fencing"*.

Basic Configuration Examples

*Example 7.1, "**cluster.conf** Sample: Basic Configuration"* and *Example 7.2, "**cluster.conf** Sample: Basic Two-Node Configuration"* (for a two-node cluster) each provide a very basic sample cluster configuration file as a starting point. Subsequent procedures in this chapter provide information about configuring fencing and HA services.

Example 7.1. **cluster.conf** Sample: Basic Configuration

```
<cluster name="mycluster" config_version="2">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
   </fencedevices>
   <rm>
   </rm>
</cluster>
```

Example 7.2. **cluster.conf** Sample: Basic Two-Node Configuration

```
<cluster name="mycluster" config_version="2">
   <cman two_node="1" expected_votes="1"/>
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
   </fencedevices>
   <rm>
   </rm>
</cluster>
```

The consensus Value for totem in a Two-Node Cluster

When you create a two-node cluster and you do not intend to add additional nodes to the cluster at a later time, then you should omit the **consensus** value in the **totem** tag in the **cluster.conf** file so that the **consensus** value is calculated automatically. When the **consensus** value is calculated automatically, the following rules are used:

- If there are two nodes or fewer, the **consensus** value will be (token * 0.2), with a ceiling of 2000 msec and a floor of 200 msec.
- If there are three or more nodes, the **consensus** value will be (token + 2000 msec)

If you let the **cman** utility configure your consensus timeout in this fashion, then moving at a later time from two to three (or more) nodes will require a cluster restart, since the consensus timeout will need to change to the larger value based on the token timeout.

If you are configuring a two-node cluster and intend to upgrade in the future to more than two nodes, you can override the consensus timeout so that a cluster restart is not required when moving from two to three (or more) nodes. This can be done in the **cluster.conf** as follows:

```
<totem token="X" consensus="X + 2000" />
```

Note that the configuration parser does not calculate X + 2000 automatically. An integer value must be used rather than an equation.

The advantage of using the optimized consensus timeout for two-node clusters is that overall failover time is reduced for the two-node case, since consensus is not a function of the token timeout.

Note that for two-node autodetection in **cman**, the number of physical nodes is what matters and not the presence of the **two_node=1** directive in the **cluster.conf** file.

7.3. Configuring Fencing

Configuring fencing consists of (a) specifying one or more fence devices in a cluster and (b) specifying one or more fence methods for each node (using a fence device or fence devices specified).

Based on the type of fence devices and fence methods required for your configuration, configure **cluster.conf** as follows:

1. In the **fencedevices** section, specify each fence device, using a **fencedevice** element and fence-device dependent attributes. *Example 7.3, "APC Fence Device Added to **cluster.conf**"* shows an example of a configuration file with an APC fence device added to it.

2. At the **clusternodes** section, within the **fence** element of each **clusternode** section, specify each fence method of the node. Specify the fence method name, using the **method** attribute, **name**. Specify the fence device for each fence method, using the **device** element and its attributes, **name** and fence-device-specific parameters. *Example 7.4, "Fence Methods Added to **cluster.conf**"* shows an example of a fence method with one fence device for each node in the cluster.

3. For non-power fence methods (that is, SAN/storage fencing), at the **clusternodes** section, add an **unfence** section. This ensures that a fenced node is not re-enabled until the node has been rebooted. For more information about unfencing a node, refer to the **fence_node**(8) man page.

   The **unfence** section does not contain **method** sections like the **fence** section does. It contains **device** references directly, which mirror the corresponding device sections for **fence**, with the notable addition of the explicit action (**action**) of "on" or "enable". The same **fencedevice** is referenced by both **fence** and **unfence device** lines, and the same per-node arguments should be repeated.

   Specifying the **action** attribute as "on" or "enable" enables the node when rebooted. *Example 7.4, "Fence Methods Added to **cluster.conf**"* and *Example 7.5, "**cluster.conf**: Multiple Fence Methods per Node"* include examples of the **unfence** elements and attributed.

   For more information about **unfence** refer to the **fence_node** man page.

4. Update the **config_version** attribute by incrementing its value (for example, changing from **config_version="2"** to **config_version="3">**).

5. Save **/etc/cluster/cluster.conf**.

6. **(Optional)** Validate the updated file against the cluster schema (**cluster.rng**) by running the **ccs_config_validate** command. For example:

   ```
   [root@example-01 ~]# ccs_config_validate
   Configuration validates
   ```

7. Run the **cman_tool version -r** command to propagate the configuration to the rest of the cluster nodes. This will also run additional validation. It is necessary that **ricci** be running in each cluster node to be able to propagate updated cluster configuration information.

8. Verify that the updated configuration file has been propagated.

9. Proceed to *Section 7.4, "Configuring Failover Domains"*.

If required, you can configure complex configurations with multiple fence methods per node and with multiple fence devices per fence method. When specifying multiple fence methods per node, if fencing

fails using the first method, **fenced**, the fence daemon, tries the next method, and continues to cycle through methods until one succeeds.

Sometimes, fencing a node requires disabling two I/O paths or two power ports. This is done by specifying two or more devices within a fence method. **fenced** runs the fence agent once for each fence-device line; all must succeed for fencing to be considered successful.

More complex configurations are shown in *the section called "Fencing Configuration Examples"*.

You can find more information about configuring specific fence devices from a fence-device agent man page (for example, the man page for **fence_apc**). In addition, you can get more information about fencing parameters from *Appendix A, Fence Device Parameters*, the fence agents in **/usr/sbin/**, the cluster schema at **/usr/share/cluster/cluster.rng**, and the annotated schema at **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** (for example, **/usr/share/doc/cman-3.0.12/cluster_conf.html**).

Fencing Configuration Examples

The following examples show a simple configuration with one fence method per node and one fence device per fence method:

- *Example 7.3, "APC Fence Device Added to **cluster.conf**"*
- *Example 7.4, "Fence Methods Added to **cluster.conf**"*

The following examples show more complex configurations:

- *Example 7.5, "**cluster.conf**: Multiple Fence Methods per Node"*
- *Example 7.6, "**cluster.conf**: Fencing, Multipath Multiple Ports"*
- *Example 7.7, "**cluster.conf**: Fencing Nodes with Dual Power Supplies"*

Note

The examples in this section are not exhaustive; that is, there may be other ways to configure fencing depending on your requirements.

Example 7.3. APC Fence Device Added to **cluster.conf**

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
         </fence>
```

```
      </clusternode>
   </clusternodes>
   <fencedevices>
         <fencedevice agent="fence_apc" ipaddr="apc_ip_example" login="login_example"
 name="apc" passwd="password_example"/>
   </fencedevices>
   <rm>
   </rm>
</cluster>
```

In this example, a fence device (**fencedevice**) has been added to the **fencedevices** element, specifying the fence agent (**agent**) as **fence_apc**, the IP address (**ipaddr**) as **apc_ip_example**, the login (**login**) as **login_example**, the name of the fence device (**name**) as **apc**, and the password (**passwd**) as **password_example**.

Example 7.4. Fence Methods Added to **cluster.conf**

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC">
              <device name="apc" port="1"/>
             </method>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="APC">
              <device name="apc" port="2"/>
            </method>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
            <method name="APC">
              <device name="apc" port="3"/>
            </method>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
         <fencedevice agent="fence_apc" ipaddr="apc_ip_example" login="login_example"
 name="apc" passwd="password_example"/>
   </fencedevices>
   <rm>
   </rm>
</cluster>
```

In this example, a fence method (**method**) has been added to each node. The name of the fence method (**name**) for each node is **APC**. The device (**device**) for the fence method in each node specifies the name (**name**) as **apc** and a unique APC switch power port number (**port**) for each node. For example, the port number for node-01.example.com is **1** (**port="1"**). The device name for each node (**device name="apc"**) points to the fence device by the name (**name**) of **apc** in this line of the **fencedevices** element: **fencedevice agent="fence_apc" ipaddr="apc_ip_example" login="login_example" name="apc" passwd="password_example"**.

Example 7.5. **cluster.conf**: Multiple Fence Methods per Node

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC">
              <device name="apc" port="1"/>
             </method>
            <method name="SAN">
       <device name="sanswitch1" port="11"/>
             </method>
         </fence>
         <unfence>
             <device name="sanswitch1" port="11" action="on"/>
         </unfence
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="APC">
              <device name="apc" port="2"/>
            </method>
            <method name="SAN">
       <device name="sanswitch1" port="12"/>
             </method>
         </fence>
         <unfence>
             <device name="sanswitch1" port="12" action="on"/>
         </unfence
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
            <method name="APC">
              <device name="apc" port="3"/>
            </method>
            <method name="SAN">
       <device name="sanswitch1" port="13"/>
             </method>
         </fence>
         <unfence>
             <device name="sanswitch1" port="13" action="on"/>
         </unfence
     </clusternode>
   </clusternodes>
   <fencedevices>
        <fencedevice agent="fence_apc" ipaddr="apc_ip_example" login="login_example"
 name="apc" passwd="password_example"/>
        <fencedevice agent="fence_sanbox2" ipaddr="san_ip_example"
login="login_example" name="sanswitch1" passwd="password_example"/>
   </fencedevices>
   <rm>
   </rm>
</cluster>
```

Example 7.6. **cluster.conf**: Fencing, Multipath Multiple Ports

```
<cluster name="mycluster" config_version="3">
```

```
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="SAN-multi">
       <device name="sanswitch1" port="11"/>
       <device name="sanswitch2" port="11"/>
     </method>
         </fence>
         <unfence>
             <device name="sanswitch1" port="11" action="on"/>
             <device name="sanswitch2" port="11" action="on"/>
         </unfence
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="SAN-multi">
       <device name="sanswitch1" port="12"/>
       <device name="sanswitch2" port="12"/>
             </method>
         </fence>
         <unfence>
             <device name="sanswitch1" port="12" action="on"/>
             <device name="sanswitch2" port="12" action="on"/>
         </unfence
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
            <method name="SAN-multi">
       <device name="sanswitch1" port="13"/>
       <device name="sanswitch2" port="13"/>
             </method>
         </fence>
         <unfence>
             <device name="sanswitch1" port="13" action="on"/>
             <device name="sanswitch2" port="13" action="on"/>
         </unfence
     </clusternode>
   </clusternodes>
   <fencedevices>
        <fencedevice agent="fence_sanbox2" ipaddr="san_ip_example"
login="login_example" name="sanswitch1" passwd="password_example"/>
        <fencedevice agent="fence_sanbox2" ipaddr="san_ip_example"
login="login_example" name="sanswitch2" passwd="password_example"/>
   </fencedevices>
   <rm>
   </rm>
</cluster>
```

Example 7.7. **`cluster.conf`**: Fencing Nodes with Dual Power Supplies

```


<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC-dual">
              <device name="apc1" port="1"action="off"/>
              <device name="apc2" port="1"action="off"/>
              <device name="apc1" port="1"action="on"/>
              <device name="apc2" port="1"action="on"/>
             </method>
         </fence>
```

```
    </clusternode>
    <clusternode name="node-02.example.com" nodeid="2">
        <fence>
           <method name="APC-dual">
             <device name="apc1" port="2"action="off"/>
             <device name="apc2" port="2"action="off"/>
             <device name="apc1" port="2"action="on"/>
             <device name="apc2" port="2"action="on"/>
           </method>
        </fence>
    </clusternode>
    <clusternode name="node-03.example.com" nodeid="3">
        <fence>
           <method name="APC-dual">
             <device name="apc1" port="3"action="off"/>
             <device name="apc2" port="3"action="off"/>
             <device name="apc1" port="3"action="on"/>
             <device name="apc2" port="3"action="on"/>
           </method>
        </fence>
    </clusternode>
  </clusternodes>
  <fencedevices>
      <fencedevice agent="fence_apc" ipaddr="apc_ip_example" login="login_example"
 name="apc1" passwd="password_example"/>
      <fencedevice agent="fence_apc" ipaddr="apc_ip_example" login="login_example"
 name="apc2" passwd="password_example"/>
  </fencedevices>
  <rm>
  </rm>
</cluster>
```

When using power switches to fence nodes with dual power supplies, the agents must be told to turn off both power ports before restoring power to either port. The default off-on behavior of the agent could result in the power never being fully disabled to the node.

7.4. Configuring Failover Domains

A failover domain is a named subset of cluster nodes that are eligible to run a cluster service in the event of a node failure. A failover domain can have the following characteristics:

- Unrestricted — Allows you to specify that a subset of members are preferred, but that a cluster service assigned to this domain can run on any available member.
- Restricted — Allows you to restrict the members that can run a particular cluster service. If none of the members in a restricted failover domain are available, the cluster service cannot be started (either manually or by the cluster software).
- Unordered — When a cluster service is assigned to an unordered failover domain, the member on which the cluster service runs is chosen from the available failover domain members with no priority ordering.
- Ordered — Allows you to specify a preference order among the members of a failover domain. Ordered failover domains select the node with the lowest priority number first. That is, the node in a failover domain with a priority number of "1" specifies the highest priority, and therefore is the most preferred node in a failover domain. After that node, the next preferred node would be the node with the next highest priority number, and so on.

- Failback — Allows you to specify whether a service in the failover domain should fail back to the node that it was originally running on before that node failed. Configuring this characteristic is useful in circumstances where a node repeatedly fails and is part of an ordered failover domain. In that circumstance, if a node is the preferred node in a failover domain, it is possible for a service to fail over and fail back repeatedly between the preferred node and another node, causing severe impact on performance.

Note

The failback characteristic is applicable only if ordered failover is configured.

Note

Changing a failover domain configuration has no effect on currently running services.

Note

Failover domains are *not* required for operation.

By default, failover domains are unrestricted and unordered.

In a cluster with several members, using a restricted failover domain can minimize the work to set up the cluster to run a cluster service (such as **`httpd`**), which requires you to set up the configuration identically on all members that run the cluster service. Instead of setting up the entire cluster to run the cluster service, you can set up only the members in the restricted failover domain that you associate with the cluster service.

Note

To configure a preferred member, you can create an unrestricted failover domain comprising only one cluster member. Doing that causes a cluster service to run on that cluster member primarily (the preferred member), but allows the cluster service to fail over to any of the other members.

To configure a failover domain, use the following procedures:

1. Open **`/etc/cluster/cluster.conf`** at any node in the cluster.
2. Add the following skeleton section within the **`rm`** element for each failover domain to be used:

```
        <failoverdomains>
            <failoverdomain name="" nofailback="" ordered="" restricted="">
                <failoverdomainnode name="" priority=""/>
                <failoverdomainnode name="" priority=""/>
                <failoverdomainnode name="" priority=""/>
            </failoverdomain>
        </failoverdomains>
```

Note

The number of **failoverdomainnode** attributes depends on the number of nodes in the failover domain. The skeleton **failoverdomain** section in preceding text shows three **failoverdomainnode** elements (with no node names specified), signifying that there are three nodes in the failover domain.

3. In the **failoverdomain** section, provide the values for the elements and attributes. For descriptions of the elements and attributes, refer to the *failoverdomain* section of the annotated cluster schema. The annotated cluster schema is available at **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** (for example **/usr/share/doc/cman-3.0.12/cluster_conf.html**) in any of the cluster nodes. For an example of a **failoverdomains** section, refer to *Example 7.8, "A Failover Domain Added to **cluster.conf**"*.
4. Update the **config_version** attribute by incrementing its value (for example, changing from **config_version="2"** to **config_version="3">**).
5. Save **/etc/cluster/cluster.conf**.
6. **(Optional)** Validate the file with against the cluster schema (**cluster.rng**) by running the **ccs_config_validate** command. For example:

```
[root@example-01 ~]# ccs_config_validate
Configuration validates
```

7. Run the **cman_tool version -r** command to propagate the configuration to the rest of the cluster nodes.
8. Proceed to *Section 7.5, "Configuring HA Services"*.

*Example 7.8, "A Failover Domain Added to **cluster.conf**"* shows an example of a configuration with an ordered, unrestricted failover domain.

Example 7.8. A Failover Domain Added to **cluster.conf**

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC">
              <device name="apc" port="1"/>
```

```
            </method>
        </fence>
    </clusternode>
    <clusternode name="node-02.example.com" nodeid="2">
        <fence>
           <method name="APC">
             <device name="apc" port="2"/>
           </method>
        </fence>
    </clusternode>
    <clusternode name="node-03.example.com" nodeid="3">
        <fence>
           <method name="APC">
             <device name="apc" port="3"/>
           </method>
        </fence>
    </clusternode>
  </clusternodes>
  <fencedevices>
        <fencedevice agent="fence_apc" ipaddr="apc_ip_example" login="login_example"
 name="apc" passwd="password_example"/>
  </fencedevices>
  <rm>
      <failoverdomains>
          <failoverdomain name="example_pri" nofailback="0" ordered="1" restricted="0">
              <failoverdomainnode name="node-01.example.com" priority="1"/>
              <failoverdomainnode name="node-02.example.com" priority="2"/>
              <failoverdomainnode name="node-03.example.com" priority="3"/>
          </failoverdomain>
      </failoverdomains>
  </rm>
</cluster>
```

The **failoverdomains** section contains a **failoverdomain** section for each failover domain in the cluster. This example has one failover domain. In the **failoverdomain** line, the name (**name**) is specified as **example_pri**. In addition, it specifies no failback (**failback="0"**), that failover is ordered (**ordered="1"**), and that the failover domain is unrestricted (**restricted="0"**).

7.5. Configuring HA Services

Configuring HA (High Availability) services consists of configuring resources and assigning them to services.

The following sections describe how to edit **/etc/cluster/cluster.conf** to add resources and services.

- *Section 7.5.1, “Adding Cluster Resources”*
- *Section 7.5.2, “Adding a Cluster Service to the Cluster”*

Important

There can be a wide range of configurations possible with High Availability resources and services. For a better understanding about resource parameters and resource behavior, refer to *Appendix B, HA Resource Parameters* and *Appendix C, HA Resource Behavior*. For optimal performance and to ensure that your configuration can be supported, contact an authorized Red Hat support representative.

7.5.1. Adding Cluster Resources

You can configure two types of resources:

- Global — Resources that are available to any service in the cluster. These are configured in the **resources** section of the configuration file (within the **rm** element).
- Service-specific — Resources that are available to only one service. These are configured in each **service** section of the configuration file (within the **rm** element).

This section describes how to add a global resource. For procedures about configuring service-specific resources, refer to *Section 7.5.2, "Adding a Cluster Service to the Cluster"*.

To add a global cluster resource, follow the steps in this section.

1. Open **/etc/cluster/cluster.conf** at any node in the cluster.
2. Add a **resources** section within the **rm** element. For example:

```
    <rm>
        <resources>

        </resources>
    </rm>
```

3. Populate it with resources according to the services you want to create. For example, here are resources that are to be used in an Apache service. They consist of a file system (**fs**) resource, an IP (**ip**) resource, and an Apache (**apache**) resource.

```
    <rm>
        <resources>
           <fs name="web_fs" device="/dev/sdd2" mountpoint="/var/www" fstype="ext3"/>
           <ip address="127.143.131.100" monitor_link="yes" sleeptime="10"/>
           <apache config_file="conf/httpd.conf" name="example_server" server_root="/etc/
httpd" shutdown_wait="0"/>
        </resources>
    </rm>
```

*Example 7.9, "**cluster.conf** File with Resources Added "* shows an example of a **cluster.conf** file with the **resources** section added.

4. Update the **config_version** attribute by incrementing its value (for example, changing from **config_version="2"** to **config_version="3"**).

5. Save **/etc/cluster/cluster.conf**.

6. **(Optional)** Validate the file with against the cluster schema (**cluster.rng**) by running the **ccs_config_validate** command. For example:

```
[root@example-01 ~]# ccs_config_validate
Configuration validates
```

7. Run the **cman_tool version -r** command to propagate the configuration to the rest of the cluster nodes.

8. Verify that the updated configuration file has been propagated.

9. Proceed to *Section 7.5.2, "Adding a Cluster Service to the Cluster"*.

Example 7.9. **cluster.conf** File with Resources Added

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC">
              <device name="apc" port="1"/>
             </method>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="APC">
              <device name="apc" port="2"/>
            </method>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
            <method name="APC">
              <device name="apc" port="3"/>
            </method>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
         <fencedevice agent="fence_apc" ipaddr="apc_ip_example" login="login_example"
 name="apc" passwd="password_example"/>
   </fencedevices>
   <rm>
       <failoverdomains>
           <failoverdomain name="example_pri" nofailback="0" ordered="1" restricted="0">
               <failoverdomainnode name="node-01.example.com" priority="1"/>
               <failoverdomainnode name="node-02.example.com" priority="2"/>
               <failoverdomainnode name="node-03.example.com" priority="3"/>
           </failoverdomain>
       </failoverdomains>
       <resources>
           <fs name="web_fs" device="/dev/sdd2" mountpoint="/var/www" fstype="ext3"/>
           <ip address="127.143.131.100" monitor_link="yes" sleeptime="10"/>
```

```
          <apache config_file="conf/httpd.conf" name="example_server" server_root="/etc/
httpd" shutdown_wait="0"/>
        </resources>

   </rm>
</cluster>
```

7.5.2. Adding a Cluster Service to the Cluster

To add a cluster service to the cluster, follow the steps in this section.

1. Open **/etc/cluster/cluster.conf** at any node in the cluster.
2. Add a **service** section within the **rm** element for each service. For example:

```
    <rm>
        <service autostart="1" domain="" exclusive="0" name="" recovery="restart">

        </service>
    </rm>
```

3. Configure the following parameters (attributes) in the **service** element:
   - **autostart** — Specifies whether to autostart the service when the cluster starts. Use '1' to enable and '0' to disable; the default is enabled.
   - **domain** — Specifies a failover domain (if required).
   - **exclusive** — Specifies a policy wherein the service only runs on nodes that have no other services running on them.
   - **recovery** — Specifies a recovery policy for the service. The options are to relocate, restart, disable, or restart-disable the service.
4. Depending on the type of resources you want to use, populate the service with global or service-specific resources

   For example, here is an Apache service that uses global resources:

```
    <rm>
        <resources>
                <fs name="web_fs" device="/dev/sdd2" mountpoint="/var/www" fstype="ext3"/
>
                <ip address="127.143.131.100" monitor_link="yes" sleeptime="10"/>
                <apache config_file="conf/httpd.conf" name="example_server"
 server_root="/etc/httpd" shutdown_wait="0"/>
        </resources>
        <service autostart="1" domain="example_pri" exclusive="0" name="example_apache"
 recovery="relocate">
                <fs ref="web_fs"/>
                <ip ref="127.143.131.100"/>
                <apache ref="example_server"/>
        </service>
```

```
    </rm>
```

For example, here is an Apache service that uses service-specific resources:

```
   <rm>
       <service autostart="0" domain="example_pri" exclusive="0" name="example_apache2"
recovery="relocate">
               <fs name="web_fs2" device="/dev/sdd3" mountpoint="/var/www2"
fstype="ext3"/>
               <ip address="127.143.131.101" monitor_link="yes" sleeptime="10"/>
               <apache config_file="conf/httpd.conf" name="example_server2"
server_root="/etc/httpd" shutdown_wait="0"/>
       </service>
   </rm>
```

*Example 7.10, "**cluster.conf** with Services Added: One Using Global Resources and One Using Service-Specific Resources "* shows an example of a **cluster.conf** file with two services:

- **example_apache** — This service uses global resources **web_fs**, **127.143.131.100**, and **example_server**.
- **example_apache2** — This service uses service-specific resources **web_fs2**, **127.143.131.101**, and **example_server2**.

5. Update the **config_version** attribute by incrementing its value (for example, changing from **config_version="2"** to **config_version="3">**).
6. Save **/etc/cluster/cluster.conf**.
7. **(Optional)** Validate the updated file against the cluster schema (**cluster.rng**) by running the **ccs_config_validate** command. For example:

```
[root@example-01 ~]# ccs_config_validate
Configuration validates
```

8. Run the **cman_tool version -r** command to propagate the configuration to the rest of the cluster nodes.
9. Verify that the updated configuration file has been propagated.
10. Proceed to *Section 7.6, "Verifying a Configuration"*.

Example 7.10. **cluster.conf** with Services Added: One Using Global Resources and One Using Service-Specific Resources

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC">
              <device name="apc" port="1"/>
             </method>
```

```
        </fence>
    </clusternode>
    <clusternode name="node-02.example.com" nodeid="2">
        <fence>
           <method name="APC">
             <device name="apc" port="2"/>
           </method>
        </fence>
    </clusternode>
    <clusternode name="node-03.example.com" nodeid="3">
        <fence>
           <method name="APC">
             <device name="apc" port="3"/>
           </method>
        </fence>
    </clusternode>
  </clusternodes>
  <fencedevices>
        <fencedevice agent="fence_apc" ipaddr="apc_ip_example" login="login_example"
 name="apc" passwd="password_example"/>
  </fencedevices>
  <rm>
      <failoverdomains>
          <failoverdomain name="example_pri" nofailback="0" ordered="1" restricted="0">
              <failoverdomainnode name="node-01.example.com" priority="1"/>
              <failoverdomainnode name="node-02.example.com" priority="2"/>
              <failoverdomainnode name="node-03.example.com" priority="3"/>
          </failoverdomain>
      </failoverdomains>
      <resources>
          <fs name="web_fs" device="/dev/sdd2" mountpoint="/var/www" fstype="ext3"/>
          <ip address="127.143.131.100" monitor_link="yes" sleeptime="10"/>
          <apache config_file="conf/httpd.conf" name="example_server" server_root="/etc/
httpd" shutdown_wait="0"/>
      </resources>
      <service autostart="1" domain="example_pri" exclusive="0" name="example_apache"
 recovery="relocate">
          <fs ref="web_fs"/>
          <ip ref="127.143.131.100"/>
          <apache ref="example_server"/>
      </service>
      <service autostart="0" domain="example_pri" exclusive="0" name="example_apache2"
 recovery="relocate">
          <fs name="web_fs2" device="/dev/sdd3" mountpoint="/var/www2" fstype="ext3"/>
          <ip address="127.143.131.101" monitor_link="yes" sleeptime="10"/>
          <apache config_file="conf/httpd.conf" name="example_server2" server_root="/etc/
httpd" shutdown_wait="0"/>
      </service>
  </rm>
</cluster>
```

7.6. Verifying a Configuration

Once you have created your cluster configuration file, verify that it is running correctly by performing the following steps:

1. At each node, restart the cluster software. That action ensures that any configuration additions that are checked only at startup time are included in the running configuration. You can restart the cluster software by running **`service cman restart`**. For example:

```
[root@example-01 ~]# service cman restart
Stopping cluster:
```

```
   Leaving fence domain...                                 [  OK  ]
   Stopping gfs_controld...                                [  OK  ]
   Stopping dlm_controld...                                [  OK  ]
   Stopping fenced...                                      [  OK  ]
   Stopping cman...                                        [  OK  ]
   Waiting for corosync to shutdown:                       [  OK  ]
   Unloading kernel modules...                             [  OK  ]
   Unmounting configfs...                                  [  OK  ]
Starting cluster:
   Checking Network Manager...                             [  OK  ]
   Global setup...                                         [  OK  ]
   Loading kernel modules...                               [  OK  ]
   Mounting configfs...                                    [  OK  ]
   Starting cman...                                        [  OK  ]
   Waiting for quorum...                                   [  OK  ]
   Starting fenced...                                      [  OK  ]
   Starting dlm_controld...                                [  OK  ]
   Starting gfs_controld...                                [  OK  ]
   Unfencing self...                                       [  OK  ]
   Joining fence domain...                                 [  OK  ]
```

2. Run **`service clvmd start`**, if CLVM is being used to create clustered volumes. For example:

```
[root@example-01 ~]# service clvmd start
Activating VGs:                                            [  OK  ]
```

3. Run **`service gfs2 start`**, if you are using Red Hat GFS2. For example:

```
[root@example-01 ~]# service gfs2 start
Mounting GFS2 filesystem (/mnt/gfsA):                      [  OK  ]
Mounting GFS2 filesystem (/mnt/gfsB):                      [  OK  ]
```

4. Run **`service rgmanager start`**, if you using high-availability (HA) services. For example:

```
[root@example-01 ~]# service rgmanager start
Starting Cluster Service Manager:                          [  OK  ]
```

5. At any cluster node, run **`cman_tool nodes`** to verify that the nodes are functioning as members in the cluster (signified as "M" in the status column, "Sts"). For example:

```
[root@example-01 ~]# cman_tool nodes
Node  Sts   Inc   Joined               Name
   1   M    548   2010-09-28 10:52:21  node-01.example.com
   2   M    548   2010-09-28 10:52:21  node-02.example.com
   3   M    544   2010-09-28 10:52:21  node-03.example.com
```

6. At any node, using the **`clustat`** utility, verify that the HA services are running as expected. In addition, **`clustat`** displays status of the cluster nodes. For example:

```
[root@example-01 ~]#clustat
Cluster Status for mycluster @ Wed Nov 17 05:40:00 2010
Member Status: Quorate

 Member Name                             ID   Status
```

```
------ ----                              ---- ------
node-03.example.com                          3 Online, rgmanager
node-02.example.com                          2 Online, rgmanager
node-01.example.com                          1 Online, Local, rgmanager

Service Name                   Owner (Last)                   State
------- ----                   ----- ------                   -----
service:example_apache         node-01.example.com            started
service:example_apache2        (none)                         disabled
```

7. If the cluster is running as expected, you are done with creating a configuration file. You can manage the cluster with command-line tools described in *Chapter 8, Managing Red Hat High Availability Add-On With Command Line Tools*.

Managing Red Hat High Availability Add-On With Command Line Tools

This chapter describes various administrative tasks for managing Red Hat High Availability Add-On and consists of the following sections:

- *Section 8.1, "Starting and Stopping the Cluster Software"*
- *Section 8.2, "Deleting or Adding a Node"*
- *Section 8.3, "Managing High-Availability Services"*
- *Section 8.4, "Updating a Configuration"*

Important

Make sure that your deployment of Red Hat High Availability Add-On meets your needs and can be supported. Consult with an authorized Red Hat representative to verify your configuration prior to deployment. In addition, allow time for a configuration burn-in period to test failure modes.

Important

This chapter references commonly used **`cluster.conf`** elements and attributes. For a comprehensive list and description of **`cluster.conf`** elements and attributes, refer to the cluster schema at **`/usr/share/cluster/cluster.rng`**, and the annotated schema at **`/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html`** (for example **`/usr/share/doc/cman-3.0.12/cluster_conf.html`**).

Important

Certain procedure in this chapter call for using the **`cman_tool version -r`** command to propagate a cluster configuration throughout a cluster. Using that command requires that **`ricci`** is running.

Note

Procedures in this chapter, may include specific commands for some of the command-line tools listed in *Appendix E, Command Line Tools Summary*. For more information about all commands and variables, refer to the man page for each command-line tool.

8.1. Starting and Stopping the Cluster Software

You can start or stop cluster software on a node according to *Section 8.1.1, "Starting Cluster Software"* and *Section 8.1.2, "Stopping Cluster Software"*. Starting cluster software on a node causes it to join the cluster; stopping the cluster software on a node causes it to leave the cluster.

8.1.1. Starting Cluster Software

To start the cluster software on a node, type the following commands in this order:

1. **`service cman start`**
2. **`service clvmd start`**, if CLVM has been used to create clustered volumes
3. **`service gfs2 start`**, if you are using Red Hat GFS2
4. **`service rgmanager start`**, if you using high-availability (HA) services (**`rgmanager`**).

For example:

```
[root@example-01 ~]# service cman start
Starting cluster:
   Checking Network Manager...                             [  OK  ]
   Global setup...                                         [  OK  ]
   Loading kernel modules...                               [  OK  ]
   Mounting configfs...                                    [  OK  ]
   Starting cman...                                        [  OK  ]
   Waiting for quorum...                                   [  OK  ]
   Starting fenced...                                      [  OK  ]
   Starting dlm_controld...                                [  OK  ]
   Starting gfs_controld...                                [  OK  ]
   Unfencing self...                                       [  OK  ]
   Joining fence domain...                                 [  OK  ]
[root@example-01 ~]# service clvmd start
Starting clvmd:                                            [  OK  ]
Activating VG(s):   2 logical volume(s) in volume group "vg_example" now active
                                                           [  OK  ]
[root@example-01 ~]# service gfs2 start
Mounting GFS2 filesystem (/mnt/gfsA):                      [  OK  ]
Mounting GFS2 filesystem (/mnt/gfsB):                      [  OK  ]
[root@example-01 ~]# service rgmanager start
Starting Cluster Service Manager:                          [  OK  ]
[root@example-01 ~]#
```

8.1.2. Stopping Cluster Software

To stop the cluster software on a node, type the following commands in this order:

1. **`service rgmanager stop`**, if you using high-availability (HA) services (**`rgmanager`**).
2. **`service gfs2 stop`**, if you are using Red Hat GFS2
3. **`umount -at gfs2`**, if you are using Red Hat GFS2 in conjunction with **`rgmanager`**, to ensure that any GFS2 files mounted during **`rgmanager`** startup (but not unmounted during shutdown) were also unmounted.
4. **`service clvmd stop`**, if CLVM has been used to create clustered volumes
5. **`service cman stop`**

For example:

```
[root@example-01 ~]# service rgmanager stop
Stopping Cluster Service Manager:                          [  OK  ]
[root@example-01 ~]# service gfs2 stop
Unmounting GFS2 filesystem (/mnt/gfsA):                    [  OK  ]
Unmounting GFS2 filesystem (/mnt/gfsB):                    [  OK  ]
[root@example-01 ~]# umount -at gfs2
[root@example-01 ~]# service clvmd stop
Signaling clvmd to exit                                    [  OK  ]
clvmd terminated                                           [  OK  ]
[root@example-01 ~]# service cman stop
Stopping cluster:
   Leaving fence domain...                                 [  OK  ]
   Stopping gfs_controld...                                [  OK  ]
   Stopping dlm_controld...                                [  OK  ]
   Stopping fenced...                                      [  OK  ]
   Stopping cman...                                        [  OK  ]
   Waiting for corosync to shutdown:                       [  OK  ]
   Unloading kernel modules...                             [  OK  ]
   Unmounting configfs...                                  [  OK  ]
[root@example-01 ~]#
```

Note

Stopping cluster software on a node causes its HA services to fail over to another node. As an alternative to that, consider relocating or migrating HA services to another node before stopping cluster software. For information about managing HA services, refer to *Section 8.3, "Managing High-Availability Services"*.

8.2. Deleting or Adding a Node

This section describes how to delete a node from a cluster and add a node to a cluster. You can delete a node from a cluster according to *Section 8.2.1, "Deleting a Node from a Cluster"*; you can add a node to a cluster according to *Section 8.2.2, "Adding a Node to a Cluster"*.

8.2.1. Deleting a Node from a Cluster

Deleting a node from a cluster consists of shutting down the cluster software on the node to be deleted and updating the cluster configuration to reflect the change.

Important

If deleting a node from the cluster causes a transition from greater than two nodes to two nodes, you must restart the cluster software at each node after updating the cluster configuration file.

To delete a node from a cluster, perform the following steps:

1. At any node, use the **clusvcadm** utility to relocate, migrate, or stop each HA service running on the node that is being deleted from the cluster. For information about using **clusvcadm**, refer to *Section 8.3, "Managing High-Availability Services"*.

2. At the node to be deleted from the cluster, stop the cluster software according to *Section 8.1.2, "Stopping Cluster Software"*. For example:

```
[root@example-01 ~]# service rgmanager stop
Stopping Cluster Service Manager:                          [  OK  ]
[root@example-01 ~]# service gfs2 stop
Unmounting GFS2 filesystem (/mnt/gfsA):                    [  OK  ]
Unmounting GFS2 filesystem (/mnt/gfsB):                    [  OK  ]
[root@example-01 ~]# service clvmd stop
Signaling clvmd to exit                                    [  OK  ]
clvmd terminated                                           [  OK  ]
[root@example-01 ~]# service cman stop
Stopping cluster:
   Leaving fence domain...                                 [  OK  ]
   Stopping gfs_controld...                                [  OK  ]
   Stopping dlm_controld...                                [  OK  ]
   Stopping fenced...                                      [  OK  ]
   Stopping cman...                                        [  OK  ]
   Waiting for corosync to shutdown:                       [  OK  ]
   Unloading kernel modules...                             [  OK  ]
   Unmounting configfs...                                  [  OK  ]
[root@example-01 ~]#
```

3. At any node in the cluster, edit the **/etc/cluster/cluster.conf** to remove the **clusternode** section of the node that is to be deleted. For example, in *Example 8.1, "Three-node Cluster Configuration"*, if node-03.example.com is supposed to be removed, then delete the **clusternode** section for that node. If removing a node (or nodes) causes the cluster to be a two-node cluster, you can add the following line to the configuration file to allow a single node to maintain quorum (for example, if one node fails):

   <cman two_node="1" expected_votes="1"/>

   Refer to *Section 8.2.3, "Examples of Three-Node and Two-Node Configurations"* for comparison between a three-node and a two-node configuration.

4. Update the **config_version** attribute by incrementing its value (for example, changing from **config_version="2"** to **config_version="3">**).

5. Save **/etc/cluster/cluster.conf**.

6. **(Optional)** Validate the updated file against the cluster schema (**cluster.rng**) by running the **ccs_config_validate** command. For example:

```
[root@example-01 ~]# ccs_config_validate
Configuration validates
```

7. Run the **cman_tool version -r** command to propagate the configuration to the rest of the cluster nodes.

8. Verify that the updated configuration file has been propagated.

9. If the node count of the cluster has transitioned from greater than two nodes to two nodes, you must restart the cluster software as follows:

a. At each node, stop the cluster software according to *Section 8.1.2, "Stopping Cluster Software"*. For example:

```
[root@example-01 ~]# service rgmanager stop
Stopping Cluster Service Manager:                          [  OK  ]
[root@example-01 ~]# service gfs2 stop
Unmounting GFS2 filesystem (/mnt/gfsA):                    [  OK  ]
Unmounting GFS2 filesystem (/mnt/gfsB):                    [  OK  ]
[root@example-01 ~]# service clvmd stop
Signaling clvmd to exit                                    [  OK  ]
clvmd terminated                                           [  OK  ]
[root@example-01 ~]# service cman stop
Stopping cluster:
   Leaving fence domain...                                 [  OK  ]
   Stopping gfs_controld...                                [  OK  ]
   Stopping dlm_controld...                                [  OK  ]
   Stopping fenced...                                      [  OK  ]
   Stopping cman...                                        [  OK  ]
   Waiting for corosync to shutdown:                       [  OK  ]
   Unloading kernel modules...                             [  OK  ]
   Unmounting configfs...                                  [  OK  ]
[root@example-01 ~]#
```

b. At each node, start the cluster software according to *Section 8.1.1, "Starting Cluster Software"*. For example:

```
[root@example-01 ~]# service cman start
Starting cluster:
   Checking Network Manager...                             [  OK  ]
   Global setup...                                         [  OK  ]
   Loading kernel modules...                               [  OK  ]
   Mounting configfs...                                    [  OK  ]
   Starting cman...                                        [  OK  ]
   Waiting for quorum...                                   [  OK  ]
   Starting fenced...                                      [  OK  ]
   Starting dlm_controld...                                [  OK  ]
   Starting gfs_controld...                                [  OK  ]
   Unfencing self...                                       [  OK  ]
   Joining fence domain...                                 [  OK  ]
[root@example-01 ~]# service clvmd start
Starting clvmd:                                            [  OK  ]
Activating VG(s):   2 logical volume(s) in volume group "vg_example" now active
                                                           [  OK  ]
[root@example-01 ~]# service gfs2 start
Mounting GFS2 filesystem (/mnt/gfsA):                      [  OK  ]
Mounting GFS2 filesystem (/mnt/gfsB):                      [  OK  ]
[root@example-01 ~]# service rgmanager start
Starting Cluster Service Manager:                          [  OK  ]
[root@example-01 ~]#
```

c. At any cluster node, run **cman_tool nodes** to verify that the nodes are functioning as members in the cluster (signified as "M" in the status column, "Sts"). For example:

```
[root@example-01 ~]# cman_tool nodes
Node  Sts   Inc   Joined               Name
   1   M    548   2010-09-28 10:52:21  node-01.example.com
   2   M    548   2010-09-28 10:52:21  node-02.example.com
```

d. At any node, using the **clustat** utility, verify that the HA services are running as expected. In addition, **clustat** displays status of the cluster nodes. For example:

```
[root@example-01 ~]#clustat
Cluster Status for mycluster @ Wed Nov 17 05:40:00 2010
Member Status: Quorate

 Member Name                             ID   Status
 ------ ----                             ---- ------
 node-02.example.com                          2 Online, rgmanager
 node-01.example.com                          1 Online, Local, rgmanager

 Service Name                   Owner (Last)                   State
 ------- ----                   ----- ------                   -----
 service:example_apache         node-01.example.com            started
 service:example_apache2        (none)                         disabled
```

8.2.2. Adding a Node to a Cluster

Adding a node to a cluster consists of updating the cluster configuration, propagating the updated configuration to the node to be added, and starting the cluster software on that node. To add a node to a cluster, perform the following steps:

1. At any node in the cluster, edit the **/etc/cluster/cluster.conf** to add a **clusternode** section for the node that is to be added. For example, in *Example 8.2, "Two-node Cluster Configuration"*, if node-03.example.com is supposed to be added, then add a **clusternode** section for that node. If adding a node (or nodes) causes the cluster to transition from a two-node cluster to a cluster with three or more nodes, remove the following **cman** attributes from **/etc/cluster/cluster.conf**:

   - **cman two_node="1"**
   - **expected_votes="1"**

   Refer to *Section 8.2.3, "Examples of Three-Node and Two-Node Configurations"* for comparison between a three-node and a two-node configuration.

2. Update the **config_version** attribute by incrementing its value (for example, changing from **config_version="2"** to **config_version="3">**).

3. Save **/etc/cluster/cluster.conf**.

4. **(Optional)** Validate the updated file against the cluster schema (**cluster.rng**) by running the **ccs_config_validate** command. For example:

```
[root@example-01 ~]# ccs_config_validate
Configuration validates
```

5. Run the **cman_tool version -r** command to propagate the configuration to the rest of the cluster nodes.

6. Verify that the updated configuration file has been propagated.

7. Propagate the updated configuration file to **/etc/cluster/** in each node to be added to the cluster. For example, use the **scp** command to send the updated configuration file to each node to be added to the cluster.

8. If the node count of the cluster has transitioned from two nodes to greater than two nodes, you must restart the cluster software in the existing cluster nodes as follows:

   a. At each node, stop the cluster software according to *Section 8.1.2, "Stopping Cluster Software"*. For example:

```
[root@example-01 ~]# service rgmanager stop
Stopping Cluster Service Manager:                          [  OK  ]
[root@example-01 ~]# service gfs2 stop
Unmounting GFS2 filesystem (/mnt/gfsA):                    [  OK  ]
Unmounting GFS2 filesystem (/mnt/gfsB):                    [  OK  ]
[root@example-01 ~]# service clvmd stop
Signaling clvmd to exit                                    [  OK  ]
clvmd terminated                                           [  OK  ]
[root@example-01 ~]# service cman stop
Stopping cluster:
   Leaving fence domain...                                 [  OK  ]
   Stopping gfs_controld...                                [  OK  ]
   Stopping dlm_controld...                                [  OK  ]
   Stopping fenced...                                      [  OK  ]
   Stopping cman...                                        [  OK  ]
   Waiting for corosync to shutdown:                       [  OK  ]
   Unloading kernel modules...                             [  OK  ]
   Unmounting configfs...                                  [  OK  ]
[root@example-01 ~]#
```

   b. At each node, start the cluster software according to *Section 8.1.1, "Starting Cluster Software"*. For example:

```
[root@example-01 ~]# service cman start
Starting cluster:
   Checking Network Manager...                             [  OK  ]
   Global setup...                                         [  OK  ]
   Loading kernel modules...                               [  OK  ]
   Mounting configfs...                                    [  OK  ]
   Starting cman...                                        [  OK  ]
   Waiting for quorum...                                   [  OK  ]
   Starting fenced...                                      [  OK  ]
   Starting dlm_controld...                                [  OK  ]
   Starting gfs_controld...                                [  OK  ]
   Unfencing self...                                       [  OK  ]
   Joining fence domain...                                 [  OK  ]
[root@example-01 ~]# service clvmd start
Starting clvmd:                                            [  OK  ]
Activating VG(s):   2 logical volume(s) in volume group "vg_example" now active
                                                           [  OK  ]
[root@example-01 ~]# service gfs2 start
Mounting GFS2 filesystem (/mnt/gfsA):                      [  OK  ]
Mounting GFS2 filesystem (/mnt/gfsB):                      [  OK  ]
[root@example-01 ~]# service rgmanager start
Starting Cluster Service Manager:                          [  OK  ]
[root@example-01 ~]#
```

9. At each node to be added to the cluster, start the cluster software according to *Section 8.1.1, "Starting Cluster Software"*. For example:

```
[root@example-01 ~]# service cman start
Starting cluster:
   Checking Network Manager...                             [  OK  ]
   Global setup...                                         [  OK  ]
   Loading kernel modules...                               [  OK  ]
   Mounting configfs...                                    [  OK  ]
   Starting cman...                                        [  OK  ]
   Waiting for quorum...                                   [  OK  ]
   Starting fenced...                                      [  OK  ]
   Starting dlm_controld...                                [  OK  ]
   Starting gfs_controld...                                [  OK  ]
   Unfencing self...                                       [  OK  ]
   Joining fence domain...                                 [  OK  ]
[root@example-01 ~]# service clvmd start
Starting clvmd:                                            [  OK  ]
Activating VG(s):   2 logical volume(s) in volume group "vg_example" now active
                                                           [  OK  ]
[root@example-01 ~]# service gfs2 start
Mounting GFS2 filesystem (/mnt/gfsA):                      [  OK  ]
Mounting GFS2 filesystem (/mnt/gfsB):                      [  OK  ]

[root@example-01 ~]# service rgmanager start
Starting Cluster Service Manager:                          [  OK  ]
[root@example-01 ~]#
```

10. At any node, using the **clustat** utility, verify that each added node is running and part of the cluster. For example:

```
[root@example-01 ~]#clustat
Cluster Status for mycluster @ Wed Nov 17 05:40:00 2010
Member Status: Quorate

 Member Name                             ID   Status
 ------ ----                             ---- ------
 node-03.example.com                         3 Online, rgmanager
 node-02.example.com                         2 Online, rgmanager
 node-01.example.com                         1 Online, Local, rgmanager

 Service Name                   Owner (Last)                   State
 ------- ----                   ----- ------                   -----
 service:example_apache         node-01.example.com            started
 service:example_apache2        (none)                         disabled
```

For information about using **clustat**, refer to *Section 8.3, "Managing High-Availability Services"*.

In addition, you can use **cman_tool status** to verify node votes, node count, and quorum count. For example:

```
[root@example-01 ~]#cman_tool status
Version: 6.2.0
Config Version: 19
Cluster Name: mycluster
Cluster Id: 3794
Cluster Member: Yes
Cluster Generation: 548
Membership state: Cluster-Member
Nodes: 3
Expected votes: 3
Total votes: 3
```

```
Node votes: 1
Quorum: 2
Active subsystems: 9
Flags:
Ports Bound: 0 11 177
Node name: node-01.example.com
Node ID: 3
Multicast addresses: 239.192.14.224
Node addresses: 10.15.90.58
```

11. At any node, you can use the **clusvcadm** utility to migrate or relocate a running service to the newly joined node. Also, you can enable any disabled services. For information about using **clusvcadm**, refer to *Section 8.3, "Managing High-Availability Services"*

8.2.3. Examples of Three-Node and Two-Node Configurations

Refer to the examples that follow for comparison between a three-node and a two-node configuration.

Example 8.1. Three-node Cluster Configuration

```
<cluster name="mycluster" config_version="3">
   <cman/>
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC">
              <device name="apc" port="1"/>
             </method>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="APC">
              <device name="apc" port="2"/>
            </method>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
            <method name="APC">
              <device name="apc" port="3"/>
            </method>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
         <fencedevice agent="fence_apc" ipaddr="apc_ip_example" login="login_example"
 name="apc" passwd="password_example"/>
   </fencedevices>
   <rm>
       <failoverdomains>
           <failoverdomain name="example_pri" nofailback="0" ordered="1" restricted="0">
               <failoverdomainnode name="node-01.example.com" priority="1"/>
               <failoverdomainnode name="node-02.example.com" priority="2"/>
               <failoverdomainnode name="node-03.example.com" priority="3"/>
           </failoverdomain>
       </failoverdomains>
       <resources>
           <fs name="web_fs" device="/dev/sdd2" mountpoint="/var/www" fstype="ext3"/>
           <ip address="127.143.131.100" monitor_link="yes" sleeptime="10"/>
           <apache config_file="conf/httpd.conf" name="example_server" server_root="/etc/
httpd" shutdown_wait="0"/>
```

```
        </resources>
        <service autostart="0" domain="example_pri" exclusive="0" name="example_apache"
 recovery="relocate">
            <fs ref="web_fs"/>
            <ip ref="127.143.131.100"/>
            <apache ref="example_server"/>
        </service>
        <service autostart="0" domain="example_pri" exclusive="0" name="example_apache2"
 recovery="relocate">
            <fs name="web_fs2" device="/dev/sdd3" mountpoint="/var/www" fstype="ext3"/>
            <ip address="127.143.131.101" monitor_link="yes" sleeptime="10"/>
            <apache config_file="conf/httpd.conf" name="example_server2" server_root="/etc/
httpd" shutdown_wait="0"/>
        </service>
    </rm>
</cluster>
```

Example 8.2. Two-node Cluster Configuration

```
<cluster name="mycluster" config_version="3">
   <cman two_node="1" expected_votes="1"/>
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC">
              <device name="apc" port="1"/>
             </method>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="APC">
              <device name="apc" port="2"/>
            </method>
         </fence>
   </clusternodes>
   <fencedevices>
         <fencedevice agent="fence_apc" ipaddr="apc_ip_example" login="login_example"
 name="apc" passwd="password_example"/>
   </fencedevices>
   <rm>
       <failoverdomains>
           <failoverdomain name="example_pri" nofailback="0" ordered="1" restricted="0">
               <failoverdomainnode name="node-01.example.com" priority="1"/>
               <failoverdomainnode name="node-02.example.com" priority="2"/>
           </failoverdomain>
       </failoverdomains>
       <resources>
           <fs name="web_fs" device="/dev/sdd2" mountpoint="/var/www" fstype="ext3"/>
           <ip address="127.143.131.100" monitor_link="yes" sleeptime="10"/>
           <apache config_file="conf/httpd.conf" name="example_server" server_root="/etc/
httpd" shutdown_wait="0"/>
       </resources>
       <service autostart="0" domain="example_pri" exclusive="0" name="example_apache"
 recovery="relocate">
           <fs ref="web_fs"/>
           <ip ref="127.143.131.100"/>
           <apache ref="example_server"/>
       </service>
       <service autostart="0" domain="example_pri" exclusive="0" name="example_apache2"
 recovery="relocate">
           <fs name="web_fs2" device="/dev/sdd3" mountpoint="/var/www" fstype="ext3"/>
```

```
            <ip address="127.143.131.101" monitor_link="yes" sleeptime="10"/>
            <apache config_file="conf/httpd.conf" name="example_server2" server_root="/etc/
httpd" shutdown_wait="0"/>
        </service>
    </rm>
</cluster>
```

8.3. Managing High-Availability Services

You can manage high-availability services using the **Cluster Status Utility**, **`clustat`**, and the **Cluster User Service Administration Utility**, **`clusvcadm`**. **`clustat`** displays the status of a cluster and **`clusvcadm`** provides the means to manage high-availability services.

This section provides basic information about managing HA services using the **`clustat`** and **`clusvcadm`** commands. It consists of the following subsections:

- *Section 8.3.1, "Displaying HA Service Status with **`clustat`**"*
- *Section 8.3.2, "Managing HA Services with **`clusvcadm`**"*

8.3.1. Displaying HA Service Status with `clustat`

`clustat` displays cluster-wide status. It shows membership information, quorum view, the state of all high-availability services, and indicates which node the **`clustat`** command is being run at (Local). *Table 8.1, "Services Status"* describes the states that services can be in and are displayed when running **`clustat`**. *Example 8.3, "**`clustat`** Display"* shows an example of a **`clustat`** display. For more detailed information about running the **`clustat`** command refer to the **`clustat`** man page.

Table 8.1. Services Status

Services Status	Description
Started	The service resources are configured and available on the cluster system that owns the service.
Recovering	The service is pending start on another node.
Disabled	The service has been disabled, and does not have an assigned owner. A disabled service is never restarted automatically by the cluster.
Stopped	In the stopped state, the service will be evaluated for starting after the next service or node transition. This is a temporary state. You may disable or enable the service from this state.
Failed	The service is presumed dead. A service is placed into this state whenever a resource's *stop* operation fails. After a service is placed into this state, you must verify that there are no resources allocated (mounted file systems, for example) prior to issuing a **`disable`** request. The only operation that can take place when a service has entered this state is **`disable`**.
Uninitialized	This state can appear in certain cases during startup and running **`clustat -f`**.

Example 8.3. **`clustat`** Display

```
[root@example-01 ~]#clustat
Cluster Status for mycluster @ Wed Nov 17 05:40:15 2010
Member Status: Quorate

 Member Name                             ID   Status
 ------ ----                             ---- ------
 node-03.example.com                         3 Online, rgmanager
 node-02.example.com                         2 Online, rgmanager
 node-01.example.com                         1 Online, Local, rgmanager

 Service Name                   Owner (Last)                   State
 ------- ----                   ----- ------                   -----
 service:example_apache         node-01.example.com            started
 service:example_apache2        (none)                         disabled
```

8.3.2. Managing HA Services with `clusvcadm`

You can manage HA services using the **`clusvcadm`** command. With it you can perform the following operations:

- Enable and start a service.
- Disable a service.
- Stop a service.
- Freeze a service
- Unfreeze a service
- Migrate a service (for virtual machine services only)
- Relocate a service.
- Restart a service.

Table 8.2, "Service Operations" describes the operations in more detail. For a complete description on how to perform those operations, refer to the **`clusvcadm`** utility man page.

Table 8.2. Service Operations

Service Operation	Description	Command Syntax
Enable	Start the service, optionally on a preferred target and optionally according to failover domain rules. In the absence of either a preferred target or failover domain rules, the local host where **`clusvcadm`** is run will start the service. If the original *start* fails, the service behaves as though a *relocate* operation was requested (refer to **Relocate** in this table). If the operation succeeds, the service is placed in the started state.	**`clusvcadm -e <service_name>`** or **`clusvcadm -e <service_name> -m <member>`** (Using the -m option specifies the preferred target member on which to start the service.)
Disable	Stop the service and place into the disabled state. This is the only	**`clusvcadm -d <service_name>`**

Service Operation	Description	Command Syntax
	permissible operation when a service is in the *failed* state.	
Relocate	Move the service to another node. Optionally, you may specify a preferred node to receive the service, but the inability of the service to run on that host (for example, if the service fails to start or the host is offline) does not prevent relocation, and another node is chosen. **rgmanager** attempts to start the service on every permissible node in the cluster. If no permissible target node in the cluster successfully starts the service, the relocation fails and the service is attempted to be restarted on the original owner. If the original owner cannot restart the service, the service is placed in the *stopped* state.	**clusvcadm -r <service_name>** or **clusvcadm -r <service_name> -m <member>** (Using the -m option specifies the preferred target member on which to start the service.)
Stop	Stop the service and place into the *stopped* state.	**clusvcadm -s <service_name>**
Freeze	Freeze a service on the node where it is currently running. This prevents status checks of the service as well as failover in the event the node fails or rgmanager is stopped. This can be used to suspend a service to allow maintenance of underlying resources. Refer to *the section called "Considerations for Using the Freeze and Unfreeze Operations"* for important information about using the *freeze* and *unfreeze* operations.	**clusvcadm -Z <service_name>**
Unfreeze	Unfreeze takes a service out of the *freeze* state. This re-enables status checks. Refer to *the section called "Considerations for Using the Freeze and Unfreeze Operations"* for important information about using the *freeze* and *unfreeze* operations.	**clusvcadm -U <service_name>**
Migrate	Migrate a virtual machine to another node. You must specify a target node. Depending on the failure, a failure to migrate may result with the virtual machine in the *failed* state or in the started state on the original owner.	**clusvcadm -M <service_name> -m <member>**

Service Operation	Description	Command Syntax
		Important For the *migrate* operation, you *must specify* a target node using the **-m <member>** option.
Restart	Restart a service on the node where it is currently running.	**clusvcadm -R <service_name>**

Considerations for Using the Freeze and Unfreeze Operations

Using the *freeze* operation allows maintenance of parts of **rgmanager** services. For example, if you have a database and a web server in one **rgmanager** service, you may freeze the **rgmanager** service, stop the database, perform maintenance, restart the database, and unfreeze the service.

When a service is frozen, it behaves as follows:

- *Status* checks are disabled.
- *Start* operations are disabled.
- *Stop* operations are disabled.
- Failover will not occur (even if you power off the service owner).

Important

Failure to follow these guidelines may result in resources being allocated on multiple hosts:

- You *must not* stop all instances of rgmanager when a service is frozen unless you plan to reboot the hosts prior to restarting rgmanager.
- You *must not* unfreeze a service until the reported owner of the service rejoins the cluster and restarts rgmanager.

8.4. Updating a Configuration

Updating the cluster configuration consists of editing the cluster configuration file (**/etc/cluster/cluster.conf**) and propagating it to each node in the cluster. You can update the configuration using either of the following procedures:

- *Section 8.4.1, "Updating a Configuration Using **cman_tool version -r**"*
- *Section 8.4.2, "Updating a Configuration Using **scp**"*

8.4.1. Updating a Configuration Using cman_tool version -r

To update the configuration using the **cman_tool version -r** command, perform the following steps:

1. At any node in the cluster, edit the **/etc/cluster/cluster.conf** file.
2. Update the **config_version** attribute by incrementing its value (for example, changing from **config_version="2"** to **config_version="3">**).
3. Save **/etc/cluster/cluster.conf**.
4. Run the **cman_tool version -r** command to propagate the configuration to the rest of the cluster nodes. It is necessary that **ricci** be running in each cluster node to be able to propagate updated cluster configuration information.
5. Verify that the updated configuration file has been propagated.
6. You may skip this step (restarting cluster software) if you have made only the following configuration changes:
   - Deleting a node from the cluster configuration—*except* where the node count changes from greater than two nodes to two nodes. For information about deleting a node from a cluster and transitioning from greater than two nodes to two nodes, refer to *Section 8.2, "Deleting or Adding a Node"*.
   - Adding a node to the cluster configuration—*except* where the node count changes from two nodes to greater than two nodes. For information about adding a node to a cluster and transitioning from two nodes to greater than two nodes, refer to *Section 8.2.2, "Adding a Node to a Cluster"*.
   - Changes to how daemons log information.
   - HA service/VM maintenance (adding, editing, or deleting).
   - Resource maintenance (adding, editing, or deleting).
   - Failover domain maintenance (adding, editing, or deleting).

   Otherwise, you must restart the cluster software as follows:

   a. At each node, stop the cluster software according to *Section 8.1.2, "Stopping Cluster Software"*. For example:

```
[root@example-01 ~]# service rgmanager stop
Stopping Cluster Service Manager:                          [  OK  ]
[root@example-01 ~]# service gfs2 stop
Unmounting GFS2 filesystem (/mnt/gfsA):                    [  OK  ]
Unmounting GFS2 filesystem (/mnt/gfsB):                    [  OK  ]
[root@example-01 ~]# service clvmd stop
Signaling clvmd to exit                                    [  OK  ]
clvmd terminated                                           [  OK  ]
[root@example-01 ~]# service cman stop
Stopping cluster:
   Leaving fence domain...                                 [  OK  ]
   Stopping gfs_controld...                                [  OK  ]
   Stopping dlm_controld...                                [  OK  ]
   Stopping fenced...                                      [  OK  ]
   Stopping cman...                                        [  OK  ]
```

```
   Waiting for corosync to shutdown:                          [  OK  ]
   Unloading kernel modules...                                [  OK  ]
   Unmounting configfs...                                     [  OK  ]
[root@example-01 ~]#
```

b. At each node, start the cluster software according to *Section 8.1.1, "Starting Cluster Software"*. For example:

```
[root@example-01 ~]# service cman start
Starting cluster:
   Checking Network Manager...                                [  OK  ]
   Global setup...                                            [  OK  ]
   Loading kernel modules...                                  [  OK  ]
   Mounting configfs...                                       [  OK  ]
   Starting cman...                                           [  OK  ]
   Waiting for quorum...                                      [  OK  ]
   Starting fenced...                                         [  OK  ]
   Starting dlm_controld...                                   [  OK  ]
   Starting gfs_controld...                                   [  OK  ]
   Unfencing self...                                          [  OK  ]
   Joining fence domain...                                    [  OK  ]
[root@example-01 ~]# service clvmd start
Starting clvmd:                                               [  OK  ]
Activating VG(s):   2 logical volume(s) in volume group "vg_example" now active
                                                              [  OK  ]
[root@example-01 ~]# service gfs2 start
Mounting GFS2 filesystem (/mnt/gfsA):                         [  OK  ]
Mounting GFS2 filesystem (/mnt/gfsB):                         [  OK  ]
[root@example-01 ~]# service rgmanager start
Starting Cluster Service Manager:                             [  OK  ]
[root@example-01 ~]#
```

Stopping and starting the cluster software ensures that any configuration changes that are checked only at startup time are included in the running configuration.

7. At any cluster node, run **`cman_tool nodes`** to verify that the nodes are functioning as members in the cluster (signified as "M" in the status column, "Sts"). For example:

```
[root@example-01 ~]# cman_tool nodes
Node  Sts   Inc   Joined               Name
   1   M    548   2010-09-28 10:52:21  node-01.example.com
   2   M    548   2010-09-28 10:52:21  node-02.example.com
   3   M    544   2010-09-28 10:52:21  node-03.example.com
```

8. At any node, using the **`clustat`** utility, verify that the HA services are running as expected. In addition, **`clustat`** displays status of the cluster nodes. For example:

```
[root@example-01 ~]#clustat
Cluster Status for mycluster @ Wed Nov 17 05:40:00 2010
Member Status: Quorate

 Member Name                             ID   Status
 ------ ----                             ---- ------
 node-03.example.com                         3 Online, rgmanager
 node-02.example.com                         2 Online, rgmanager
 node-01.example.com                         1 Online, Local, rgmanager

 Service Name                   Owner (Last)                   State
```

```
------- ----                    ----- ------                    -----
service:example_apache          node-01.example.com             started
service:example_apache2         (none)                          disabled
```

9. If the cluster is running as expected, you are done updating the configuration.

8.4.2. Updating a Configuration Using scp

To update the configuration using the **scp** command, perform the following steps:

1. At each node, stop the cluster software according to *Section 8.1.2, "Stopping Cluster Software"*. For example:

```
[root@example-01 ~]# service rgmanager stop
Stopping Cluster Service Manager:                          [  OK  ]
[root@example-01 ~]# service gfs2 stop
Unmounting GFS2 filesystem (/mnt/gfsA):                    [  OK  ]
Unmounting GFS2 filesystem (/mnt/gfsB):                    [  OK  ]
[root@example-01 ~]# service clvmd stop
Signaling clvmd to exit                                    [  OK  ]
clvmd terminated                                           [  OK  ]
[root@example-01 ~]# service cman stop
Stopping cluster:
   Leaving fence domain...                                 [  OK  ]
   Stopping gfs_controld...                                [  OK  ]
   Stopping dlm_controld...                                [  OK  ]
   Stopping fenced...                                      [  OK  ]
   Stopping cman...                                        [  OK  ]
   Waiting for corosync to shutdown:                       [  OK  ]
   Unloading kernel modules...                             [  OK  ]
   Unmounting configfs...                                  [  OK  ]
[root@example-01 ~]#
```

2. At any node in the cluster, edit the **/etc/cluster/cluster.conf** file.

3. Update the **config_version** attribute by incrementing its value (for example, changing from **config_version="2"** to **config_version="3">**).

4. Save **/etc/cluster/cluster.conf**.

5. Validate the updated file against the cluster schema (**cluster.rng**) by running the **ccs_config_validate** command. For example:

```
[root@example-01 ~]# ccs_config_validate
Configuration validates
```

6. If the updated file is valid, use the **scp** command to propagate it to **/etc/cluster/** in each cluster node.

7. Verify that the updated configuration file has been propagated.

8. At each node, start the cluster software according to *Section 8.1.1, "Starting Cluster Software"*. For example:

```
[root@example-01 ~]# service cman start
```

```
Starting cluster:
   Checking Network Manager...                             [  OK  ]
   Global setup...                                         [  OK  ]
   Loading kernel modules...                               [  OK  ]
   Mounting configfs...                                    [  OK  ]
   Starting cman...                                        [  OK  ]
   Waiting for quorum...                                   [  OK  ]
   Starting fenced...                                      [  OK  ]
   Starting dlm_controld...                                [  OK  ]
   Starting gfs_controld...                                [  OK  ]
   Unfencing self...                                       [  OK  ]
   Joining fence domain...                                 [  OK  ]
[root@example-01 ~]# service clvmd start
Starting clvmd:                                            [  OK  ]
Activating VG(s):   2 logical volume(s) in volume group "vg_example" now active
                                                           [  OK  ]
[root@example-01 ~]# service gfs2 start
Mounting GFS2 filesystem (/mnt/gfsA):                      [  OK  ]
Mounting GFS2 filesystem (/mnt/gfsB):                      [  OK  ]
[root@example-01 ~]# service rgmanager start
Starting Cluster Service Manager:                          [  OK  ]
[root@example-01 ~]#
```

9. At any cluster node, run **cman_tool nodes** to verify that the nodes are functioning as members in the cluster (signified as "M" in the status column, "Sts"). For example:

```
[root@example-01 ~]# cman_tool nodes
Node  Sts   Inc   Joined               Name
   1   M    548   2010-09-28 10:52:21  node-01.example.com
   2   M    548   2010-09-28 10:52:21  node-02.example.com
   3   M    544   2010-09-28 10:52:21  node-03.example.com
```

10. At any node, using the **clustat** utility, verify that the HA services are running as expected. In addition, **clustat** displays status of the cluster nodes. For example:

```
[root@example-01 ~]#clustat
Cluster Status for mycluster @ Wed Nov 17 05:40:00 2010
Member Status: Quorate

 Member Name                             ID   Status
 ------ ----                             ---- ------
 node-03.example.com                         3 Online, rgmanager
 node-02.example.com                         2 Online, rgmanager
 node-01.example.com                         1 Online, Local, rgmanager

 Service Name                   Owner (Last)                   State
 ------- ----                   ----- ------                   -----
 service:example_apache         node-01.example.com            started
 service:example_apache2        (none)                         disabled
```

11. If the cluster is running as expected, you are done updating the configuration.

Diagnosing and Correcting Problems in a Cluster

Clusters problems, by nature, can be difficult to troubleshoot. This is due to the increased complexity that a cluster of systems introduces as opposed to diagnosing issues on a single system. However, there are common issues that system administrators are more likely to encounter when deploying or administering a cluster. Understanding how to tackle those common issues can help make deploying and administering a cluster much easier.

This chapter provides information about some common cluster issues and how to troubleshoot them. Additional help can be found in our knowledge base and by contacting an authorized Red Hat support representative. If your issue is related to the GFS2 file system specifically, you can find information about troubleshooting common GFS2 issues in the *Global File System 2* document.

9.1. Cluster Does Not Form

If you find you are having trouble getting a new cluster to form, check for the following things:

- Make sure you have name resolution set up correctly. The cluster node name in the **`cluster.conf`** file should correspond to the name used to resolve that cluster's address over the network that cluster will be using to communicate. For example, if your cluster's node names are **`nodea`** and **`nodeb`** make sure both nodes have entries in the **`/etc/cluster/cluster.conf`** file and **`/etc/hosts`** file that match those names.
- If the cluster uses multicast for communication between nodes, make sure that multicast traffic is not being blocked, delayed, or otherwise interfered with on the network that the cluster is using to communicate. Note that some Cisco switches have features that may cause delays in multicast traffic.
- Use **`telnet`** or **`SSH`** to verify whether you can reach remote nodes.
- Execute the **`ethtool eth1 | grep link`** command to check whether the ethernet link is up.
- Use the **`tcpdump`** command at each node to check the network traffic.
- Ensure that you do not have firewall rules blocking communication between your nodes.
- Ensure that the interfaces you are passing cluster traffic over are not using any bonding mode other than 1 and are not using VLAN tagging.

9.2. Nodes Unable to Rejoin Cluster after Fence or Reboot

If your nodes do not rejoin the cluster after a fence or reboot, check for the following things:

- Clusters that are passing their traffic through a Cisco Catalyst switch may experience this problem.
- Ensure that all cluster nodes have the same version of the **`cluster.conf`** file. If the **`cluster.conf`** file is different on any of the nodes, then nodes may be unable to join the cluster post fence.

  As of Red Hat Enterprise Linux 6.1, you can use the following command to verify that all of the nodes specified in the host's cluster configuration file have the identical cluster configuration file:

```
ccs -h host --checkconf
```

For information on the **ccs** command, see *Chapter 5, Configuring Red Hat High Availability Add-On With the **ccs** Command* and *Chapter 6, Managing Red Hat High Availability Add-On With **ccs***.

- Make sure that you have configured **chkconfig on** for cluster services in the node that is attempting to join the cluster.
- Ensure that no firewall rules are blocking the node from communicating with other nodes in the cluster.

9.3. Cluster Services Hang

When the cluster services attempt to fence a node, the cluster services stop until the fence operation has successfully completed. Therefore, if your cluster-controlled storage or services hang and the cluster nodes show different views of cluster membership or if your cluster hangs when you try to fence a node and you need to reboot nodes to recover, check for the following conditions:

- The cluster may have attempted to fence a node and the fence operation may have failed.
- Look through the **/var/log/messages** file on all nodes and see if there are any failed fence messages. If so, then reboot the nodes in the cluster and configure fencing correctly.
- Verify that a network partition did not occur, as described in *Section 9.6, "Each Node in a Two-Node Cluster Reports Second Node Down"*. and verify that communication between nodes is still possible and that the network is up.
- If nodes leave the cluster the remaining nodes may be inquorate. The cluster needs to be quorate to operate. If nodes are removed such that the cluster is no longer quorate then services and storage will hang. Either adjust the expected votes or return the required amount of nodes to the cluster.

Note

You can fence a node manually with the **fence_node** command or with **Conga**. For information, see the **fence_node** man page and *Section 4.2.2, "Causing a Node to Leave or Join a Cluster"*.

9.4. Cluster Service Will Not Start

If a cluster-controlled service will not start, check for the following conditions.

- There may be a syntax error in the service configuration in the **cluster.conf** file. You can use the **rg_test** command to validate the syntax in your configuration. If there are any configuration or syntax faults, the **rg_test** will inform you what the problem is.

```
$ rg_test test /etc/cluster/cluster.conf start service servicename
```

For more information on the **rg_test** command, see *Section C.5, "Debugging and Testing Services and Resource Ordering"*.

If the configuration is valid, then increase the resource group manager's logging and then read the messages logs to determine what is causing the service start to fail. You can increase the log level by adding the **loglevel="7"** parameter to the **rm** tag in the **cluster.conf** file. You will then

get increased verbosity in your messages logs with regards to starting, stopping, and migrating clustered services.

9.5. Cluster-Controlled Services Fails to Migrate

If a cluster-controlled service fails to migrate to another node but the service will start on some specific node, check for the following conditions.

- Ensure that the resources required to run a given service are present on all nodes in the cluster that may be required to run that service. For example, if your clustered service assumes a script file in a specific location or a file system mounted at a specific mount point then you must ensure that those resources are available in the expected places on all nodes in the cluster.
- Ensure that failover domains, service dependency, and service exclusivity are not configured in such a way that you are unable to migrate services to nodes as you'd expect.
- If the service in question is a virtual machine resource, check the documentation to ensure that all of the correct configuration work has been completed.
- Increase the resource group manager's logging, as described in *Section 9.4, "Cluster Service Will Not Start"*, and then read the messages logs to determine what is causing the service start to fail to migrate.

9.6. Each Node in a Two-Node Cluster Reports Second Node Down

If your cluster is a two-node cluster and each node reports that it is up but that the other node is down, this indicates that your cluster nodes are unable to communicate with each other via multicast over the cluster heartbeat network. This is known as "split brain" or a "network partition." To address this, check the conditions outlined in *Section 9.1, "Cluster Does Not Form"*.

9.7. Nodes are Fenced on LUN Path Failure

If a node or nodes in your cluster get fenced whenever you have a LUN path failure, this may be a result of the use of a quorum disk over multipathed storage. If you are using a quorum disk, and your quorum disk is over multipathed storage, ensure that you have all of the correct timings set up to tolerate a path failure.

9.8. Quorum Disk Does Not Appear as Cluster Member

If you have configured your system to use a quorum disk but the quorum disk does not appear as a member of your cluster, check for the following conditions.

- Ensure that you have set **`chkconfig on`** for the **`qdisk`** service.
- Ensure that you have started the **`qdisk`** service.
- Note that it may take multiple minutes for the quorum disk to register with the cluster. This is normal and expected behavior.

9.9. Unusual Failover Behavior

A common problem with cluster servers is unusual failover behavior. Services will stop when other services start or services will refuse to start on failover. This can be due to having complex systems of

failover consisting of failover domains, service dependency, and service exclusivity. Try scaling back to a simpler service or failover domain configuration and see if the issue persists. Avoid features like service exclusivity and dependency unless you fully understand how those features may effect failover under all conditions.

9.10. Fencing Occurs at Random

If you find that a node is being fenced at random, check for the following conditions.

- The root cause of fences is *always* a node losing token, meaning that it lost communication with the rest of the cluster and stopped returning heartbeat.
- Any situation that results in a system not returning heartbeat within the specified token interval could lead to a fence. By default the token interval is 10 seconds. It can be specified by adding the desired value (in milliseconds) to the token parameter of the totem tag in the **`cluster.conf`** file (for example, setting **`totem token="30000"`** for 30 seconds).
- Ensure that the network is sound and working as expected.
- Ensure that exotic bond modes and VLAN tagging are not in use on interfaces that the cluster uses for inter-node communication.
- Take measures to determine if the system is "freezing" or kernel panicking. Set up the **`kdump`** utility and see if you get a core during one of these fences.
- Make sure some situation is not arising that you are wrongly attributing to a fence, for example the quorum disk ejecting a node due to a storage failure or a third party product like Oracle RAC rebooting a node due to some outside condition. The messages logs are often very helpful in determining such problems. Whenever fences or node reboots occur it should be standard practice to inspect the messages logs of all nodes in the cluster from the time the reboot/fence occurred.
- Thoroughly inspect the system for hardware faults that may lead to the system not responding to heartbeat when expected.

SNMP Configuration with the Red Hat High Availability Add-On

As of the Red Hat Enterprise Linux 6.1 release and later, the Red Hat High Availability Add-On provides support for SNMP traps. This chapter describes how to configure your system for SNMP followed by a summary of the traps that the Red Hat High Availability Add-On emits for specific cluster events.

10.1. SNMP and the Red Hat High Availability Add-On

The Red Hat High Availability Add-On SNMP subagent is **foghorn**, which emits the SNMP traps. The **foghorn** subagent talks to the **snmpd** daemon by means of the AgentX Protocol. The **foghorn** subagent only creates SNMP traps; it does not support other SNMP operations such as **get** or **set**.

There are currently no **config** options for the **foghorn** subagent. It cannot be configured to use a specific socket; only the default AgentX socket is currently supported.

10.2. Configuring SNMP with the Red Hat High Availability Add-On

To configure SNMP with the Red Hat High Availability Add-On, perform the following steps on each node in the cluster to ensure that the necessary services are enabled and running.

1. To use SNMP traps with the Red Hat High Availability Add-On, the **snmpd** service is required and acts as the master agent. Since the **foghorn** service is the subagent and uses the AgentX protocol, you must add the following line to the **/etc/snmp/snmpd.conf** file to enable AgentX support:

   ```
   master agentx
   ```

2. To specify the host where the SNMP trap notifications should be sent, add the following line to the to the **/etc/snmp/snmpd.conf** file:

   ```
   trap2sink host
   ```

   For more information on notification handling, see the **snmpd.conf** man page.

3. Make sure that the **snmpd** daemon is enabled and running by executing the following commands:

   ```
   # chkconfig snmpd on
   # service snmpd start
   ```

4. If the **messagebus** daemon is not already enabled and running, execute the following commands:

   ```
   # chkconfig messagebus on
   # service messagebus start
   ```

5. Make sure that the **foghorn** daemon is enabled and running by executing the following commands:

```
# chkconfig foghorn on
# service foghorn start
```

6. Execute the following command to configure your system so that the **COROSYNC-MIB** generates SNMP traps and to ensure that the **corosync-notifyd** daemon is enabled and running:

```
# echo "OPTIONS=\"-d\" " > /etc/sysconfig/corosync-notifyd
# chkconfig corosync-notifyd on
# service corosync-notifyd start
```

After you have configured each node in the cluster for SNMP and ensured that the necessary services are running, D-bus signals will be received by the **foghorn** service and translated into SNMPv2 traps. These traps are then passed to the host that you defined with the **trapsink** entry to receive SNMPv2 traps.

10.3. Forwarding SNMP traps

It is possible to forward SNMP traps to a machine that is not part of the cluster where you can use the **snmptrapd** daemon on the external machine and customize how to respond to the notifications.

Perform the following steps to forward SNMP traps in a cluster to a machine that is not one of the cluster nodes:

1. For each node in the cluster, follow the procedure described in *Section 10.2, "Configuring SNMP with the Red Hat High Availability Add-On"*, setting the **trap2sink *host*** entry in the **/etc/snmp/snmpd.conf** file to specify the external host that will be running the **snmptrapd** daemon.

2. On the external host that will receive the traps, edit the **/etc/snmp/snmptrapd.conf** configuration file to specify your community strings. For example, you can use the following entry to allow the **snmptrapd** daemon to process notifications using the **public** community string.

```
authCommunity log,execute,net public
```

3. On the external host that will receive the traps, make sure that the **snmptrapd** daemon is enabled and running by executing the following commands:

```
# chkconfig snmptrapd on
# service snmptrapd start
```

For further information on processing SNMP notifications, see the **snmptrapd.conf** man page.

10.4. SNMP Traps Produced by Red Hat High Availability Add-On

The **foghorn** daemon generates the following traps:

- **fenceNotifyFenceNode**

This trap occurs whenever a fenced node attempts to fence another node. Note that this trap is only generated on one node - the node that attempted to perform the fence operation. The notification includes the following fields:

- **fenceNodeName** - name of the fenced node
- **fenceNodeID** - node id of the fenced node
- **fenceResult** - the result of the fence operation (0 for success, -1 for something went wrong, -2 for no fencing methods defined)

- **rgmanagerServiceStateChange**

  This trap occurs when the state of a cluster service changes. The notification includes the following fields:

  - **rgmanagerServiceName** - the name of the service, which includes the service type (for example, **service:foo** or **vm:foo**).
  - **rgmanagerServiceState** - the state of the service. This excludes transitional states such as **starting** and **stopping** to reduce clutter in the traps.
  - **rgmanagerServiceFlags** - the service flags. There are currently two supported flags: **frozen**, indicating a service which has been frozen using **clusvcadm -Z**, and **partial**, indicating a service in which a failed resource has been flagged as **non-critical** so that the resource may fail and its components manually restarted without the entire service being affected.
  - **rgmanagerServiceCurrentOwner** - the service owner. If the service is not running, this will be **(none)**.
  - **rgmanagerServicePreviousOwner** - the last service owner, if known. If the last owner is not known, this may indicate **(none)**.

The **corosync-nodifyd** daemon generates the following traps:

- **corosyncNoticesNodeStatus**

  This trap occurs when a node joins or leaves the cluster. The notification includes the following fields:

  - **corosyncObjectsNodeName** - node name
  - **corosyncObjectsNodeID** - node id
  - **corosyncObjectsNodeAddress** - node IP address
  - **corosyncObjectsNodeStatus** - node status (**joined** or **left**)

- **corosyncNoticesQuorumStatus**

  This trap occurs when the quorum state changes. The notification includes the following fields:

  - **corosyncObjectsNodeName** - node name
  - **corosyncObjectsNodeID** - node id
  - **corosyncObjectsQuorumStatus** - new state of the quorum (**quorate** or **NOT quorate**)

- **corosyncNoticesAppStatus**

  This trap occurs when a client application connects or disconnects from Corosync.

  - **corosyncObjectsNodeName** - node name
  - **corosyncObjectsNodeID** - node id
  - **corosyncObjectsAppName** - application name
  - **corosyncObjectsAppStatus** - new state of the application (**connected** or **disconnected**)

Clustered Samba Configuration

The Red Hat Enterprise Linux 6.2 release provides support for running Clustered Samba in an active/active configuration. This requires that you install and configure CTDB on all nodes in a cluster, which you use in conjunction with GFS2 clustered file systems.

Note

Red Hat Enterprise Linux 6 supports a maximum of four nodes running clustered Samba.

This chapter describes the procedure for configuring CTDB by configuring an example system. For information on configuring GFS2 file systems, refer to *Global File System 2*. For information on configuring logical volumes, refer to *Logical Volume Manager Administration*.

11.1. CTDB Overview

CTDB is a cluster implementation of the TDB database used by Samba. To use CTDB, a clustered file system must be available and shared on all nodes in the cluster. CTDB provides clustered features on top of this clustered file system. As of the Red Hat Enterprise Linux 6.2 release, CTDB also runs a cluster stack in parallel to the one provided by Red Hat Enterprise Linux clustering. CTDB manages node membership, recovery/failover, IP relocation and Samba services.

11.2. Required Packages

In addition to the standard packages required to run the Red Hat High Availability Add-On and the Red Hat Resilient Storage Add-On, running Samba with Red Hat Enterprise Linux clustering requires the following packages:

- **`ctdb`**
- **`samba`**
- **`samba-common`**
- **`samba-winbind-clients`**

11.3. GFS2 Configuration

Configuring Samba with the Red Hat Enterprise Linux clustering requires two GFS2 file systems: One small file system for CTDB, and a second file system for the Samba share. This example shows how to create the two GFS2 file systems.

Before creating the GFS2 file systems, first create an LVM logical volume for each of the file systems. For information on creating LVM logical volumes, refer to *Logical Volume Manager Administration*. This example uses the following logical volumes:

- **`/dev/csmb_vg/csmb_lv`**, which will hold the user data that will be exported via a Samba share and should be sized accordingly. This example creates a logical volume that is 100GB in size.
- **`/dev/csmb_vg/ctdb_lv`**, which will store the shared CTDB state information and needs to be 1GB in size.

You create clustered volume groups and logical volumes on one node of the cluster only.

To create a GFS2 file system on a logical volume, run the **mkfs.gfs2** command. You run this command on one cluster node only.

To create the file system to host the Samba share on the logical volume **/dev/csmb_vg/csmb_lv**, execute the following command:

```
[root@clusmb-01 ~]# mkfs.gfs2 -j3 -p lock_dlm -t csmb:gfs2 /dev/csmb_vg/csmb_lv
```

The meaning of the parameters is as follows:

-j
Specifies the number of journals to create in the filesystem. This example uses a cluster with three nodes, so we create one journal per node.

-p
Specifies the locking protocol. **lock_dlm** is the locking protocol GFS2 uses for inter-node communication.

-t
Specifies the lock table name and is of the format *cluster_name:fs_name*. In this example, the cluster name as specified in the **cluster.conf** file is **csmb**, and we use **gfs2** as the name for the file system.

The output of this command appears as follows:

```
This will destroy any data on /dev/csmb_vg/csmb_lv.
  It appears to contain a gfs2 filesystem.

Are you sure you want to proceed? [y/n] y

Device:
/dev/csmb_vg/csmb_lv
Blocksize:  4096
Device Size  100.00 GB (26214400 blocks)
Filesystem Size: 100.00 GB (26214398 blocks)
Journals:  3
Resource Groups:  400
Locking Protocol:   "lock_dlm"
Lock Table:   "csmb:gfs2"
UUID:
  94297529-ABG3-7285-4B19-182F4F2DF2D7
```

In this example, the **/dev/csmb_vg/csmb_lv** file system will be mounted at **/mnt/gfs2** on all nodes. This mount point must match the value that you specify as the location of the **share** directory with the **path =** option in the **/etc/samba/smb.conf** file, as described in *Section 11.5, "Samba Configuration"*.

To create the file system to host the CTDB state information on the logical volume **/dev/csmb_vg/ctdb_lv**, execute the following command:

```
[root@clusmb-01 ~]# mkfs.gfs2 -j3 -p lock_dlm -t csmb:ctdb_state /dev/csmb_vg/ctdb_lv
```

Note that this command specifies a different lock table name than the lock table in the example that created the filesystem on **/dev/csmb_vg/csmb_lv**. This distinguishes the lock table names for the different devices used for the file systems.

The output of the **mkfs.gfs2** appears as follows:

```
This will destroy any data on /dev/csmb_vg/ctdb_lv.
  It appears to contain a gfs2 filesystem.

Are you sure you want to proceed? [y/n] y

Device:
/dev/csmb_vg/ctdb_lv
Blocksize:    4096
Device Size   1.00 GB (262144 blocks)
Filesystem Size:  1.00 GB (262142 blocks)
Journals:  3
Resource Groups:  4
Locking Protocol:  "lock_dlm"
Lock Table:   "csmb:ctdb_state"
UUID:
  BCDA8025-CAF3-85BB-B062-CC0AB8849A03
```

In this example, the **/dev/csmb_vg/ctdb_lv** file system will be mounted at **/mnt/ctdb** on all nodes. This mount point must match the value that you specify as the location of the **.ctdb.lock** file with the **CTDB_RECOVERY_LOCK** option in the **/etc/sysconfig/ctdb** file, as described in *Section 11.4, "CTDB Configuration"*.

11.4. CTDB Configuration

The CTDB configuration file is located at **/etc/sysconfig/ctdb**. The mandatory fields that must be configured for CTDB operation are as follows:

- **CTDB_NODES**
- **CTDB_PUBLIC_ADDRESSES**
- **CTDB_RECOVERY_LOCK**
- **CTDB_MANAGES_SAMBA** (must be enabled)
- **CTDB_MANAGES_WINBIND** (must be enabled if running on a member server)

The following example shows a configuration file with the mandatory fields for CTDB operation set with example parameters:

```
CTDB_NODES=/etc/ctdb/nodes
CTDB_PUBLIC_ADDRESSES=/etc/ctdb/public_addresses
CTDB_RECOVERY_LOCK="/mnt/ctdb/.ctdb.lock"
CTDB_MANAGES_SAMBA=yes
CTDB_MANAGES_WINBIND=yes
```

The meaning of these parameters is as follows.

CTDB_NODES
: Specifies the location of the file which contains the cluster node list.

The **/etc/ctdb/nodes** file that **CTDB_NODES** references simply lists the IP addresses of the cluster nodes, as in the following example:

```
192.168.1.151
192.168.1.152
192.168.1.153
```

In this example, there is only one interface/IP on each node that is used for both cluster/CTDB communication and serving clients. However, it is highly recommended that each cluster node have two network interfaces so that one set of interfaces can be dedicated to cluster/CTDB communication and another set of interfaces can be dedicated to public client access. Use the appropriate IP addresses of the cluster network here and make sure the hostnames/IP addresses used in the **cluster.conf** file are the same. Similarly, use the appropriate interfaces of the public network for client access in the **public_addresses** file.

It is critical that the **/etc/ctdb/nodes** file is identical on all nodes because the ordering is important and CTDB will fail if it finds different information on different nodes.

CTDB_PUBLIC_ADDRESSES

Specifies the location of the file that lists the IP addresses that can be used to access the Samba shares exported by this cluster. These are the IP addresses that you should configure in DNS for the name of the clustered Samba server and are the addresses that CIFS clients will connect to. Configure the name of the clustered Samba server as one DNS type A record with multiple IP addresses and let round-robin DNS distribute the clients across the nodes of the cluster.

For this example, we have configured a round-robin DNS entry **csmb-server** with all the addresses listed in the **/etc/ctdb/public_addresses** file. DNS will distribute the clients that use this entry across the cluster in a round-robin fashion.

The contents of the **/etc/ctdb/public_addresses** file on each node are as follows:

```
192.168.1.201/0 eth0
192.168.1.202/0 eth0
192.168.1.203/0 eth0
```

This example uses three addresses that are currently unused on the network. In your own configuration, choose addresses that can be accessed by the intended clients.

Alternately, this example shows the contents of the **/etc/ctdb/public_addresses** files in a cluster in which there are three nodes but a total of four public addresses. In this example, IP address 198.162.2.1 can be hosted by either node 0 or node 1 and will be available to clients as long as at least one of these nodes is available. Only if both nodes 0 and 1 fail does this public address become unavailable to clients. All other public addresses can only be served by one single node respectively and will therefore only be available if the respective node is also available.

The **/etc/ctdb/public_addresses** file on node 0 includes the following contents:

```
198.162.1.1/24 eth0
198.162.2.1/24 eth1
```

The **/etc/ctdb/public_addresses** file on node 1 includes the following contents:

```
198.162.2.1/24 eth1
198.162.3.1/24 eth2
```

The **/etc/ctdb/public_addresses** file on node 2 includes the following contents:

```
198.162.3.2/24 eth2
```

CTDB_RECOVERY_LOCK

Specifies a lock file that CTDB uses internally for recovery. This file must reside on shared storage such that all the cluster nodes have access to it. The example in this section uses the GFS2 file system that will be mounted at **/mnt/ctdb** on all nodes. This is different from the GFS2 file system that will host the Samba share that will be exported. This recovery lock file is used to prevent split-brain scenarios. With newer versions of CTDB (1.0.112 and later), specifying this file is optional as long as it is substituted with another split-brain prevention mechanism.

CTDB_MANAGES_SAMBA

When enabling by setting it to **yes**, specifies that CTDB is allowed to start and stop the Samba service as it deems necessary to provide service migration/failover.

When **CTDB_MANAGES_SAMBA** is enabled, you should disable automatic **init** startup of the **smb** and **nmb** daemons by executing the following commands:

```
[root@clusmb-01 ~]# chkconfig snb off
[root@clusmb-01 ~]# chkconfig nmb off
```

CTDB_MANAGES_WINBIND

When enabling by setting it to **yes**, specifies that CTDB is allowed to start and stop the **winbind** daemon as required. This should be enabled when you are using CTDB in a Windows domain or in active directory security mode.

When **CTDB_MANAGES_WINBIND** is enabled, you should disable automatic **init** startup of the **winbind** daemon by executing the following command:

```
[root@clusmb-01 ~]# chkconfig windinbd off
```

11.5. Samba Configuration

The Samba configuration file **smb.conf** is located at **/etc/samba/smb.conf** in this example. It contains the following parameters:

```
[global]
 guest ok = yes
 clustering = yes
 netbios name = csmb-server
[csmb]
 comment = Clustered Samba
  public = yes
 path = /mnt/gfs2/share
 writeable = yes
 ea support = yes
```

This example exports a share with name **csmb** located at **/mnt/gfs2/share**. This is different from the GFS2 shared filesystem at **/mnt/ctdb/.ctdb.lock** that we specified as the **CTDB_RECOVERY_LOCK** parameter in the CTDB configuration file at **/etc/sysconfig/ctdb**.

In this example, we will create the **share** directory in **/mnt/gfs2** when we mount it for the first time. The **clustering = yes** entry instructs Samba to use CTDB. The **netbios name = csmb-server** entry explicitly sets all the nodes to have a common NetBIOS name. The **ea support** parameter is required if you plan to use extended attributes.

The **smb.conf** configuration file must be identical on all of the cluster nodes.

Samba also offers registry-based configuration using the **net conf** command to automatically keep configuration in sync between cluster members without having to manually copy configuration files among the cluster nodes. For information on the **net conf** command, refer to the **net**(8) man page.

11.6. Starting CTDB and Samba Services

After starting up the cluster, you must mount the GFS2 file systems that you created, as described in *Section 11.3, "GFS2 Configuration"*. The permissions on the Samba **share** directory and user accounts on the cluster nodes should be set up for client access.

Execute the following command on all of the nodes to start up the **ctdbd** daemon. Since this example configured CTDB with **CTDB_MANAGES_SAMBA=yes**, CTDB will also start up the Samba service on all nodes and export all configured Samba shares.

```
[root@clusmb-01 ~]# service ctdb start
```

It can take a couple of minutes for CTDB to start Samba, export the shares, and stabilize. Executing **ctdb status** shows the status of CTDB, as in the following example:

```
[root@clusmb-01 ~]# ctdb status
Number of nodes:3
pnn:0 192.168.1.151     OK (THIS NODE)
pnn:1 192.168.1.152     OK
pnn:2 192.168.1.153     OK
Generation:1410259202
Size:3
hash:0 lmaster:0
hash:1 lmaster:1
hash:2 lmaster:2
Recovery mode:NORMAL (0)
Recovery master:0
```

When you see that all nodes are "OK", it is safe to move on to use the clustered Samba server, as described in *Section 11.7, "Using the Clustered Samba Server"*.

11.7. Using the Clustered Samba Server

Clients can connect to the Samba share that was exported by connecting to one of the IP addresses specified in the **/etc/ctdb/public_addresses** file, or using the **csmb-server** DNS entry we configured earlier, as shown below:

```
[root@clusmb-01 ~]# mount -t cifs //csmb-server/csmb /mnt/sambashare -o user=testmonkey
```

or

```
[user@clusmb-01 ~]$ smbclient //csmb-server/csmb
```

Appendix A. Fence Device Parameters

This appendix provides tables with parameter descriptions of fence devices as well as the name of the fence agent for each of those devices.

Note

The **Name** parameter for a fence device specifies an arbitrary name for the device that will be used by Red Hat High Availability Add-On. This is not the same as the DNS name for the device.

Note

Certain fence devices have an optional **Password Script** parameter. The **Password Script** parameter allows you to specify that a fence-device password is supplied from a script rather than from the **Password** parameter. Using the **Password Script** parameter supersedes the **Password** parameter, allowing passwords to not be visible in the cluster configuration file (`/etc/cluster/cluster.conf`).

Table A.1. APC Power Switch (telnet/SSH)

Field	Description
Name	A name for the APC device connected to the cluster into which the fence daemon logs via telnet/ssh.
IP Address	The IP address or hostname assigned to the device.
Login	The login name used to access the device.
Password	The password used to authenticate the connection to the device.
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.
Port	The port.
Switch (optional)	The switch number for the APC switch that connects to the node when you have multiple daisy-chained switches.
Use SSH	Indicates that system will use SSH to access the device.
Path to the SSH identity file	The identity file for SSH.
Power wait	Number of seconds to wait after issuing a power off or power on command.
`fence_apc`	The fence agent for APC over telnet/SSH.

Table A.2. APC Power Switch over SNMP

Field	Description
Name	A name for the APC device connected to the cluster into which the fence daemon logs via the SNMP protocol.
IP Address	The IP address or hostname assigned to the device.

Field	Description
UDP/TCP port	The UDP/TCP port to use for connection with the device; the default value is 161.
Login	The login name used to access the device.
Password	The password used to authenticate the connection to the device.
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.
Port	The port.
Switch (optional)	The switch number for the APC switch that connects to the node when you have multiple daisy-chained switches.
SNMP version	The SNMP version to use (1, 2c, 3); the default value is 1.
SNMP community	The SNMP community string; the default value is **`private`**.
SNMP security level	The SNMP security level (noAuthNoPriv, authNoPriv, authPriv).
SNMP authentication protocol	The SNMP authentication protocol (MD5, SHA).
SNMP privacy protocol	The SNMP privacy protocol (DES, AES).
SNMP privacy protocol password	The SNMP privacy protocol password.
SNMP privacy protocol script	The script that supplies a password for SNMP privacy protocol. Using this supersedes the **SNMP privacy protocol password** parameter.
Power wait	Number of seconds to wait after issuing a power off or power on command.
`fence_apc_snmp`	The fence agent for APC that logs into the SNP device via the SNMP protocol.

Table A.3. Brocade Fabric Switch

Field	Description
Name	A name for the Brocade device connected to the cluster.
IP Address	The IP address assigned to the device.
Login	The login name used to access the device.
Password	The password used to authenticate the connection to the device.
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.
Port	The switch outlet number.
`fence_brocade`	The fence agent for Brocade FC switches.

Table A.4. Cisco MDS

Field	Description
Name	A name for the Cisco MDS 9000 series device with SNMP enabled.
IP Address	The IP address or hostname assigned to the device.
Login	The login name used to access the device.
Password	The password used to authenticate the connection to the device.

Field	Description
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.
Port	The port.
SNMP version	The SNMP version to use (1, 2c, 3).
SNMP community	The SNMP community string.
SNMP authentication protocol	The SNMP authentication protocol (MD5, SHA).
SNMP security level	The SNMP security level (noAuthNoPriv, authNoPriv, authPriv).
SNMP privacy protocol	The SNMP privacy protocol (DES, AES).
SNMP privacy protocol password	The SNMP privacy protocol password.
SNMP privacy protocol script	The script that supplies a password for SNMP privacy protocol. Using this supersedes the **SNMP privacy protocol password** parameter.
Power wait	Number of seconds to wait after issuing a power off or power on command.
`fence_cisco_mds`	The fence agent for Cisco MDS.

Table A.5. Cisco UCS

Field	Description
Name	A name for the Cisco UCS device.
IP Address	The IP address or hostname assigned to the device.
Login	The login name used to access the device.
Password	The password used to authenticate the connection to the device.
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.
SSL	The SSL connection.
IP port (optional)	The TCP port to use to connect to the device.
Port	Name of virtual machine.
Power wait	Number of seconds to wait after issuing a power off or power on command.
Power timeout	Number of seconds to test for a status change after issuing a power off or power on command.
Shell timeout	Number of seconds to wait for a command prompt after issuing a command.
Retry on	Number of attempts to retry power on.
`fence_cisco_ucs`	The fence agent for Cisco UCS.

Table A.6. Dell DRAC 5

Field	Description
Name	The name assigned to the DRAC.
IP Address	The IP address or hostname assigned to the DRAC.
IP port (optional)	The TCP port to use to connect to the device.
Login	The login name used to access the DRAC.
Password	The password used to authenticate the connection to the DRAC.

Field	Description
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.
Module name	(optional) The module name for the DRAC when you have multiple DRAC modules.
Force command prompt	The command prompt to use. The default value is '\$'.
Use SSH	Indicates that system will use SSH to access the device.
Path to the SSH identity file	The identity file for SSH.
Power wait	Number of seconds to wait after issuing a power off or power on command.
fence_drac5	The fence agent for Dell DRAC 5.

Table A.7. Egenera SAN Controller

Field	Description
Name	A name for the Egenera BladeFrame device connected to the cluster.
CServer	The hostname (and optionally the username in the form of **username@hostname**) assigned to the device. Refer to the fence_egenera(8) man page for more information.
ESH Path (optional)	The path to the esh command on the cserver (default is /opt/pan- mgr/bin/ esh)
lpan	The logical process area network (LPAN) of the device.
pserver	The processing blade (pserver) name of the device.
fence_egenera	The fence agent for the Egenera BladeFrame.

Table A.8. ePowerSwitch

Field	Description
Name	A name for the ePowerSwitch device connected to the cluster.
IP Address	The IP address or hostname assigned to the device.
Login	The login name used to access the device.
Password	The password used to authenticate the connection to the device.
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.
Port	Physical plug number or name of virtual machine.
Hidden page	The name of the hidden page for the device.
fence_eps	The fence agent for ePowerSwitch.

Table A.9. Fujitsu Siemens Remoteview Service Board (RSB)

Field	Description
Name	A name for the RSB to use as a fence device.
Hostname	The hostname assigned to the device.
Login	The login name used to access the device.
Password	The password used to authenticate the connection to the device.
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.

Field	Description
TCP port	The port number on which the telnet service listens.
fence_rsb	The fence agent for Fujitsu-Siemens RSB.

Table A.10. Fence virt

Field	Description
Name	A name for the Fence virt fence device.
Port	Virtual machine (domain UUID or name) to fence.
Serial device	On the host, the serial device must be mapped in each domain's configuration file. For more information, see the **fence_virt.conf** man page. If this field is specified, it causes the **fence_virt** fencing agent to operate in serial mode. Not specifying a value causes the **fence_virt** fencing agent to operate in VM channel mode.
Serial parameters	The serial parameters. The default is 115200, 8N1.
VM channel IP address	The channel IP. The default value is 10.0.2.179.
Channel port	The channel port. The default value is 1229
fence_virt	The fence agent for a Fence virt fence device.

Table A.11. HP iLO/iLO2 (Integrated Lights Out)

Field	Description
Name	A name for the server with HP iLO support.
IP Address	The IP address or hostname assigned to the device.
IP port (optional)	TCP port to use for connection with the device.
Login	The login name used to access the device.
Password	The password used to authenticate the connection to the device.
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.
Power wait	Number of seconds to wait after issuing a power off or power on command.
fence_ilo	The fence agent for HP iLO devices.

Table A.12. HP iLO (Integrated Lights Out) MP

Field	Description
Name	A name for the server with HP iLO support.
Hostname	The hostname assigned to the device.
IP port (optional)	TCP port to use for connection with the device.
Login	The login name used to access the device.
Password	The password used to authenticate the connection to the device.
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.
SSH	Indicates that the system will use SSH to access the device.
Path to the SSH identity file	The identity file for SSH.

Field	Description
Force command prompt	The command prompt to use. The default value is 'MP>', 'hpiLO->'.
Power wait	Number of seconds to wait after issuing a power off or power on command.
`fence_ilo_mp`	The fence agent for HP iLO MP devices.

Table A.13. IBM BladeCenter

Field	Description
Name	A name for the IBM BladeCenter device connected to the cluster.
IP Address	The IP address or hostname assigned to the device.
IP port (optional)	TCP port to use for connection with the device.
Login	The login name used to access the device.
Password	The password used to authenticate the connection to the device.
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.
Power wait	Number of seconds to wait after issuing a power off or power on command.
Use SSH	Indicates that system will use SSH to access the device.
Path to the SSH identity file	The identity file for SSH.
`fence_bladecenter`	The fence agent for IBM BladeCenter.

Table A.14. IBM BladeCenter SNMP

Field	Description
Name	A name for the IBM BladeCenter SNMP device connected to the cluster.
IP Address	The IP address or hostname assigned to the device.
UDP/TCP port (optional)	UDP/TCP port to use for connections with the device; the default value is 161.
Login	The login name used to access the device.
Password	The password used to authenticate the connection to the device.
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.
Port	Physical plug number or name of virtual machine.
SNMP version	The SNMP version to use (1, 2c, 3); the default value is 1.
SNMP community	The SNMP community string.
SNMP security level	The SNMP security level (noAuthNoPriv, authNoPriv, authPriv).
SNMP authentication protocol	The SNMP authentication protocol (MD5, SHA).
SNMP privacy protocol	The SNMP privacy protocol (DES, AES).
SNMP privacy protocol password	The SNMP privacy protocol password.
SNMP privacy protocol script	The script that supplies a password for SNMP privacy protocol. Using this supersedes the **SNMP privacy protocol password** parameter.

Field	Description
Power wait	Number of seconds to wait after issuing a power off or power on command.
`fence_bladecenter`	The fence agent for IBM BladeCenter.

Table A.15. IF MIB

Field	Description
Name	A name for the IF MIB device connected to the cluster.
IP Address	The IP address or hostname assigned to the device.
UDP/TCP port(optional)	The UDP/TCP port to use for connection with the device; the default value is 161.
Login	The login name used to access the device.
Password	The password used to authenticate the connection to the device.
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.
SNMP version	The SNMP version to use (1, 2c, 3); the default value is 1.
SNMP community	The SNMP community string.
SNMP security level	The SNMP security level (noAuthNoPriv, authNoPriv, authPriv).
SNMP authentication protocol	The SNMP authentication protocol (MD5, SHA).
SNMP privacy protocol	The SNMP privacy protocol (DES, AES).
SNMP privacy protocol password	The SNMP privacy protocol password.
SNMP privacy protocol script	The script that supplies a password for SNMP privacy protocol. Using this supersedes the **SNMP privacy protocol password** parameter.
Power wait	Number of seconds to wait after issuing a power off or power on command.
Port	Physical plug number or name of virtual machine.
`fence_ifmib`	The fence agent for IF-MIB devices.

Table A.16. Intel Modular

Field	Description
Name	A name for the Intel Modular device connected to the cluster.
IP Address	The IP address or hostname assigned to the device.
Login	The login name used to access the device.
Password	The password used to authenticate the connection to the device.
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.
Port	Physical plug number or name of virtual machine.
SNMP version	The SNMP version to use (1, 2c, 3); the default value is 1.
SNMP community	The SNMP community string; the default value is **`private`**.
SNMP security level	The SNMP security level (noAuthNoPriv, authNoPriv, authPriv).
SNMP authentication protocol	The SNMP authentication protocol (MD5, SHA).

Field	Description
SNMP privacy protocol	The SNMP privacy protocol (DES, AES).
SNMP privacy protocol password	The SNMP privacy protocol password.
SNMP privacy protocol script	The script that supplies a password for SNMP privacy protocol. Using this supersedes the **SNMP privacy protocol password** parameter.
Power wait	Number of seconds to wait after issuing a power off or power on command.
`fence_intelmodular`	The fence agent for APC.

Table A.17. IPMI (Intelligent Platform Management Interface) LAN

Field	Description
Name	A name for the IPMI LAN device connected to the cluster.
IP Address	The IP address or hostname assigned to the device.
Login	The login name of a user capable of issuing power on/off commands to the given IPMI port.
Password	The password used to authenticate the connection to the IPMI port.
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.
Authentication Type	**`none`**, **`password`**, **`md2`**, or **`md5`**.
Use Lanplus	**`True`** or **`1`**. If blank, then value is **`False`**.
Ciphersuite to use	The remote server authentication, integrity, and encryption algorithms to use for IPMIv2 lanplus connections.
`fence_ipmilan`	The fence agent for machines controlled by IPMI.

Table A.18. RHEV-M REST API (RHEL 6.2 and later against RHEV 3.0 and later)

Field	Description
Name	Name of the RHEV-M REST API fencing device.
Hostname	The IP address or hostname assigned to the device.
IP Port	The TCP port to use for connection with the device.
Login	The login name used to access the device.
Password	The password used to authenticate the connection to the device.
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.
Separator	Separator for CSV created by operation list The default value is a comma(,).
Use SSL connections	Use SSL connections to communicate with the device.
Power wait	Number of seconds to wait after issuing a power off or power on command.
Port	Physical plug number or name of virtual machine.
Power timeout	Number of seconds to test for a status change after issuing a power off or power on command.
Shell timeout	Number of seconds to wait for a command prompt after issuing a command.
`fence_rhevm`	The fence agent for RHEV-M REST API.

Table A.19. SCSI Fencing

Field	Description
Name	A name for the SCSI fence device.
Node name	Name of the node to be fenced. Refer to the **fence_scsi**(8) man page for more information.
fence_scsi	The fence agent for SCSI persistent reservations.

Note

Use of SCSI persistent reservations as a fence method is supported with the following limitations:

- When using SCSI fencing, all nodes in the cluster must register with the same devices so that each node can remove another node's registration key from all the devices it is registered with.
- Devices used for the cluster volumes should be a complete LUN, not partitions. SCSI persistent reservations work on an entire LUN, meaning that access is controlled to each LUN, not individual partitions.

Table A.20. VMware (SOAP Interface) (Red Hat Enterprise Linux 6.2 and later)

Field	Description
Name	Name of the virtual machine fencing device.
Hostname	The IP address or hostname assigned to the device.
IP Port	The TCP port to use for connection with the device.
Login	The login name used to access the device.
Password	The password used to authenticate the connection to the device.
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.
Separator	Separator for CSV created by operation list The default value is a comma(,).
Use SSL connections	Use SSL connections to communicate with the device.
Power wait	Number of seconds to wait after issuing a power off or power on command.
Virtual machine (VM) name	Name of virtual machine in inventory path format (e.g., /datacenter/vm/Discovered_virtual_machine/myMachine).
Virtual machine (VM) UUID	The UUID of the virtual machine to fence.
fence_vmware_soap	The fence agent for VMWare over SOAP API.

Table A.21. WTI Power Switch

Field	Description
Name	A name for the WTI power switch connected to the cluster.
IP Address	The IP or hostname address assigned to the device.

Field	Description
IP port (optional)	The TCP port to use to connect to the device.
Login	The login name used to access the device.
Password	The password used to authenticate the connection to the device.
Password Script (optional)	The script that supplies a password for access to the fence device. Using this supersedes the **Password** parameter.
Port	Physical plug number or name of virtual machine.
Force command prompt	The command prompt to use. The default value is ['RSM>', '>MPC', 'IPS>', 'TPS>', 'NBB>', 'NPS>', 'VMR>']
Power wait	Number of seconds to wait after issuing a power off or power on command.
Use SSH	Indicates that system will use SSH to access the device.
Path to the SSH identity file	The identity file for SSH.
`fence_wti`	The fence agent for the WTI network power switch.

Appendix B. HA Resource Parameters

This appendix provides descriptions of HA resource parameters. You can configure the parameters with **luci**, by using the **ccs** command, or by editing **etc/cluster/cluster.conf**. *Table B.1, "HA Resource Summary"* lists the resources, their corresponding resource agents, and references to other tables containing parameter descriptions. To understand resource agents in more detail you can view them in **/usr/share/cluster** of any cluster node.

For a comprehensive list and description of **cluster.conf** elements and attributes, refer to the cluster schema at **/usr/share/cluster/cluster.rng**, and the annotated schema at **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** (for example **/usr/share/doc/cman-3.0.12/cluster_conf.html**).

Table B.1. HA Resource Summary

Resource	Resource Agent	Reference to Parameter Description
Apache	apache.sh	*Table B.2, "Apache Server"*
File System	fs.sh	*Table B.3, "File System"*
GFS2 File System	clusterfs.sh	*Table B.4, "GFS2"*
IP Address	ip.sh	*Table B.5, "IP Address"*
LVM	lvm.sh	*Table B.6, "LVM"*
MySQL	mysql.sh	*Table B.7, "MySQL"*
NFS Client	nfsclient.sh	*Table B.8, "NFS Client"*
NFS Export	nfsexport.sh	*Table B.9, "NFS Export"*
NFS/CIFS Mount	netfs.sh	*Table B.10, "NFS/CIFS Mount"*
Open LDAP	openldap.sh	*Table B.11, "Open LDAP"*
Oracle 10g	oracledb.sh	*Table B.12, "Oracle 10g"*
PostgreSQL 8	postgres-8.sh	*Table B.13, "PostgreSQL 8"*
SAP Database	SAPDatabase	*Table B.14, "SAP Database"*
SAP Instance	SAPInstance	*Table B.15, "SAP Instance"*
Samba	samba.sh	*Table B.16, "Samba Service"*
Script	script.sh	*Table B.17, "Script"*
Service	service.sh	*Table B.18, "Service"*
Sybase ASE	ASEHAagent.sh	*Table B.19, "Sybase ASE Failover Instance"*
Tomcat 6	tomcat-6.sh	*Table B.20, "Tomcat 6"*
Virtual Machine	vm.sh	*Table B.21, "Virtual Machine"* NOTE: **luci** displays this as a virtual service if the host cluster can support virtual machines.

Table B.2. Apache Server

Field	Description
Name	The name of the Apache Service.

Field	Description
Server Root	The default value is **/etc/httpd**.
Config File	Specifies the Apache configuration file. The default valuer is **/etc/httpd/conf**.
httpd Options	Other command line options for **httpd**.
Shutdown Wait (seconds)	Specifies the number of seconds to wait for correct end of service shutdown.

Table B.3. File System

Field	Description
Name	Specifies a name for the file system resource.
File System Type	If not specified, **mount** tries to determine the file system type.
Mount Point	Path in file system hierarchy to mount this file system.
Device	Specifies the device associated with the file system resource. This can be a block device, file system label, or UUID of a file system.
Options	Mount options; that is, options used when the file system is mounted. These may be file-system specific. Refer to the ***mount**(8)* man page for supported mount options.
File System ID	**Note** *File System ID* is used only by NFS services. When creating a new file system resource, you can leave this field blank. Leaving the field blank causes a file system ID to be assigned automatically after you commit the parameter during configuration. If you need to assign a file system ID explicitly, specify it in this field.
Force unmount	If enabled, forces the file system to unmount. The default setting is *disabled*. *Force Unmount* kills all processes using the mount point to free up the mount when it tries to unmount.
Reboot host node if unmount fails	If enabled, reboots the node if unmounting this file system fails. The default setting is *disabled*.
Check file system before mounting	If enabled, causes **fsck** to be run on the file system before mounting it. The default setting is *disabled*.

Table B.4. GFS2

Field	Description
Name	The name of the file system resource.
Mount Point	The path to which the file system resource is mounted.
Device	The device file associated with the file system resource.
Options	Mount options.

Field	Description
File System ID	**Note** *File System ID* is used only by NFS services. When creating a new GFS2 resource, you can leave this field blank. Leaving the field blank causes a file system ID to be assigned automatically after you commit the parameter during configuration. If you need to assign a file system ID explicitly, specify it in this field.
Force Unmount	If enabled, forces the file system to unmount. The default setting is *disabled*. *Force Unmount* kills all processes using the mount point to free up the mount when it tries to unmount. With GFS2 resources, the mount point is *not* unmounted at service tear-down unless *Force Unmount* is *enabled*.
Reboot Host Node if Unmount Fails (self fence)	If enabled and unmounting the file system fails, the node will immediately reboot. Generally, this is used in conjunction with force-unmount support, but it is not required.

Table B.5. IP Address

Field	Description
IP Address	The IP address for the resource. This is a virtual IP address. IPv4 and IPv6 addresses are supported, as is NIC link monitoring for each IP address.
Monitor Link	Enabling this causes the status check to fail if the link on the NIC to which this IP address is bound is not present.

Table B.6. LVM

Field	Description
Name	A unique name for this LVM resource.
Volume Group Name	A descriptive name of the volume group being managed.
Logical Volume Name (optional)	Name of the logical volume being managed. This parameter is optional if there is more than one logical volume in the volume group being managed.

Table B.7. MySQL

Field	Description
Name	Specifies a name of the MySQL server resource.
Config File	Specifies the configuration file. The default value is **/etc/my.cnf**.
Listen Address	Specifies an IP address for MySQL server. If an IP address is not provided, the first IP address from the service is taken.
mysqld Options	Other command line options for **httpd**.
Shutdown Wait (seconds)	Specifies the number of seconds to wait for correct end of service shutdown.

Table B.8. NFS Client

Field	Description
Name	This is a symbolic name of a client used to reference it in the resource tree. This is *not* the same thing as the *`Target`* option.
Target	This is the server from which you are mounting. It can be specified using a hostname, a wildcard (IP address or hostname based), or a netgroup defining a host or hosts to export to.
Option	Defines a list of options for this client — for example, additional client access rights. For more information, refer to the ***`exports`*** *(5)* man page, *General Options*.

Table B.9. NFS Export

Field	Description
Name	Descriptive name of the resource. The NFS Export resource ensures that NFS daemons are running. It is fully reusable; typically, only one NFS Export resource is needed.  **Tip** Name the NFS Export resource so it is clearly distinguished from other NFS resources.

Table B.10. NFS/CIFS Mount

Field	Description
Name	Symbolic name for the NFS or CIFS mount.  **Note** This resource is required when a cluster service is configured to be an NFS client.
Mount Point	Path to which the file system resource is mounted.
Host	NFS/CIFS server IP address or hostname.
NFS Export Path or CIFS share	NFS Export directory name or CIFS share name.
File System type	File system type: • *`NFS`* — Specifies using the default NFS version. This is the default setting. • *`NFS v4`* — Specifies using NFSv4 protocol. • *`CIFS`* — Specifies using CIFS protocol.

Field	Description
Options	Mount options. Specifies a list of mount options. If none are specified, the file system is mounted **-o sync**.
Force Unmount	If *Force Unmount* is enabled, the cluster kills all processes using this file system when the service is stopped. Killing all processes using the file system frees up the file system. Otherwise, the unmount will fail, and the service will be restarted.
No Unmount	If enabled, specifies that the file system should not be unmounted during a stop or relocation operation.

Table B.11. Open LDAP

Field	Description
Name	Specifies a service name for logging and other purposes.
Config File	Specifies an absolute path to a configuration file. The default value is **/etc/openldap/slapd.conf**.
URL List	The default value is **ldap:///**.
slapd Options	Other command line options for **slapd**.
Shutdown Wait (seconds)	Specifies the number of seconds to wait for correct end of service shutdown.

Table B.12. Oracle 10g

Field	Description
Instance name (SID) of Oracle instance	Instance name.
Oracle user name	This is the user name of the Oracle user that the Oracle AS instance runs as.
Oracle application home directory	This is the Oracle (application, not user) home directory. It is configured when you install Oracle.
Virtual hostname (optional)	Virtual Hostname matching the installation hostname of Oracle 10g. Note that during the start/stop of an oracledb resource, your hostname is changed temporarily to this hostname. Therefore, you should configure an oracledb resource as part of an exclusive service only.

Table B.13. PostgreSQL 8

Field	Description
Name	Specifies a service name for logging and other purposes.
Config File	Define absolute path to configuration file. The default value is **/var/lib/pgsql/data/postgresql.conf**.
Postmaster User	User who runs the database server because it cannot be run by root. The default value is postgres.
Postmaster Options	Other command line options for postmaster.
Shutdown Wait (seconds)	Specifies the number of seconds to wait for correct end of service shutdown.

Table B.14. SAP Database

Field	Description
SAP Database Name	Specifies a unique SAP system identifier. For example, P01.
SAP executable directory	Specifies the fully qualified path to **sapstartsrv** and **sapcontrol**.
Database type	Specifies one of the following database types: Oracle, DB6, or ADA.
Oracle TNS listener name	Specifies Oracle TNS listener name.
ABAP stack is not installed, only Java stack is installed	If you do not have an ABAP stack installed in the SAP database, enable this parameter.
J2EE instance bootstrap directory	The fully qualified path the J2EE instance bootstrap directory. For example, **/usr/sap/P01/J00/j2ee/cluster/bootstrap**.
J2EE security store path	The fully qualified path the J2EE security store directory. For example, **/usr/sap/P01/SYS/global/security/lib/tools**.

Table B.15. SAP Instance

Field	Description
SAP Instance Name	The fully qualified SAP instance name. For example, P01_DVEBMGS00_sapp01ci.
SAP executable directory	The fully qualified path to **sapstartsrv** and **sapcontrol**.
Directory containing the SAP START profile	The fully qualified path to the SAP START profile.
Name of the SAP START profile	Specifies name of the SAP START profile.

Note

Regarding *Table B.16, "Samba Service"*, when creating or editing a cluster service, connect a Samba-service resource directly to the service, *not* to a resource within a service.

Table B.16. Samba Service

Field	Description
Name	Specifies the name of the Samba server.

Table B.17. Script

Field	Description
Name	Specifies a name for the custom user script. The script resource allows a standard LSB-compliant init script to be used to start a clustered service.
File (with path)	Enter the path where this custom script is located (for example, **/etc/init.d/*userscript***).

Table B.18. Service

Field	Description
Service name	Name of service. This defines a collection of resources, known as a resource group or cluster service.
Automatically start this service	If enabled, this service (or resource group) is started automatically after the cluster forms a quorum. If this parameter is *disabled*, this service is *not* started automatically after the cluster forms a quorum; the service is put into the *`disabled`* state.
Run exclusive	If enabled, this service (resource group) can only be relocated to run on another node exclusively; that is, to run on a node that has no other services running on it. If no nodes are available for a service to run exclusively, the service is not restarted after a failure. Additionally, other services do not automatically relocate to a node running this service as *`Run exclusive`*. You can override this option by manual start or relocate operations.
Failover Domain	Defines lists of cluster members to try in the event that a service fails.
Recovery policy	*`Recovery policy`* provides the following options: • *`Disable`* — Disables the resource group if any component fails. • *`Relocate`* — Tries to restart service in another node; that is, it does not try to restart in the current node. • *`Restart`* — Tries to restart failed parts of this service locally (in the current node) before trying to relocate (default) to service to another node. • *`Restart-Disable`* — The service will be restarted in place if it fails. However, if restarting the service fails the service will be disabled instead of being moved to another host in the cluster.

Table B.19. Sybase ASE Failover Instance

Field	Description
Instance Name	Specifies the instance name of the Sybase ASE resource.
ASE server name	The ASE server name that is configured for the HA service.
Sybase home directory	The home directory of Sybase products.
Login file	The full path of login file that contains the login-password pair.
Interfaces file	The full path of the interfaces file that is used to start/access the ASE server.
SYBASE_ASE directory name	The directory name under sybase_home where ASE products are installed.
SYBASE_OCS directory name	The directory name under sybase_home where OCS products are installed. For example, ASE-15_0.

Field	Description
Sybase user	The user who can run ASE server.
Deep probe timeout	The maximum seconds to wait for the response of ASE server before determining that the server had no response while running deep probe.

Table B.20. Tomcat 6

Field	Description
Name	Specifies a service name for logging and other purposes.
Config File	Specifies the absolute path to the configuration file. The default value is **`/etc/tomcat6/tomcat6.conf`**.
Tomcat User	User who runs the Tomcat server. The default value is *tomcat*.
Catalina Options	Other command line options for Catalina.
Catalina Base	Catalina base directory (differs for each service) The default value is /usr/share/tomcat6.
Shutdown Wait (seconds)	Specifies the number of seconds to wait for correct end of service shutdown. The default value is 30.

Table B.21. Virtual Machine

Field	Description
Name	Specifies the name of the virtual machine. When using the **luci** interface, you specify this as a service name.
Automatically start this virtual machine	If enabled, this virtual machine is started automatically after the cluster forms a quorum. If this parameter is *disabled*, this virtual machine is *not* started automatically after the cluster forms a quorum; the virtual machine is put into the *`disabled`* state.
Run exclusive	If enabled, this virtual machine can only be relocated to run on another node exclusively; that is, to run on a node that has no other virtual machines running on it. If no nodes are available for a virtual machine to run exclusively, the virtual machine is not restarted after a failure. Additionally, other virtual machines do not automatically relocate to a node running this virtual machine as *`Run exclusive`*. You can override this option by manual start or relocate operations.
Failover domain	Defines lists of cluster members to try in the event that a virtual machine fails.
Recovery policy	*`Recovery policy`* provides the following options: • *`Disable`* — Disables the virtual machine if it fails. • *`Relocate`* — Tries to restart the virtual machine in another node; that is, it does not try to restart in the current node. • *`Restart`* — Tries to restart the virtual machine locally (in the current node) before trying to relocate (default) to virtual machine to another node. • *`Restart-Disable`* — The service will be restarted in place if it fails. However, if restarting the service fails the service will be disabled instead of moved to another host in the cluster.
Restart options	With **Restart** or **Restart-Disable** selected as the recovery policy for a service, specifies the maximum number of restart failures before relocating or disabling the service and specifies the length of time in seconds after which to forget a restart.

Field	Description
Migration type	Specifies a migration type of *live* or *pause*. The default setting is *live*.
Migration mapping	Specifies an alternate interface for migration. You can specify this when, for example, the network address used for virtual machine migration on a node differs from the address of the node used for cluster communication. Specifying the following indicates that when you migrate a virtual machine from **member** to **member2**, you actually migrate to **target2**. Similarly, when you migrate from **member2** to **member**, you migrate using **target**. **member:target,member2:target2**
Status Program	Status program to run in addition to the standard check for the presence of a virtual machine. If specified, the status program is executed once per minute. This allows you to ascertain the status of critical services within a virtual machine. For example, if a virtual machine runs a web server, your status program could check to see whether a web server is up and running; if the status check fails (signified by returning a non-zero value), the virtual machine is recovered. After a virtual machine is started, the virtual machine resource agent will periodically call the status program and wait for a successful return code (zero) prior to returning. This times out after five minutes.
Path to XML file used to create the VM	Full path to **libvirt** XML file containing the **libvirt** domain definition.
VM configuration file path	A colon-delimited path specification that the Virtual Machine Resource Agent (**vm.sh**) searches for the virtual machine configuration file. For example: **/mnt/guests/config:/etc/libvirt/qemu**. **Important** The path should *never* directly point to a virtual machine configuration file.
Path to VM snapshot directory	Path to the snapshot directory where the virtual machine image will be stored.
Hypervisor URI	Hypervisor URI (normally automatic).
Migration URI	Migration URI (normally automatic).

Appendix C. HA Resource Behavior

This appendix describes common behavior of HA resources. It is meant to provide ancillary information that may be helpful in configuring HA services. You can configure the parameters with **Luci** or by editing **`etc/cluster/cluster.conf`**. For descriptions of HA resource parameters, refer to *Appendix B, HA Resource Parameters*. To understand resource agents in more detail you can view them in **`/usr/share/cluster`** of any cluster node.

Note

To fully comprehend the information in this appendix, you may require detailed understanding of resource agents and the cluster configuration file, **`/etc/cluster/cluster.conf`**.

An HA service is a group of cluster resources configured into a coherent entity that provides specialized services to clients. An HA service is represented as a resource tree in the cluster configuration file, **`/etc/cluster/cluster.conf`** (in each cluster node). In the cluster configuration file, each resource tree is an XML representation that specifies each resource, its attributes, and its relationship among other resources in the resource tree (parent, child, and sibling relationships).

Note

Because an HA service consists of resources organized into a hierarchical tree, a service is sometimes referred to as a *resource tree* or *resource group*. Both phrases are synonymous with *HA service*.

At the root of each resource tree is a special type of resource — a *service resource*. Other types of resources comprise the rest of a service, determining its characteristics. Configuring an HA service consists of creating a service resource, creating subordinate cluster resources, and organizing them into a coherent entity that conforms to hierarchical restrictions of the service.

This appendix consists of the following sections:

- *Section C.1, “Parent, Child, and Sibling Relationships Among Resources”*
- *Section C.2, “Sibling Start Ordering and Resource Child Ordering”*
- *Section C.3, “Inheritance, the <resources> Block, and Reusing Resources”*
- *Section C.4, “Failure Recovery and Independent Subtrees”*
- *Section C.5, “Debugging and Testing Services and Resource Ordering”*

Note

The sections that follow present examples from the cluster configuration file, **/etc/cluster/cluster.conf**, for illustration purposes only.

C.1. Parent, Child, and Sibling Relationships Among Resources

A cluster service is an integrated entity that runs under the control of **rgmanager**. All resources in a service run on the same node. From the perspective of **rgmanager**, a cluster service is one entity that can be started, stopped, or relocated. Within a cluster service, however, the hierarchy of the resources determines the order in which each resource is started and stopped.The hierarchical levels consist of parent, child, and sibling.

Example C.1, "Resource Hierarchy of Service foo" shows a sample resource tree of the service *foo*. In the example, the relationships among the resources are as follows:

- **fs:myfs** (<fs name="myfs" ...>) and **ip:10.1.1.2** (<ip address="10.1.1.2 .../>) are siblings.
- **fs:myfs** (<fs name="myfs" ...>) is the parent of **script:script_child** (<script name="script_child"/>).
- **script:script_child** (<script name="script_child"/>) is the child of **fs:myfs** (<fs name="myfs" ...>).

Example C.1. Resource Hierarchy of Service foo

```
<service name="foo" ...>
    <fs name="myfs" ...>
        <script name="script_child"/>
    </fs>
    <ip address="10.1.1.2" .../>
</service>
```

The following rules apply to parent/child relationships in a resource tree:

- Parents are started before children.
- Children must all stop cleanly before a parent may be stopped.
- For a resource to be considered in good health, all its children must be in good health.

C.2. Sibling Start Ordering and Resource Child Ordering

The Service resource determines the start order and the stop order of a child resource according to whether it designates a child-type attribute for a child resource as follows:

- Designates child-type attribute (*typed* child resource) — If the Service resource designates a child-type attribute for a child resource, the child resource is *typed*. The child-type attribute explicitly determines the start and the stop order of the child resource.

- *Does not designate* child-type attribute (*non-typed* child resource) — If the Service resource *does not designate* a child-type attribute for a child resource, the child resource is *non-typed*. The Service resource does not explicitly control the starting order and stopping order of a non-typed child resource. However, a non-typed child resource is started and stopped according to its order in **`/etc/cluster.cluster.conf`** In addition, non-typed child resources are started after all typed child resources have started and are stopped before any typed child resources have stopped.

Note

The only resource to implement defined *child resource type* ordering is the Service resource.

For more information about typed child resource start and stop ordering, refer to *Section C.2.1, "Typed Child Resource Start and Stop Ordering"*. For more information about non-typed child resource start and stop ordering, refer to *Section C.2.2, "Non-typed Child Resource Start and Stop Ordering"*.

C.2.1. Typed Child Resource Start and Stop Ordering

For a typed child resource, the type attribute for the child resource defines the start order and the stop order of each resource type with a number that can range from 1 to 100; one value for start, and one value for stop. The lower the number, the earlier a resource type starts or stops. For example, *Table C.1, "Child Resource Type Start and Stop Order"* shows the start and stop values for each resource type; *Example C.2, "Resource Start and Stop Values: Excerpt from Service Resource Agent, **`service.sh`**"* shows the start and stop values as they appear in the Service resource agent, **`service.sh`**. For the Service resource, all LVM children are started first, followed by all File System children, followed by all Script children, and so forth.

Table C.1. Child Resource Type Start and Stop Order

Resource	Child Type	Start-order Value	Stop-order Value
LVM	lvm	1	9
File System	fs	2	8
GFS2 File System	clusterfs	3	7
NFS Mount	netfs	4	6
NFS Export	nfsexport	5	5
NFS Client	nfsclient	6	4
IP Address	ip	7	2
Samba	smb	8	3
Script	script	9	1

Example C.2. Resource Start and Stop Values: Excerpt from Service Resource Agent, **`service.sh`**

```
<special tag="rgmanager">
    <attributes root="1" maxinstances="1"/>
    <child type="lvm" start="1" stop="9"/>
    <child type="fs" start="2" stop="8"/>
    <child type="clusterfs" start="3" stop="7"/>
    <child type="netfs" start="4" stop="6"/>
    <child type="nfsexport" start="5" stop="5"/>
```

```
    <child type="nfsclient" start="6" stop="4"/>
    <child type="ip" start="7" stop="2"/>
    <child type="smb" start="8" stop="3"/>
    <child type="script" start="9" stop="1"/>
</special>
```

Ordering within a resource type is preserved as it exists in the cluster configuration file, **/etc/cluster/cluster.conf**. For example, consider the starting order and stopping order of the typed child resources in *Example C.3, "Ordering Within a Resource Type"*.

Example C.3. Ordering Within a Resource Type

```
<service name="foo">
  <script name="1" .../>
  <lvm name="1" .../>
  <ip address="10.1.1.1" .../>
  <fs name="1" .../>
  <lvm name="2" .../>
</service>
```

Typed Child Resource Starting Order

In *Example C.3, "Ordering Within a Resource Type"*, the resources are started in the following order:

1. **lvm:1** — This is an LVM resource. All LVM resources are started first. **lvm:1** (**<lvm name="1" .../>**) is the first LVM resource started among LVM resources because it is the first LVM resource listed in the Service *foo* portion of **/etc/cluster/cluster.conf**.

2. **lvm:2** — This is an LVM resource. All LVM resources are started first. **lvm:2** (**<lvm name="2" .../>**) is started after **lvm:1** because it is listed after **lvm:1** in the Service *foo* portion of **/etc/cluster/cluster.conf**.

3. **fs:1** — This is a File System resource. If there were other File System resources in Service *foo*, they would start in the order listed in the Service *foo* portion of **/etc/cluster/cluster.conf**.

4. **ip:10.1.1.1** — This is an IP Address resource. If there were other IP Address resources in Service *foo*, they would start in the order listed in the Service *foo* portion of **/etc/cluster/cluster.conf**.

5. **script:1** — This is a Script resource. If there were other Script resources in Service *foo*, they would start in the order listed in the Service *foo* portion of **/etc/cluster/cluster.conf**.

Typed Child Resource Stopping Order

In *Example C.3, "Ordering Within a Resource Type"*, the resources are stopped in the following order:

1. **script:1** — This is a Script resource. If there were other Script resources in Service *foo*, they would stop in the reverse order listed in the Service *foo* portion of **/etc/cluster/cluster.conf**.

2. **ip:10.1.1.1** — This is an IP Address resource. If there were other IP Address resources in Service *foo*, they would stop in the reverse order listed in the Service *foo* portion of **/etc/cluster/cluster.conf**.

3. **fs:1** — This is a File System resource. If there were other File System resources in Service *foo*, they would stop in the reverse order listed in the Service *foo* portion of **/etc/cluster/cluster.conf**.

4. **lvm:2** — This is an LVM resource. All LVM resources are stopped last. **lvm:2** (**<lvm name="2" .../>**) is stopped before **lvm:1**; resources within a group of a resource type are stopped in the reverse order listed in the Service *foo* portion of **/etc/cluster/cluster.conf**.

5. **lvm:1** — This is an LVM resource. All LVM resources are stopped last. **lvm:1** (**<lvm name="1" .../>**) is stopped after **lvm:2**; resources within a group of a resource type are stopped in the reverse order listed in the Service *foo* portion of **/etc/cluster/cluster.conf**.

C.2.2. Non-typed Child Resource Start and Stop Ordering

Additional considerations are required for non-typed child resources. For a non-typed child resource, starting order and stopping order are not explicitly specified by the Service resource. Instead, starting order and stopping order are determined according to the order of the child resource in **/etc/cluster.cluster.conf**. Additionally, non-typed child resources are started after all typed child resources and stopped before any typed child resources.

For example, consider the starting order and stopping order of the non-typed child resources in *Example C.4, "Non-typed and Typed Child Resource in a Service"*.

Example C.4. Non-typed and Typed Child Resource in a Service

```
<service name="foo">
  <script name="1" .../>
  <nontypedresource name="foo"/>
  <lvm name="1" .../>
  <nontypedresourcetwo name="bar"/>
  <ip address="10.1.1.1" .../>
  <fs name="1" .../>
  <lvm name="2" .../>
</service>
```

Non-typed Child Resource Starting Order

In *Example C.4, "Non-typed and Typed Child Resource in a Service"*, the child resources are started in the following order:

1. **lvm:1** — This is an LVM resource. All LVM resources are started first. **lvm:1** (**<lvm name="1" .../>**) is the first LVM resource started among LVM resources because it is the first LVM resource listed in the Service *foo* portion of **/etc/cluster/cluster.conf**.

2. **lvm:2** — This is an LVM resource. All LVM resources are started first. **lvm:2** (**<lvm name="2" .../>**) is started after **lvm:1** because it is listed after **lvm:1** in the Service *foo* portion of **/etc/cluster/cluster.conf**.

3. **fs:1** — This is a File System resource. If there were other File System resources in Service *foo*, they would start in the order listed in the Service *foo* portion of **/etc/cluster/cluster.conf**.

4. **ip:10.1.1.1** — This is an IP Address resource. If there were other IP Address resources in Service *foo*, they would start in the order listed in the Service *foo* portion of **/etc/cluster/cluster.conf**.

5. **`script:1`** — This is a Script resource. If there were other Script resources in Service *foo*, they would start in the order listed in the Service *foo* portion of **`/etc/cluster/cluster.conf`**.

6. **`nontypedresource:foo`** — This is a non-typed resource. Because it is a non-typed resource, it is started after the typed resources start. In addition, its order in the Service resource is before the other non-typed resource, **`nontypedresourcetwo:bar`**; therefore, it is started before **`nontypedresourcetwo:bar`**. (Non-typed resources are started in the order that they appear in the Service resource.)

7. **`nontypedresourcetwo:bar`** — This is a non-typed resource. Because it is a non-typed resource, it is started after the typed resources start. In addition, its order in the Service resource is after the other non-typed resource, **`nontypedresource:foo`**; therefore, it is started after **`nontypedresource:foo`**. (Non-typed resources are started in the order that they appear in the Service resource.)

Non-typed Child Resource Stopping Order

In *Example C.4, "Non-typed and Typed Child Resource in a Service"*, the child resources are stopped in the following order:

1. **`nontypedresourcetwo:bar`** — This is a non-typed resource. Because it is a non-typed resource, it is stopped before the typed resources are stopped. In addition, its order in the Service resource is after the other non-typed resource, **`nontypedresource:foo`**; therefore, it is stopped before **`nontypedresource:foo`**. (Non-typed resources are stopped in the reverse order that they appear in the Service resource.)

2. **`nontypedresource:foo`** — This is a non-typed resource. Because it is a non-typed resource, it is stopped before the typed resources are stopped. In addition, its order in the Service resource is before the other non-typed resource, **`nontypedresourcetwo:bar`**; therefore, it is stopped after **`nontypedresourcetwo:bar`**. (Non-typed resources are stopped in the reverse order that they appear in the Service resource.)

3. **`script:1`** — This is a Script resource. If there were other Script resources in Service *foo*, they would stop in the reverse order listed in the Service *foo* portion of **`/etc/cluster/cluster.conf`**.

4. **`ip:10.1.1.1`** — This is an IP Address resource. If there were other IP Address resources in Service *foo*, they would stop in the reverse order listed in the Service *foo* portion of **`/etc/cluster/cluster.conf`**.

5. **`fs:1`** — This is a File System resource. If there were other File System resources in Service *foo*, they would stop in the reverse order listed in the Service *foo* portion of **`/etc/cluster/cluster.conf`**.

6. **`lvm:2`** — This is an LVM resource. All LVM resources are stopped last. **`lvm:2`** (**`<lvm name="2" .../>`**) is stopped before **`lvm:1`**; resources within a group of a resource type are stopped in the reverse order listed in the Service *foo* portion of **`/etc/cluster/cluster.conf`**.

7. **`lvm:1`** — This is an LVM resource. All LVM resources are stopped last. **`lvm:1`** (**`<lvm name="1" .../>`**) is stopped after **`lvm:2`**; resources within a group of a resource type are stopped in the reverse order listed in the Service *foo* portion of **`/etc/cluster/cluster.conf`**.

C.3. Inheritance, the <resources> Block, and Reusing Resources

Some resources benefit by inheriting values from a parent resource; that is commonly the case in an NFS service. *Example C.5, "NFS Service Set Up for Resource Reuse and Inheritance"* shows a typical NFS service configuration, set up for resource reuse and inheritance.

Example C.5. NFS Service Set Up for Resource Reuse and Inheritance

```
<resources>
    <nfsclient name="bob" target="bob.example.com" options="rw,no_root_squash"/>
    <nfsclient name="jim" target="jim.example.com" options="rw,no_root_squash"/>
    <nfsexport name="exports"/>
</resources>
<service name="foo">
    <fs name="1" mountpoint="/mnt/foo" device="/dev/sdb1" fsid="12344">
        <nfsexport ref="exports">  <!-- nfsexport's path and fsid attributes
                                        are inherited from the mountpoint &
                                        fsid attribute of the parent fs
                                        resource -->
            <nfsclient ref="bob"/> <!-- nfsclient's path is inherited from the
                                        mountpoint and the fsid is added to the
                                        options string during export -->
            <nfsclient ref="jim"/>
        </nfsexport>
    </fs>
    <fs name="2" mountpoint="/mnt/bar" device="/dev/sdb2" fsid="12345">
        <nfsexport ref="exports">
            <nfsclient ref="bob"/> <!-- Because all of the critical data for this
                                        resource is either defined in the
                                        resources block or inherited, we can
                                        reference it again! -->
            <nfsclient ref="jim"/>
        </nfsexport>
    </fs>
    <ip address="10.2.13.20"/>
</service>
```

If the service were flat (that is, with no parent/child relationships), it would need to be configured as follows:

- The service would need four nfsclient resources — one per file system (a total of two for file systems), and one per target machine (a total of two for target machines).
- The service would need to specify export path and file system ID to each nfsclient, which introduces chances for errors in the configuration.

In *Example C.5, "NFS Service Set Up for Resource Reuse and Inheritance"* however, the NFS client resources *nfsclient:bob* and *nfsclient:jim* are defined once; likewise, the NFS export resource *nfsexport:exports* is defined once. All the attributes needed by the resources are inherited from parent resources. Because the inherited attributes are dynamic (and do not conflict with one another), it is possible to reuse those resources — which is why they are defined in the resources block. It may not be practical to configure some resources in multiple places. For example, configuring a file system resource in multiple places can result in mounting one file system on two nodes, therefore causing problems.

C.4. Failure Recovery and Independent Subtrees

In most enterprise environments, the normal course of action for failure recovery of a service is to restart the entire service if any component in the service fails. For example, in *Example C.6, "Service foo Normal Failure Recovery"*, if any of the scripts defined in this service fail, the normal course of action is to restart (or relocate or disable, according to the service recovery policy) the service. However, in some circumstances certain parts of a service may be considered non-critical; it may be necessary to restart only part of the service in place before attempting normal recovery. To accomplish that, you can use the `__independent_subtree` attribute. For example, in *Example C.7, "Service foo Failure Recovery with `__independent_subtree` Attribute"*, the `__independent_subtree` attribute is used to accomplish the following actions:

- If script:script_one fails, restart script:script_one, script:script_two, and script:script_three.
- If script:script_two fails, restart just script:script_two.
- If script:script_three fails, restart script:script_one, script:script_two, and script:script_three.
- If script:script_four fails, restart the whole service.

Example C.6. Service *foo* Normal Failure Recovery

```
<service name="foo">
      <script name="script_one" ...>
          <script name="script_two" .../>
      </script>
      <script name="script_three" .../>
</service>
```

Example C.7. Service *foo* Failure Recovery with `__independent_subtree` Attribute

```
<service name="foo">
      <script name="script_one" __independent_subtree="1" ...>
          <script name="script_two" __independent_subtree="1" .../>
          <script name="script_three" .../>
      </script>
      <script name="script_four" .../>
</service>
```

In some circumstances, if a component of a service fails you may want to disable only that component without disabling the entire service, to avoid affecting other services that use other components of that service. As of the Red Hat Enterprise Linux 6.1 release, you can accomplish that by using the `__independent_subtree="2"` attribute, which designates the independent subtree as non-critical.

Note

You may only use the non-critical flag on singly-referenced resources. The non-critical flag works with all resources at all levels of the resource tree, but should not be used at the top level when defining services or virtual machines.

As of the Red Hat Enterprise Linux 6.1 release, you can set maximum restart and restart expirations on a per-node basis in the resource tree for independent subtrees. To set these thresholds, you can use the following attributes:

- **__max_restarts** configures the maximum number of tolerated restarts prior to giving up.
- **__restart_expire_time** configures the amount of time, in seconds, after which a restart is no longer attempted.

C.5. Debugging and Testing Services and Resource Ordering

You can debug and test services and resource ordering with the **rg_test** utility. **rg_test** is a command-line utility provided by the **rgmanager** package that is run from a shell or a terminal (it is not available in **Conga**). *Table C.2, "**rg_test** Utility Summary"* summarizes the actions and syntax for the **rg_test** utility.

Table C.2. **rg_test** Utility Summary

Action	Syntax
Display the resource rules that **rg_test** understands.	**rg_test rules**
Test a configuration (and /usr/share/cluster) for errors or redundant resource agents.	**rg_test test /etc/cluster/cluster.conf**
Display the start and stop ordering of a service.	Display start order: **rg_test noop /etc/cluster/cluster.conf start service *servicename*** Display stop order: **rg_test noop /etc/cluster/cluster.conf stop service *servicename***
Explicitly start or stop a service.	**Important** Only do this on one node, and always disable the service in rgmanager first. Start a service: **rg_test test /etc/cluster/cluster.conf start service *servicename***

Action	Syntax
	Stop a service: **rg_test test /etc/cluster/cluster.conf stop service *servicename***
Calculate and display the resource tree delta between two cluster.conf files.	**rg_test delta *cluster.conf file 1* *cluster.conf file 2*** For example: **rg_test delta /etc/cluster/cluster.conf.bak /etc/cluster/cluster.conf**

Appendix D. Cluster Service Resource Check and Failover Timeout

This appendix describes how **rgmanager** monitors the status of cluster resources, and how to modify the status check interval. The appendix also describes the **__enforce_timeouts** service parameter, which indicates that a timeout for an operation should cause a service to fail.

Note

To fully comprehend the information in this appendix, you may require detailed understanding of resource agents and the cluster configuration file, **/etc/cluster/cluster.conf**. For a comprehensive list and description of **cluster.conf** elements and attributes, refer to the cluster schema at **/usr/share/cluster/cluster.rng**, and the annotated schema at **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** (for example **/usr/share/doc/cman-3.0.12/cluster_conf.html**).

D.1. Modifying the Resource Status Check Interval

rgmanager checks the status of individual resources, not whole services. Every 10 seconds, rgmanager scans the resource tree, looking for resources that have passed their "status check" interval.

Each resource agent specifies the amount of time between periodic status checks. Each resource utilizes these timeout values unless explicitly overridden in the **cluster.conf** file using the special **<action>** tag:

```
<action name="status" depth="*" interval="10" />
```

This tag is a special child of the resource itself in the **cluster.conf** file. For example, if you had a file system resource for which you wanted to override the status check interval you could specify the file system resource in the **cluster.conf** file as follows:

```
<fs name="test" device="/dev/sdb3">
  <action name="status" depth="*" interval="10" />
  <nfsexport...>
  </nfsexport>
</fs>
```

Some agents provide multiple "depths" of checking. For example, a normal file system status check (depth 0) checks whether the file system is mounted in the correct place. A more intensive check is depth 10, which checks whether you can read a file from the file system. A status check of depth 20 checks whether you can write to the file system. In the example given here, the **depth** is set to *****, which indicates that these values should be used for all depths. The result is that the **test** file system is checked at the highest-defined depth provided by the resource-agent (in this case, 20) every 10 seconds.

D.2. Enforcing Resource Timeouts

There is no timeout for starting, stopping, or failing over resources. Some resources take an indeterminately long amount of time to start or stop. Unfortunately, a failure to stop (including a timeout) renders the service inoperable (failed state). You can, if desired, turn on timeout enforcement on each resource in a service individually by adding **__enforce_timeouts="1"** to the reference in the **cluster.conf** file.

The following example shows a cluster service that has been configured with the **__enforce_timeouts** attribute set for the **netfs** resource. With this attribute set, then if it takes more than 30 seconds to unmount the NFS file system during a recovery process the operation will time out, causing the service to enter the failed state.

```
</screen>
<rm>
  <failoverdomains/>
  <resources>
    <netfs export="/nfstest" force_unmount="1" fstype="nfs" host="10.65.48.65"
           mountpoint="/data/nfstest" name="nfstest_data" options="rw,sync,soft"/>
  </resources>
  <service autostart="1" exclusive="0" name="nfs_client_test" recovery="relocate">
    <netfs ref="nfstest_data" __enforce_timeouts="1"/>
  </service>
</rm>
```

Appendix E. Command Line Tools Summary

Table E.1, "Command Line Tool Summary" summarizes preferred command-line tools for configuring and managing the High Availability Add-On. For more information about commands and variables, refer to the man page for each command-line tool.

Table E.1. Command Line Tool Summary

Command Line Tool	Used With	Purpose
ccs_config_dump — Cluster Configuration Dump Tool	Cluster Infrastructure	**ccs_config_dump** generates XML output of running configuration. The running configuration is, sometimes, different from the stored configuration on file because some subsystems store or set some default information into the configuration. Those values are generally not present on the on-disk version of the configuration but are required at runtime for the cluster to work properly. For more information about this tool, refer to the ccs_config_dump(8) man page.
ccs_config_validate — Cluster Configuration Validation Tool	Cluster Infrastructure	**ccs_config_validate** validates **cluster.conf** against the schema, **cluster.rng** (located in **/usr/share/cluster/cluster.rng** on each node). For more information about this tool, refer to the ccs_config_validate(8) man page.
clustat — Cluster Status Utility	High-availability Service Management Components	The **clustat** command displays the status of the cluster. It shows membership information, quorum view, and the state of all configured user services. For more information about this tool, refer to the clustat(8) man page.
clusvcadm — Cluster User Service Administration Utility	High-availability Service Management Components	The **clusvcadm** command allows you to enable, disable, relocate, and restart high-availability services in a cluster. For more information about this tool, refer to the clusvcadm(8) man page.
cman_tool — Cluster Management Tool	Cluster Infrastructure	**cman_tool** is a program that manages the CMAN cluster manager. It provides the capability to join a cluster, leave a cluster, kill a node, or change the expected quorum votes of a node in a cluster. For more information about this tool, refer to the cman_tool(8) man page.
fence_tool — Fence Tool	Cluster Infrastructure	**fence_tool** is a program used to join and leave the fence domain. For more information about this tool, refer to the fence_tool(8) man page.

Appendix F. Revision History

Revision 3.0-5 Thu Dec 1 2011 **Steven Levine *slevine@redhat.com***

Release for GA of Red Hat Enterprise Linux 6.2

Resolves: 755849
Corrects monitor_link parameter example.

Revision 3.0-4 Mon Nov 7 2011 **Steven Levine *slevine@redhat.com***

Resolves: 749857
Adds documentation for RHEV-M REST API fence device.

Revision 3.0-3 Fri Oct 21 2011 **Steven Levine *slevine@redhat.com***

Resolves: #747181, #747182, #747184, #747185, #747186, #747187, #747188, #747189, #747190, #747192
Corrects typographical errors and ambiguities found during documentation QE review for Red Hat Enterprise Linux 6.2.

Revision 3.0-2 Fri Oct 7 2011 **Steven Levine *slevine@redhat.com***

Resolves: #743757
Corrects reference to supported bonding mode in troubleshooting section.

Revision 3.0-1 Wed Sep 28 2011 **Steven Levine *slevine@redhat.com***

Initial revision for Red Hat Enterprise Linux 6.2 Beta release

Resolves: #739613
Documents support for new **`ccs`** options to display available fence devices and available services.

Resolves: #707740
Documents updates to the Conga interface and documents support for setting user permissions to administer Conga.

Resolves: #731856
Documents supports for configuring **luci** by means of the **`/etc/sysconfig/luci`** file.

Resolves: #736134
Documents support for UDPU transport.

Resolves: #736143
Documents support for clustered Samba.

Resolves: #617634
Documents how to configure the only IP address **luci** is served at.

Resolves: #713259
Documents support for **`fence_vmware_soap`** agent.

Resolves: #721009
Provides link to Support Essentials article.

Resolves: #717006
Provides information on allowing multicast traffic through the **iptables** firewall.

Resolves: #717008
Provides information about cluster service status check and failover timeout.

Resolves: #711868
Clarifies description of autostart.

Resolves: #728337
Documents procedure for adding **vm** resources with the **ccs** command.

Resolves: #725315, #733011, #733074, #733689
Corrects small typographical errors.

Revision 2.0-1 Thu May 19 2011 **Steven Levine** *slevine@redhat.com*

Initial revision for Red Hat Enterprise Linux 6.1

Resolves: #671250
Documents support for SNMP traps.

Resolves: #659753
Documents **ccs** command.

Resolves: #665055
Updates Conga documentation to reflect updated display and feature support.

Resolves: #680294
Documents need for password access for **ricci** agent.

Resolves: #687871
Adds chapter on troubleshooting.

Resolves: #673217
Fixes typographical error.

Resolves: #675805
Adds reference to **cluster.conf** schema to tables of HA resource parameters.

Resolves: #672697
Updates tables of fence device parameters to include all currently supported fencing devices.

Resolves: #677994
Corrects information for **fence_ilo** fence agent parameters.

Resolves: #629471
Adds technical note about setting consensus value in a two-node cluster.

Resolves: #579585
Updates section on upgrading Red Hat High Availability Add-On Software.

Resolves: #643216
Clarifies small issues throughout document.

Resolves: #643191
Provides improvements and corrections for the **luci** documentation.

Resolves: #704539
Updates the table of Virtual Machine resource parameters.

Revision 1.0-1 Wed Nov 10 2010 **Paul Kennedy** *pkennedy@redhat.com*
Initial release for Red Hat Enterprise Linux 6

Index

A

ACPI
  configuring, 11

B

behavior, HA resources,

C

cluster
  administration, , , ,
  diagnosing and correcting problems, 86,
  starting, stopping, restarting, 110
cluster administration, , , ,
  adding cluster node, 48, 85
  compatible hardware, 8
  configuration validation, 18
  configuring ACPI, 11
  configuring iptables, 9
  considerations for using qdisk, 21
  considerations for using quorum disk, 21
  deleting a cluster, 50
  deleting a node from the configuration; adding a node to the configuration , 111
  diagnosing and correcting problems in a cluster, 86,
  displaying HA services with clustat, 119
  enabling IP ports, 9
  general considerations, 7
  joining a cluster, 48, 85
  leaving a cluster, 48, 85
  managing cluster node, 47, 85
  managing high-availability services, 51, 119
  managing high-availability services, freeze and unfreeze, 120, 122
  network switches and multicast addresses, 22
  NetworkManager, 21
  rebooting cluster node, 47
  removing cluster node, 49
  restarting a cluster, 50
  ricci considerations, 24
  SELinux, 22
  starting a cluster, 50, 86
  starting, stopping, restarting a cluster, 110
  stopping a cluster, 50, 86
  updating a cluster configuration using cman_tool version -r, 123
  updating a cluster configuration using scp, 125
  updating configuration, 122
cluster configuration, , ,
  deleting or adding a node, 111
  updating, 122
cluster resource relationships, 164
cluster resource status check,
cluster resource types, 18
cluster service managers
  configuration, 44, 75, 103
cluster services, 44, 75, 103
  (see also adding to the cluster configuration)
cluster software
  configuration, , ,
configuration
  HA service, 16
Conga
  accessing, 5
consensus value, 91

F

failover timeout,
features, new and changed, 1
feedback, x, x

G

general
  considerations for cluster administration, 7

H

HA service configuration
  overview, 16
hardware
  compatible, 8

I

integrated fence devices
  configuring ACPI, 11
introduction,
  other Red Hat Enterprise Linux documents,

IP ports
  enabling, 9
iptables
  configuring, 9
iptables firewall, 23

M

multicast addresses
  considerations for using with network switches and multicast addresses, 22
multicast traffic, enabling, 23

N

NetworkManager
  disable for use with cluster, 21

O

overview
features, new and changed, 1

P

parameters, fence device,
parameters, HA resources,
power controller connection, configuring,
power switch,
(see also power controller)

Q

qdisk
considerations for using, 21
quorum disk
considerations for using, 21

R

relationships
cluster resource, 164
ricci
considerations for cluster administration, 24

S

SELinux
configuring, 22
status check, cluster resource,

T

tables
HA resources, parameters,
power controller connection, configuring,
timeout failover,
tools, command line,
totem tag
consensus value, 91
troubleshooting
diagnosing and correcting problems in a
cluster, 86,
types
cluster resource, 18

V

validation
cluster configuration, 18