SUSE Linux Enterprise High Availability Extension

11 SP1

www.novell.com

May 04, 2011

High Availability Guide

A NOVELL BUSINESS

High Availability Guide

All information found in this book has been compiled with utmost attention to detail. However, this does not guarantee complete accuracy. Neither Novell, Inc., SUSE LINUX Products GmbH, the authors, nor the translators shall be held liable for possible errors or the consequences thereof.

Contents

About This Guide

SUSE® Linux Enterprise High Availability Extension is an integrated suite of open source clustering technologies that enables you to implement highly available physical and virtual Linux clusters. For quick and efficient configuration and administration, the High Availability Extension includes both a graphical user interface (GUI) and a command line interface (CLI). Additionally, it comes with the HA Web Konsole, allowing you to administer your Linux cluster also via a Web interface.

This guide is intended for administrators who need to set up, configure, and maintain High Availability (HA) clusters. Both approaches (GUI and CLI) are covered in detail to help the administrators choose the appropriate tool that matches their needs for performing the key tasks.

The guide is divided into the following parts:

Installation and Setup

Before starting to install and configure your cluster, make yourself familiar with cluster fundamentals and architecture, get an overview of the key features and benefits. Learn which hardware and software requirements must be met and what preparations to take before executing the next steps. Perform the installation and basic setup of your HA cluster using YaST.

Configuration and Administration

Add, configure and manage resources, using either the graphical user interface (Pacemaker GUI) or the `crm` command line interface. Use the HA Web Konsole if you want or need to monitor your cluster via a Web interface. Learn how to make use of load balancing and fencing. In case you consider writing your own resource agents or modifying existing ones, get some background information on how to create different types of resource agents.

Storage and Data Replication

SUSE Linux Enterprise High Availability Extension ships with a cluster-aware file system and volume manager: Oracle Cluster File System (OCFS2) and the clustered Logical Volume Manager (cLVM). For replication of your data, use DRBD (Distributed Replicated Block Device) to mirror the data of an High Availability service from the active node of a cluster to its standby node. Furthermore, a clustered Samba server provides an High Availability solution also for heterogeneous environments.

Troubleshooting and Reference

Managing your own cluster requires you to perform a certain amount of troubleshooting. Learn about the most common problems and how to fix them. Find a comprehensive reference of the command line tools the High Availability Extension offers for administering your own cluster.

Appendix

Lists the new features and behavior changes of the High Availability Extension since the last release. Learn how to migrate your cluster to the most recent release version and find an example of setting up a simple testing resource.

Many chapters in this manual contain links to additional documentation resources. These include additional documentation that is available on the system as well as documentation available on the Internet.

For an overview of the documentation available for your product and the latest documentation updates, refer to `http://www.novell.com/documentation`.

1 Feedback

Several feedback channels are available:

Bugs and Enhancement Requests

For services and support options available for your product, refer to `http://www.novell.com/services/`.

To report bugs for a product component, please use `http://support.novell.com/additional/bugreport.html`.

Submit enhancement requests at `https://secure-www.novell.com/rms/rmsTool?action=ReqActions.viewAddPage&return=www`.

User Comments

We want to hear your comments and suggestions about this manual and the other documentation included in this product. Use the User Comments feature at the bottom of each page in the online documentation or go to `http://www.novell.com/documentation/feedback.html` and enter your comments there.

2 Documentation Conventions

The following typographical conventions are used in this manual:

- `/etc/passwd`: directory names and filenames
- *`placeholder`*: replace *`placeholder`* with the actual value
- `PATH`: the environment variable PATH
- `ls`, `--help`: commands, options, and parameters
- `user`: users or groups
- Alt, Alt + F1: a key to press or a key combination; keys are shown in uppercase as on a keyboard
- *File*, *File* > *Save As*: menu items, buttons
- ► **amd64 em64t:** This paragraph is only relevant for the specified architectures. The arrows mark the beginning and the end of the text block. ◄
- *Dancing Penguins* (Chapter *Penguins*, ↑Another Manual): This is a reference to a chapter in another manual.

Part I. Installation and Setup

1 Product Overview

SUSE® Linux Enterprise High Availability Extension is an integrated suite of open source clustering technologies that enables you to implement highly available physical and virtual Linux clusters, and to eliminate single points of failure. It ensures the high availability and manageability of critical network resources including data, applications, and services. Thus, it helps you maintain business continuity, protect data integrity, and reduce unplanned downtime for your mission-critical Linux workloads.

It ships with essential monitoring, messaging, and cluster resource management functionality (supporting failover, failback, and migration (load balancing) of individually managed cluster resources). The High Availability Extension is available as add-on to SUSE Linux Enterprise Server 11 SP1.

This chapter introduces the main product features and benefits of the High Availability Extension. Inside you will find several example clusters and learn about the components making up a cluster. The last section provides an overview of the architecture, describing the individual architecture layers and processes within the cluster.

For explanations of some common terms used in the context of High Availability clusters, refer to Terminology (page 395).

1.1 Key Features

SUSE® Linux Enterprise High Availability Extension helps you ensure and manage the availability of your network resources. The following sections highlight some of the key features:

1.1.1 Wide Range of Clustering Scenarios

The High Availability Extension supports the following scenarios:

- Active/active configurations
- Active/passive configurations: N+1, N+M, N to 1, N to M
- Hybrid physical and virtual clusters, allowing virtual servers to be clustered with physical servers. This improves services availability and resource utilization.

Your cluster can contain up to 32 Linux servers. Any server in the cluster can restart resources (applications, services, IP addresses, and file systems) from a failed server in the cluster.

1.1.2 Flexibility

The High Availability Extension ships with Corosync/OpenAIS messaging and membership layer and Pacemaker Cluster Resource Manager. Using Pacemaker, administrators can continually monitor the health and status of their resources, manage dependencies, and automatically stop and start services based on highly configurable rules and policies. The High Availability Extension allows you to tailor a cluster to the specific applications and hardware infrastructure that fit your organization. Time-dependent configuration enables services to automatically migrate back to repaired nodes at specified times.

1.1.3 Storage and Data Replication

With the High Availability Extension you can dynamically assign and reassign server storage as needed. It supports Fibre Channel or iSCSI storage area networks (SANs). Shared disk systems are also supported, but they are not a requirement. SUSE Linux Enterprise High Availability Extension also comes with a cluster-aware file system and volume manager: Oracle Cluster File System (OCFS2) and the clustered Logical Volume Manager (cLVM). For replication of your data, you can use DRBD (Distributed Replicated Block Device) to mirror the data of an High Availability service from the active node of a cluster to its standby node. Furthermore, SUSE Linux Enterprise High Availability Extension also supports CTDB (Clustered Trivial Database), a technology for Samba clustering.

1.1.4 Support for Virtualized Environments

SUSE Linux Enterprise High Availability Extension supports the mixed clustering of both physical and virtual Linux servers. SUSE Linux Enterprise Server 11 SP1 ships with Xen, an open source virtualization hypervisor and with KVM (Kernel-based Virtual Machine), a virtualization software for Linux which is based on hardware virtualization extensions. The cluster resource manager in the High Availability Extension is able to recognize, monitor and manage services running within virtual servers, as well as services running in physical servers. Guest systems can be managed as services by the cluster.

1.1.5 Resource Agents

SUSE Linux Enterprise High Availability Extension includes a huge number of resource agents to manage resources such as Apache, IPv4, IPv6 and many more. It also ships with resource agents for popular third party applications such as IBM WebSphere Application Server. For a list of Open Cluster Framework (OCF) resource agents included with your product, refer to Chapter 19, *HA OCF Agents* (page 269).

1.1.6 User-friendly Administration Tools

The High Availability Extension ships with a set of powerful tools for basic installation and setup of your cluster as well as effective configuration and administration:

YaST

> A graphical user interface for general system installation and administration. Use it to install the High Availability Extension on top of SUSE Linux Enterprise Server as described in Section 3.1, “Installing the High Availability Extension” (page 21). YaST also contains the following modules in the High Availability category that help you to configure your cluster or individual components:
>
> - Cluster: Basic cluster setup. For details, refer to Section 3.2, “Initial Cluster Setup” (page 22).
>
> - DRBD: Configuration of a Distributed Replicated Block Device.

- IP Load Balancing: Configuration of load balancing with Linux Virtual Server. For details, refer to Chapter 10, *Load Balancing with Linux Virtual Server* (page 137).

Pacemaker GUI

Installable graphical user interface for easy configuration and administration of your cluster. Guides you through the creation and configuration of resources and lets you execute management tasks like starting, stopping or migrating resources. For details, refer to Chapter 5, *Configuring and Managing Cluster Resources (GUI)* (page 57).

HA Web Konsole

A Web-based user interface with which you can administer your Linux cluster also from non-Linux machines. It is also an ideal solution in case your system does not provide a graphical user interface. For details, refer to Chapter 7, *Managing Cluster Resources with the Web Interface* (page 113).

`crm`

A powerful unified command line interface. Helps you to configure resources and to execute all monitoring or administration tasks. For details, refer to Chapter 6, *Configuring and Managing Cluster Resources (Command Line)* (page 89).

1.2 Benefits

The High Availability Extension allows you to configure up to 32 Linux servers into a high-availability cluster (HA cluster), where resources can be dynamically switched or moved to any server in the cluster. Resources can be configured to automatically migrate in the event of a server failure, or they can be moved manually to troubleshoot hardware or balance the workload.

The High Availability Extension provides high availability from commodity components. Lower costs are obtained through the consolidation of applications and operations onto a cluster. The High Availability Extension also allows you to centrally manage the complete cluster and to adjust resources to meet changing workload requirements (thus, manually “load balance” the cluster). Allowing clusters of more than two nodes also provides savings by allowing several nodes to share a “hot spare”.

An equally important benefit is the potential reduction of unplanned service outages as well as planned outages for software and hardware maintenance and upgrades.

Reasons that you would want to implement a cluster include:

- Increased availability
- Improved performance
- Low cost of operation
- Scalability
- Disaster recovery
- Data protection
- Server consolidation
- Storage consolidation

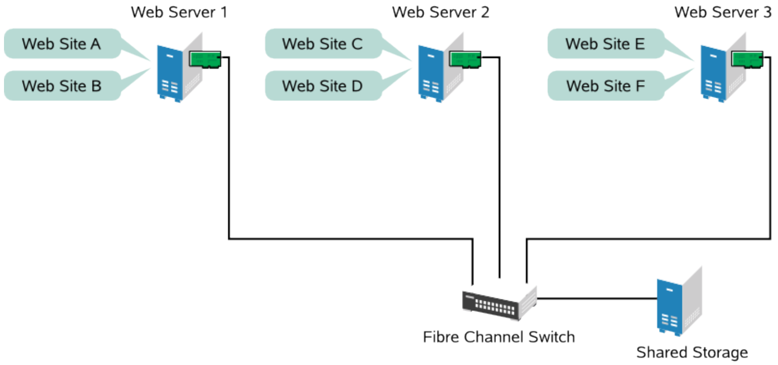
Shared disk fault tolerance can be obtained by implementing RAID on the shared disk subsystem.

The following scenario illustrates some of the benefits the High Availability Extension can provide.

Example Cluster Scenario

Suppose you have configured a three-server cluster, with a Web server installed on each of the three servers in the cluster. Each of the servers in the cluster hosts two Web sites. All the data, graphics, and Web page content for each Web site are stored on a shared disk subsystem connected to each of the servers in the cluster. The following figure depicts how this setup might look.

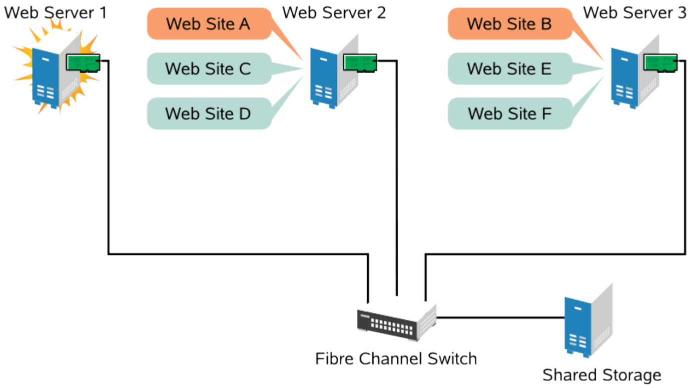
Figure 1.1 *Three-Server Cluster*



During normal cluster operation, each server is in constant communication with the other servers in the cluster and performs periodic polling of all registered resources to detect failure.

Suppose Web Server 1 experiences hardware or software problems and the users depending on Web Server 1 for Internet access, e-mail, and information lose their connections. The following figure shows how resources are moved when Web Server 1 fails.

Figure 1.2 *Three-Server Cluster after One Server Fails*



Web Site A moves to Web Server 2 and Web Site B moves to Web Server 3. IP addresses and certificates also move to Web Server 2 and Web Server 3.

When you configured the cluster, you decided where the Web sites hosted on each Web server would go should a failure occur. In the previous example, you configured Web Site A to move to Web Server 2 and Web Site B to move to Web Server 3. This way, the workload once handled by Web Server 1 continues to be available and is evenly distributed between any surviving cluster members.

When Web Server 1 failed, the High Availability Extension software did the following:

- Detected a failure and verified with STONITH that Web Server 1 was really dead. STONITH is an acronym for "Shoot The Other Node In The Head" and is a means of bringing down misbehaving nodes to prevent them from causing trouble in the cluster.

- Remounted the shared data directories that were formerly mounted on Web server 1 on Web Server 2 and Web Server 3.

- Restarted applications that were running on Web Server 1 on Web Server 2 and Web Server 3.

- Transferred IP addresses to Web Server 2 and Web Server 3.

In this example, the failover process happened quickly and users regained access to Web site information within seconds, and in most cases, without needing to log in again.

Now suppose the problems with Web Server 1 are resolved, and Web Server 1 is returned to a normal operating state. Web Site A and Web Site B can either automatically fail back (move back) to Web Server 1, or they can stay where they are. This is dependent on how you configured the resources for them. Migrating the services back to Web Server 1 will incur some down-time, so the High Availability Extension also allows you to defer the migration until a period when it will cause little or no service interruption. There are advantages and disadvantages to both alternatives.

The High Availability Extension also provides resource migration capabilities. You can move applications, Web sites, etc. to other servers in your cluster as required for system management.

For example, you could have manually moved Web Site A or Web Site B from Web Server 1 to either of the other servers in the cluster. You might want to do this to upgrade or perform scheduled maintenance on Web Server 1, or just to increase performance or accessibility of the Web sites.

1.3 Cluster Configurations: Storage

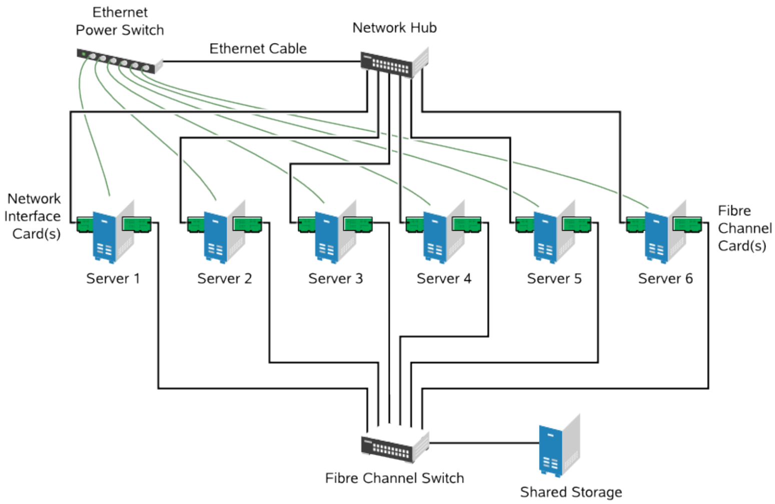
Cluster configurations with the High Availability Extension might or might not include a shared disk subsystem. The shared disk subsystem can be connected via high-speed Fibre Channel cards, cables, and switches, or it can be configured to use iSCSI. If a server fails, another designated server in the cluster automatically mounts the shared disk directories that were previously mounted on the failed server. This gives network users continuous access to the directories on the shared disk subsystem.

IMPORTANT: Shared Disk Subsystem with cLVM

When using a shared disk subsystem with cLVM, that subsystem must be connected to all servers in the cluster from which it needs to be accessed.

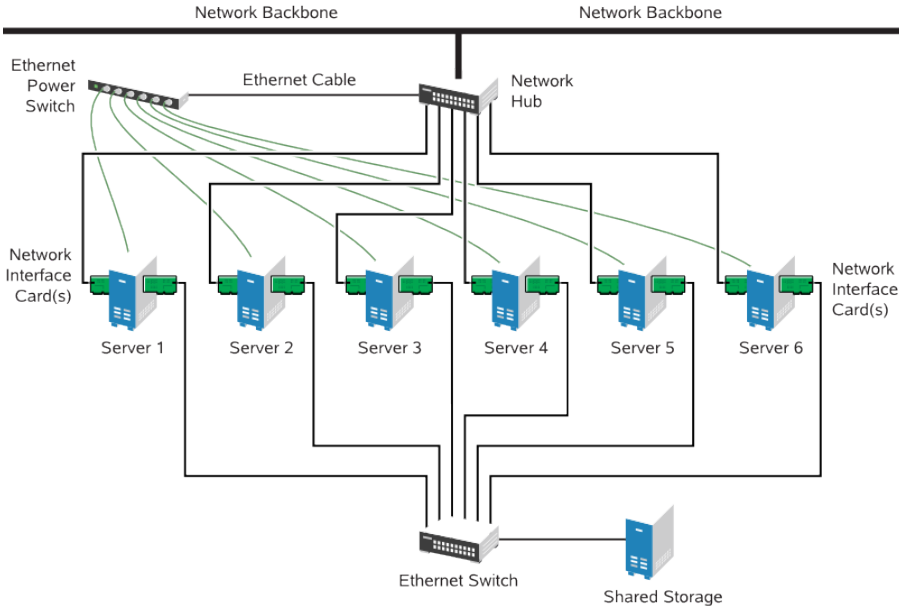
Typical resources might include data, applications, and services. The following figure shows how a typical Fibre Channel cluster configuration might look.

Figure 1.3 *Typical Fibre Channel Cluster Configuration*



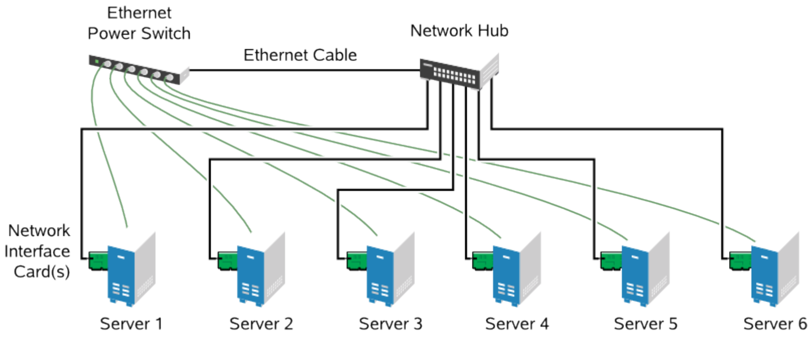
Although Fibre Channel provides the best performance, you can also configure your cluster to use iSCSI. iSCSI is an alternative to Fibre Channel that can be used to create a low-cost Storage Area Network (SAN). The following figure shows how a typical iSCSI cluster configuration might look.

Figure 1.4 *Typical iSCSI Cluster Configuration*



Although most clusters include a shared disk subsystem, it is also possible to create a cluster without a shared disk subsystem. The following figure shows how a cluster without a shared disk subsystem might look.

Figure 1.5 Typical Cluster Configuration Without Shared Storage



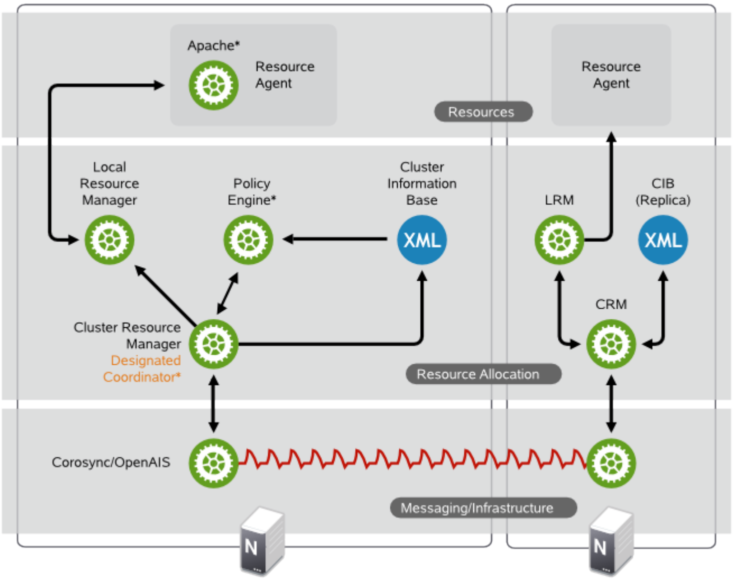
1.4 Architecture

This section provides a brief overview of the High Availability Extension architecture. It identifies and provides information on the architectural components, and describes how those components interoperate.

1.4.1 Architecture Layers

The High Availability Extension has a layered architecture. Figure 1.6, “Architecture” (page 13) illustrates the different layers and their associated components.

Figure 1.6 Architecture



Messaging and Infrastructure Layer

The primary or first layer is the messaging/infrastructure layer, also known as the Corosync/OpenAIS layer. This layer contains components that send out the messages containing “I'm alive” signals, as well as other information. The program of the High Availability Extension resides in the messaging/infrastructure layer.

Resource Allocation Layer

The next layer is the resource allocation layer. This layer is the most complex, and consists of the following components:

Cluster Resource Manager (CRM)

Every action taken in the resource allocation layer passes through the Cluster Resource Manager. If other components of the resource allocation layer (or components which are in a higher layer) need to communicate, they do so through the local CRM.

On every node, the CRM maintains the Cluster Information Base (CIB) (page 14), containing definitions of all cluster options, nodes, resources their relationship and current status. One CRM in the cluster is elected as the Designated Coordinator (DC), meaning that it has the master CIB. All other CIBs in the cluster are a replicas of the master CIB. Normal read and write operations on the CIB are serialized through the master CIB. The DC is the only entity in the cluster that can decide that a cluster-wide change needs to be performed, such as fencing a node or moving resources around.

Cluster Information Base (CIB)

The Cluster Information Base is an in-memory XML representation of the entire cluster configuration and current status. It contains definitions of all cluster options, nodes, resources, constraints and the relationship to each other. The CIB also synchronizes updates to all cluster nodes. There is one master CIB in the cluster, maintained by the DC. All other nodes contain a CIB replica.

Policy Engine (PE)

Whenever the Designated Coordinator needs to make a cluster-wide change (react to a new CIB), the Policy Engine calculates the next state of the cluster based on the current state and configuration. The PE also produces a transition graph containing a list of (resource) actions and dependencies to achieve the next cluster state. The PE runs on every node to speed up DC failover.

Local Resource Manager (LRM)

The LRM calls the local Resource Agents (see Section “Resource Layer” (page 14)) on behalf of the CRM. It can thus perform start / stop / monitor operations and report the result to the CRM. It also hides the difference between the supported script standards for Resource Agents (OCF, LSB, Heartbeat Version 1). The LRM is the authoritative source for all resource-related information on its local node.

Resource Layer

The highest layer is the Resource Layer. The Resource Layer includes one or more Resource Agents (RA). Resource Agents are programs (usually shell scripts) that have been written to start, stop, and monitor a certain kind of service (a resource). Resource Agents are called only by the LRM. Third parties can include their own agents in a defined location in the file system and thus provide out-of-the-box cluster integration for their own software.

1.4.2 Process Flow

SUSE Linux Enterprise High Availability Extension uses Pacemaker as CRM. The CRM is implemented as daemon (`crmd`) that has an instance on each cluster node. Pacemaker centralizes all cluster decision-making by electing one of the crmd instances to act as a master. Should the elected crmd process (or the node it is on) fail, a new one is established.

A CIB, reflecting the cluster's configuration and current state of all resources in the cluster is kept on each node. The contents of the CIB are automatically kept in sync across the entire cluster.

Many actions performed in the cluster will cause a cluster-wide change. These actions can include things like adding or removing a cluster resource or changing resource constraints. It is important to understand what happens in the cluster when you perform such an action.

For example, suppose you want to add a cluster IP address resource. To do this, you can use one of the command line tools or the GUI to modify the CIB. It is not required to perform the actions on the DC, you can use either tool on any node in the cluster and they will be relayed to the DC. The DC will then replicate the CIB change to all cluster nodes.

Based on the information in the CIB, the PE then computes the ideal state of the cluster and how it should be achieved and feeds a list of instructions to the DC. The DC sends commands via the messaging/infrastructure layer which are received by the crmd peers on other nodes. Each crmd uses it LRM (implemented as lrmd) to perform resource modifications. The lrmd is non-cluster aware and interacts directly with resource agents (scripts).

The peer nodes all report the results of their operations back to the DC. Once the DC concludes that all necessary operations are successfully performed in the cluster, the cluster will go back to the idle state and wait for further events. If any operation was not carried out as planned, the PE is invoked again with the new information recorded in the CIB.

In some cases, it may be necessary to power off nodes in order to protect shared data or complete resource recovery. For this Pacemaker comes with a fencing subsystem, stonithd. STONITH is an acronym for “Shoot The Other Node In The Head” and is usually implemented with a remote power switch. In Pacemaker, STONITH devices are modeled as resources (and configured in the CIB) to enable them to be easily monitored for failure. However, stonithd takes care of understanding the STONITH topology such that its clients simply request a node be fenced and it does the rest.

Getting Started 2

In the following, learn about the system requirements and which preparations to take before installing the High Availability Extension. Find a short overview of the basic steps to install and set up a cluster.

2.1 Hardware Requirements

The following list specifies hardware requirements for a cluster based on SUSE® Linux Enterprise High Availability Extension. These requirements represent the minimum hardware configuration. Additional hardware might be necessary, depending on how you intend to use your cluster.

- 1 to 32 Linux servers with software as specified in Section 2.2, “Software Requirements” (page 18). The servers do not require identical hardware (memory, disk space, etc.).

- At least two TCP/IP communication media. Cluster nodes use multicast for communication so the network equipment must support multicasting. The communication media should support a data rate of 100 Mbit/s or higher. Preferably, the Ethernet channels should be bonded.

- Optional: A shared disk subsystem connected to all servers in the cluster from where it needs to be accessed.

- A STONITH mechanism. STONITH is an acronym for “Shoot the other node in the head”. A STONITH device is a power switch which the cluster uses to reset nodes that are thought to be dead or behaving in a strange manner. Resetting non-heartbeating nodes is the only reliable way to ensure that no data corruption is performed by nodes that hang and only appear to be dead.

For more information, refer to Chapter 9, *Fencing and STONITH* (page 125).

2.2 Software Requirements

Ensure that the following software requirements are met:

- SUSE® Linux Enterprise Server 11 SP1 with all available online updates installed on all nodes that will be part of the cluster.

- SUSE Linux Enterprise High Availability Extension 11 SP1 including all available online updates installed on all nodes that will be part of the cluster.

2.3 Shared Disk System Requirements

A shared disk system (Storage Area Network, or SAN) is recommended for your cluster if you want data to be highly available. If a shared disk subsystem is used, ensure the following:

- The shared disk system is properly set up and functional according to the manufacturer's instructions.

- The disks contained in the shared disk system should be configured to use mirroring or RAID to add fault tolerance to the shared disk system. Hardware-based RAID is recommended. Host-based software RAID is not supported for all configurations.

- If you are using iSCSI for shared disk system access, ensure that you have properly configured iSCSI initiators and targets.

- When using DRBD to implement a mirroring RAID system that distributes data across two machines, make sure to only access the replicated device. Use the same (bonded) NICs that the rest of the cluster uses to leverage the redundancy provided there.

2.4 Preparations

Prior to installation of the High Availability Extension, execute the following preparatory steps:

- Configure hostname resolution and use static host information by editing the `/etc/hosts` file on each server in the cluster. For more information, see the *SUSE Linux Enterprise Server Administration Guide*, available at `http://www.novell.com/documentation`. Refer to chapter *Basic Networking > Configuring Hostname and DNS*.

  It is essential that members of the cluster are able to find each other by name. If the names are not available, internal cluster communication will fail.

- Configure time synchronization by making cluster nodes synchronize to a time server outside the cluster. For more information, see the *SUSE Linux Enterprise Server Administration Guide*, available at `http://www.novell.com/documentation`. Refer to chapter *Time Synchronization with NTP*.

  The cluster nodes will use the time server as their time synchronization source.

2.5 Overview: Installing and Setting Up a Cluster

After the preparations are done, the following basic steps are necessary to install and set up a cluster with SUSE® Linux Enterprise High Availability Extension:

1. Installation of SUSE® Linux Enterprise Server and SUSE® Linux Enterprise High Availability Extension as add-on on top of SUSE Linux Enterprise Server. For detailed information, see Section 3.1, “Installing the High Availability Extension” (page 21).

2. Initial Cluster Setup (page 22)

3. Bringing the Cluster Online (page 30)

4. Configuring global cluster options and adding cluster resources.

Both can be done either with a graphical user interface (GUI) or with command line tools. For detailed information, see Chapter 5, *Configuring and Managing Cluster Resources (GUI)* (page 57) or Chapter 6, *Configuring and Managing Cluster Resources (Command Line)* (page 89).

5. To protect your data from possible corruption by means of fencing and STONITH, make sure to configure STONITH devices as resources. For detailed information, see Chapter 9, *Fencing and STONITH* (page 125).

Depending on your requirements, you may also want to configure the following file system and storage-related components for your cluster:

- Create file systems on a shared disk (Storage Area Network, SAN). If necessary, configure those file systems as cluster resources.

- If you need cluster-aware file systems, use OCFS2.

- To allow the cluster to manage shared storage with Logical Volume Manager, use cLVM, which is a set of clustering extensions to LVM.

- To protect your data integrity, implement storage protection by using fencing mechanisms and by ensuring exclusive storage access.

- If needed, make use of data replication with DRBD.

For detailed information, see Part III, “Storage and Data Replication” (page 153).

Installation and Basic Setup with YaST 3

There are two ways to install the software needed for High Availability clusters: either from a command line, using `zypper`, or with YaST which provides a graphical user interface. After installing the software on all nodes that will be part of your cluster, the next step is to initially configure the cluster so that the nodes can communicate with each other and to start the services needed to bring the cluster online. The initial cluster setup can either be done manually (be editing and copying the configuration files) or with the YaST cluster module.

This chapter describes how to do a new installation and setup with SUSE Linux Enterprise High Availability Extension 11 SP1 from scratch. Refer to chapter Appendix C, *Upgrading Your Cluster to the Latest Product Version* (page 373) if you want to migrate an existing cluster that runs an older version of SUSE Linux Enterprise High Availability Extension or if you want to update any software packages on nodes that are part of a running cluster.

3.1 Installing the High Availability Extension

The packages needed for configuring and managing a cluster with the High Availability Extension are included in the `High Availability` installation pattern. This pattern is only available after SUSE® Linux Enterprise High Availability Extension has been installed as add-on. For information on how to install add-on products, see the SUSE Linux Enterprise 11 SP1 *Deployment Guide*, available at `http://www.novell.com/documentation`. Refer to chapter *Installing Add-On Products*.

NOTE: Installing the Software Packages

The software packages needed for High Availability clusters are not automatically copied to the cluster nodes.

If you do not want to install SUSE® Linux Enterprise Server 11 SP1 and SUSE® Linux Enterprise High Availability Extension 11 SP1 manually on all nodes that will be part of your cluster, use AutoYaST to clone existing nodes. For more information, refer to Section 3.4, “Mass Deployment with AutoYaST” (page 30).

Procedure 3.1 *Installing the High Availability Pattern*

1 Start YaST as `root` user and select *Software* > *Software Management*.

  Alternatively, start the YaST package manager as `root` on a command line with `yast2 sw_single`.

2 From the *Filter* list, select *Patterns* and activate the *High Availability* pattern in the pattern list.

3 Click *Accept* to start the installation of the packages.

3.2 Initial Cluster Setup

After having installed the HA packages, proceed with the initial cluster setup. This includes the following basic steps:

1 Defining the Communication Channels (page 23)

2 Defining Authentication Settings (page 25)

3 Transferring the Configuration to All Nodes (page 26)

The following procedures guide you through each of the steps, using the YaST cluster module. To access the cluster configuration dialog, start YaST as `root` and select *High Availability* > *Cluster*. Alternatively, start the YaST cluster module as `root` on a command line with `yast2 cluster`.

If you start the cluster module for the first time, it appears as wizard, guiding you through all the steps necessary for basic setup. Otherwise, click the categories on the left panel to access the configuration options for each step.

3.2.1 Defining the Communication Channels

For successful communication between the cluster nodes, define at least one communication channel. However, it is recommended to set up the communication via two or more redundant paths (either by using network device bonding or by adding a second communication channel with Corosync). For each communication channel, you need to define the following parameters:

Bind Network Address (`bindnetaddr`)
: The network address to bind to. To ease sharing configuration files across the cluster, OpenAIS uses network interface netmask to mask only the address bits that are used for routing the network. Set the value to the subnet you will use for cluster multicast.

Multicast Address (`mcastaddr`)
: Can be an IPv4 or IPv6 address.

Multicast Port (`mcastport`)
: The UDP port specified for `mcastaddr`.

All nodes in the cluster will know each other from using the same multicast address and the same port number. For different clusters, use a different multicast address.

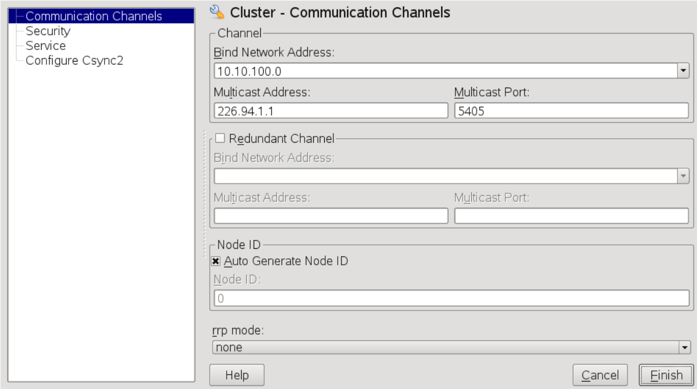
To configure redundant communication with Corosync, you need to define multiple interface sections in `/etc/corosync/corosync.conf`, each with a different ringnumber. Use the Redundant Ring Protocol (RRP) to tell the cluster how to use these interfaces. RRP can have three modes (`rrp_mode`): if set to `active`, Corosync uses all interfaces actively. If set to `passive`, Corosync uses the second interface only if the first ring fails. If rrp_mode is set to `none`, RRP is disabled. With RRP, two physically separate networks are used for communication. In case one network fails, the cluster nodes can still communicate via the other network.

If several rings are configured, each node can have multiple IP addresses. As soon as rrp_mode is enabled, the Stream Control Transmission Protocol (SCTP) is used for communication between the nodes by default (instead of TCP).

Procedure 3.2 *Defining the Communication Channels*

1 In the YaST cluster module, switch to the *Communication Channels* category.

2 Define the *Bind Network Address*, the *Multicast Address* and the *Multicast Port* to use for all cluster nodes.



Cluster - Communication Channels

Channel

Bind Network Address: 10.10.100.0

Multicast Address: 226.94.1.1

Multicast Port: 5405

☐ Redundant Channel

Bind Network Address:

Multicast Address:

Multicast Port:

Node ID

☒ Auto Generate Node ID

Node ID: 0

rrp mode: none

3 If you want to define a second channel:

3a Activate *Redundant Channel*.

3b Define the *Bind Network Address*, the *Multicast Address* and the *Multicast Port* for the redundant channel.

3c Select the *rrp_mode* you want to use. To disable the RRP, select *None*. For more information about the modes, click *Help*.

When using RRP, the primary ring (the first channel you have configured) gets the ringnumber `0`, the second ring (redundant channel) the ringnumber `1` in `/etc/corosync/corosync.conf`.

4 Activate *Auto Generate Node ID* to automatically generate a unique ID for every cluster node.

5 If you only wanted to modify the communication channels for an existing cluster, click *Finish* to write the configuration to `/etc/corosync/corosync.conf` and to close the YaST cluster module. YaST then automatically also adjusts the firewall settings and opens the UDP port used for multicast.

6 For further cluster configuration, click proceed with Procedure 3.3, “Enabling Secure Authentication” (page 25).

3.2.2 Defining Authentication Settings

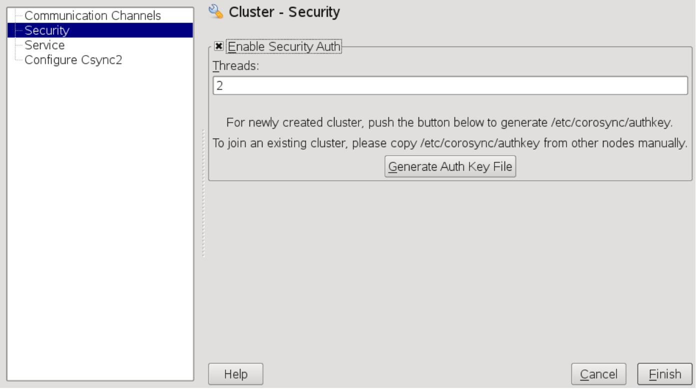
As next step, define the authentication settings for the cluster. You can use HMAC/SHA1 authentication which requires a shared secret, used to protect and authenticate messages. The authentication key (password) you specify will be used on all nodes in the cluster.

Procedure 3.3 *Enabling Secure Authentication*

1 In the YaST cluster module, switch to the *Security* category.

2 Activate *Enabling Security Auth*.

3 For a newly created cluster, click *Generate Auth Key File*. This creates an authentication key that is written to `/etc/corosync/authkey`.



☒ Enable Security Auth

Threads: 2

For newly created cluster, push the button below to generate /etc/corosync/authkey.

To join an existing cluster, please copy /etc/corosync/authkey from other nodes manually.

4 If you only want to modify the authentication settings, click *Finish* to write the configuration to `/etc/corosync/corosync.conf` and to close the YaST cluster module.

5. For further cluster configuration, proceed with Section 3.2.3, “Transferring the Configuration to All Nodes” (page 26).

3.2.3 Transferring the Configuration to All Nodes

Instead of copying the resulting configuration files to all nodes manually, use the `csync2` tool for replication across all nodes in the cluster. Csync2 can handle any number of hosts, sorted into synchronization groups. Each synchronization group has its own list of member hosts and its include/exlude patterns that define which files should be synchronized in the synchronization group. The groups, the hostnames belonging to each group, and the include/exclude rules for each group are specified in the Csync2 configuration file, `/etc/csync2/csync2.cfg`.

For authentication, Csync2 uses the IP addresses and pre-shared-keys within a synchronization group. You need to generate one key file for each synchronization group and copy it to all group members.

For more information about Csync2, refer to `http://oss.linbit.com/csync2/paper.pdf`

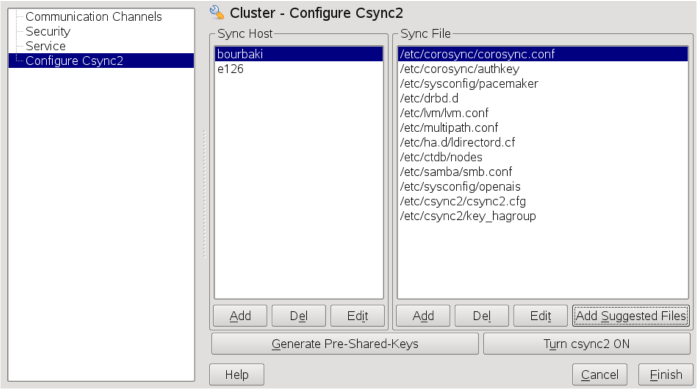
Procedure 3.4 *Configuring Csync2 with YaST*

1. In the YaST cluster module, switch to the *Csync2* category.

2. To specify the synchronization group, click *Add* in the *Sync Host* group and enter the local hostnames of all nodes in your cluster. For each node, you must use exactly the strings that are returned by the `hostname` command.

3. Click *Generate Pre-Shared-Keys* to create a key file for the synchronization group. The key file is written to `/etc/csync2/key_hagroup`. After it has been created, it must be copied manually to all members of the cluster.

4. To populate the *Sync File* list with the files that usually need to be synchronized among all nodes, click *Add Suggested Files*.



Cluster - Configure Csync2

Sync Host

```
bourbaki
e126
```

Sync File

```
/etc/corosync/corosync.conf
/etc/corosync/authkey
/etc/sysconfig/pacemaker
/etc/drbd.d
/etc/lvm/lvm.conf
/etc/multipath.conf
/etc/ha.d/ldirectord.cf
/etc/ctdb/nodes
/etc/samba/smb.conf
/etc/sysconfig/openais
/etc/csync2/csync2.cfg
/etc/csync2/key_hagroup
```

5 If you want to *Edit*, *Add* or *Remove* files from the list of files to be synchronized use the respective buttons. You must enter the absolute pathname for each file.

6 Activate Csync2 by clicking *Turn Csync2 On*. This will start Csync2 automatically at boot time.

7 If all options are set according to your wishes, click *Finish* to close the YaST cluster module. YaST then writes the Csync2 configuration to `/etc/csync2/csync2.cfg`.

After you have configured Csync2, start the synchronization process from command line as described below.

Procedure 3.5 *Synchronizing the Configuration Files with Csync2*

To successfully synchronize the files with Csync2, make sure that the following prerequisites are met:

- The same Csync2 configuration is available on all nodes. Copy the file `/etc/csync2/csync2.cfg` manually to all nodes after you have configured it as described Procedure 3.4, “Configuring Csync2 with YaST” (page 26). Also, it is recommended to include this file in the list of files to be synchronized with Csync2.

- Copy the `/etc/csync2/key_hagroup` file you have generated on one node in Step 3 (page 26) to *all* nodes in the cluster as it is needed for authentication by Csync2. However, do not try to regenerate the file on the other nodes as it needs to be the same file on all nodes.

- Make sure that `xinetd` is running on *all* nodes, as Csync2 depends on that daemon. Start `xinetd` as `root` with the following command:

  ```
  rcxinetd start
  ```

  NOTE: Starting Services at Boot Time

  If you want Csync2 and `xinetd` to start automatically at boot time, execute the following command on all nodes:

  ```
  chkconfig csync2 on
  chkconfig xinetd on
  ```

1. Start the initial file synchronization by executing the following command on *one* of the nodes:

   ```
   csync2 -xv
   ```

   This will synchronize all the files once. If all files can be synchronized successfully, Csync2 will finish with no errors.

   If one or several files that are to be synchronized have been modified also on other nodes (not only on the current one), Csync2 will report a conflict. You will get an output similar to the one below:

   ```
   While syncing file /etc/corosync/corosync.conf:
   ERROR from peer hex-14: File is also marked dirty here!
   Finished with 1 errors.
   ```

2. If you are sure that the file version on the current node is the “best” one, you can resolve the conflict by forcing this file and resynchronizing:

   ```
   csync2 -f /etc/corosync/corosync.conf
   csync2 -x
   ```

For more information on the Csync2 options run `csync2 -help`.

NOTE: Pushing Synchronization After Any Changes

Be aware that Csync2 only pushes changes and it does *not* continuously synchronize files between the nodes.

Each time you have updated any of the files that need to be synchronized, you need to push the changes to the other nodes: Run `csync2 -xv` on the node where you did the changes. If you run the command on any of the other nodes (that have no changes for the synchronized files), nothing will happen.

After you have synchronized the key files to all nodes in the cluster, start the basic services to bring the cluster online as described in Section 3.3, “Bringing the Cluster Online” (page 30).

3.2.4 Starting Services

Optionally, the YaST cluster module lets you define if to start certain services on a node at boot time. You can also use the module to start and stop the services manually (in case you do not want to use the command line for that). In order to bring the cluster nodes online and to start the cluster resource manager, OpenAIS must be started as a service.

Procedure 3.6 *Starting or Stopping Services*

1. In the YaST cluster module, switch to the *Service* category.
2. To start OpenAIS each time this cluster node is booted, select the respective option in the *Booting* group.
3. If you want to use the Pacemaker GUI for configuring, managing and monitoring cluster resources, activate *Start mgmtd as Well*. This daemon is needed for the GUI.
4. To start or stop OpenAIS immediately, click the respective button.
5. Click *Finish* to close the YaST cluster module.

If you selected *Off* in the *Booting* group, you must start OpenAIS manually each time this node is booted. To start OpenAIS manually, use the `rcopenais  start` command.

3.3 Bringing the Cluster Online

After the initial cluster configuration is done, you can now start the service needed to bring the stack online.

Procedure 3.7 *Starting OpenAIS/Corosync and Checking the Status*

1 Run the following command on each of the cluster nodes to start OpenAIS/Corosync:

```
rcopenais start
```

2 On one of the nodes, check the cluster status with the following command:

```
crm_mon
```

If all nodes are online, the output should be similar to the following:

```
============
Last updated: Tue Mar  2 18:35:34 2010
Stack: openais
Current DC: e229 - partition with quorum
Version: 1.1.1-530add2a3721a0ecccb24660a97dbfdaa3e68f51
2 Nodes configured, 2 expected votes
0 Resources configured.
============

 Online: [ e231 e229 ]
```

This output indicates that the cluster resource manager is started and is ready to manage resources.

After the basic configuration is done and the nodes are online, you can now start to configure cluster resources. Use either the `crm` command line tool or the graphical user interface. For more information, refer to Chapter 5, *Configuring and Managing Cluster Resources (GUI)* (page 57) or Chapter 6, *Configuring and Managing Cluster Resources (Command Line)* (page 89).

3.4 Mass Deployment with AutoYaST

AutoYaST is a system for installing one or more SUSE Linux Enterprise systems automatically and without user intervention. SUSE Linux Enterprise lets you create a AutoYaST profile that contains installation and configuration data. The profile tells

AutoYaST what to install and how to configure the installed system to get a completely ready-to-use system in the end. This profile can then be used for mass deployment in different ways.

For detailed instructions of how to make use of AutoYaST in various scenarios, see the SUSE Linux Enterprise 11 SP1 *Deployment Guide*, available at `http://www.novell.com/documentation`. Refer to chapter *Automated Installation*.

Procedure 3.8 *Cloning a Cluster Node with AutoYaST*

The following procedure is suitable for deploying cluster nodes which are clones of an already existing node. The cloned nodes will have the same packages installed and the same system configuration.

If you need to deploy cluster nodes on non-identical hardware, refer to the *Rule-Based Autoinstallation* section in the SUSE Linux Enterprise 11 SP1 *Deployment Guide*, available at `http://www.novell.com/documentation`.

IMPORTANT: Identical Hardware

This scenario assumes you are rolling out SUSE Linux Enterprise High Availability Extension 11 SP1 to a set of machines with exactly the same hardware configuration.

1 Make sure the node you want to clone is correctly installed and configured as described in Section 3.1, "Installing the High Availability Extension" (page 21) and Section 3.2, "Initial Cluster Setup" (page 22).

2 Follow the description outlined in the SUSE Linux Enterprise 11 SP1 *Deployment Guide* for simple mass installation. This includes the following basic steps:

2a Creating an AutoYaST profile. Use the AutoYaST GUI to create and modify a profile from the existing system configuration. In AutoYaST, choose the *High Availability* module and click the *Clone* button. If needed, adjust the configuration in the other modules and save the resulting control file as XML.

2b Determining the source of the AutoYaST profile and the parameter to pass to the installation routines for the other nodes.

2c Determining the source of the SUSE Linux Enterprise Server and SUSE Linux Enterprise High Availability Extension installation data.

2d Determining and setting up the boot scenario for autoinstallation.

2e Passing the command line to the installation routines, either by adding the parameters manually or by creating an `info` file.

2f Starting and monitoring the autoinstallation process.

After the clone has been successfully installed, execute the following steps to make the cloned node join the cluster:

Procedure 3.9 *Bringing the Cloned Node Online*

1. Transfer the key configuration files from the already configured nodes to the cloned node with Csync2 as described in Section 3.2.3, “Transferring the Configuration to All Nodes” (page 26).

2. Start the OpenAIS service on the cloned node as described in Section 3.3, “Bringing the Cluster Online” (page 30) to bring the node online.

The cloned node will now join the cluster because the `/etc/corosync/corosync.conf` file has been applied to the cloned node via Csync2. The CIB is automatically synchronized among the cluster nodes.

Part II. Configuration and Administration

Configuration and Administration Basics 4

The main purpose of an HA cluster is to manage user services. Typical examples of user services are an Apache web server or a database. From the user's point of view, the services do something specific when ordered to do so. To the cluster, however, they are just resources which may be started or stopped—the nature of the service is irrelevant to the cluster.

In this chapter, we will introduce some basic concepts you need to know when configuring resources and administering your cluster. The following chapters show you how to execute the main configuration and administration tasks with each of the management tools the High Availability Extension provides.

4.1 Global Cluster Options

Global cluster options control how the cluster behaves when confronted with certain situations. They are grouped into sets and can be viewed and modified with the cluster management tools like Pacemaker GUI and the `crm` shell. The predefined values can be kept in most cases. However, to make key functions of your cluster work correctly, you need to adjust the following parameters after basic cluster setup:

- Option `no-quorum-policy` (page 36)

- Option `stonith-enabled` (page 37)

Learn how to adjust those parameters with the GUI in Procedure 5.1, "Modifying Global Cluster Options" (page 61). If you prefer the command line approach, see Section 6.2, "Configuring Global Cluster Options" (page 96).

4.1.1 Option `no-quorum-policy`

This global option defines what to do when the cluster does not have quorum (no majority of nodes is part of the partition).

Allowed values are:

`ignore`
: The quorum state does not influence the cluster behavior at all, resource management is continued.

  This setting is useful for the following scenarios:

  - Two-node clusters: Since a single node failure would always result in a loss of majority, usually you want the cluster to carry on regardless. Resource integrity is ensured using fencing, which also prevents split-brain scenarios.

  - Resource-driven clusters: For local clusters with redundant communication channels, a split-brain scenario only has a certain probability. Thus, a loss of communication with a node most likely indicates that the node has crashed, and that the surviving nodes should recover and start serving the resources again.

    If `no-quorum-policy` is set to `ignore`, a 4-node cluster can sustain concurrent failure of three nodes before service is lost, whereas with the other settings, it would lose quorum after concurrent failure of two nodes.

`freeze`
: If quorum is lost, the cluster freezes. Resource management is continued: running resources are not stopped (but possibly restarted in response to monitor events), but no further resources are started within the affected partition.

  This setting is recommended for clusters where certain resources depend on communication with other nodes (for example, OCFS2 mounts). In this case, the default setting `no-quorum-policy=stop` is not useful, as it would lead to the following scenario: Stopping those resources would not be possible while the peer nodes are unreachable. Instead, an attempt to stop them would eventually time out and cause a `stop failure`, triggering escalated recovery and fencing.

`stop` (default value)
: If quorum is lost, all resources in the affected cluster partition are stopped in an orderly fashion.

`suicide`
: Fence all nodes in the affected cluster partition.

4.1.2 Option `stonith-enabled`

This global option defines if to apply fencing, allowing STONITH devices to shoot failed nodes and nodes with resources that cannot be stopped. By default, this global option is set to `true`, because for normal cluster operation it is necessary to use STONITH devices. According to the default value, the cluster will refuse to start any resources if no STONITH resources have been defined.

If you need to disable fencing for any reasons, set `stonith-enabled` to `false`.

For an overview of all global cluster options and their default values, see *Pacemaker 1.0—Configuration Explained*, available from `http://clusterlabs.org/wiki/Documentation`. Refer to section *Available Cluster Options*.

4.2 Cluster Resources

As a cluster administrator, you need to create cluster resources for every resource or application you run on servers in your cluster. Cluster resources can include Web sites, e-mail servers, databases, file systems, virtual machines, and any other server-based applications or services you want to make available to users at all times.

4.2.1 Resource Management

Before you can use a resource in the cluster, it must be set up. For example, if you want to use an Apache server as a cluster resource, set up the Apache server first and complete the Apache configuration before starting the respective resource in your cluster.

If a resource has specific environment requirements, make sure they are present and identical on all cluster nodes. This kind of configuration is not managed by the High Availability Extension. You must do this yourself.

NOTE: Do Not Touch Services Managed by the Cluster

When managing a resource with the High Availability Extension, the same resource must not be started or stopped otherwise (outside of the cluster, for example manually or on boot or reboot). The High Availability Extension software is responsible for all service start or stop actions.

However, if you want to check if the service is configured properly, start it manually, but make sure that it is stopped again before High Availability takes over.

After having configured the resources in the cluster, use the cluster management tools to start, stop, clean up, remove or migrate any resources manually. For details how to do so, refer to Chapter 5, *Configuring and Managing Cluster Resources (GUI)* (page 57) or to Chapter 6, *Configuring and Managing Cluster Resources (Command Line)* (page 89).

4.2.2 Supported Resource Agent Classes

For each cluster resource you add, you need to define the standard that the resource agent conforms to. Resource agents abstract the services they provide and present an accurate status to the cluster, which allows the cluster to be non-committal about the resources it manages. The cluster relies on the resource agent to react appropriately when given a start, stop or monitor command.

Typically, resource agents come in the form of shell scripts. The High Availability Extension supports the following classes of resource agents:

Legacy Heartbeat 1 Resource Agents

Heartbeat version 1 came with its own style of resource agents. As many people have written their own agents based on its conventions, these resource agents are still supported. However, it is recommended to migrate your configurations to High Availability OCF RAs if possible.

Linux Standards Base (LSB) Scripts

LSB resource agents are generally provided by the operating system/distribution and are found in /etc/init.d. To be used with the cluster, they must conform to the LSB init script specification. For example, they must have several actions implemented, which are, at minimum, start, stop, restart, reload, force-reload, and status. For more information, see http://ldn.linuxfoundation.org/lsb/lsb4-resource-page%23Specification.

The configuration of those services is not standardized. If you intend to use an LSB script with High Availability, make sure that you understand how the relevant script is configured. Often you can find information about this in the documentation of the relevant package in /usr/share/doc/packages/*PACKAGENAME*.

Open Cluster Framework (OCF) Resource Agents

OCF RA agents are best suited for use with High Availability, especially when you need master resources or special monitoring abilities. The agents are generally located in /usr/lib/ocf/resource.d/*provider*/. Their functionality is similar to that of LSB scripts. However, the configuration is always done with environmental variables which allow them to accept and process parameters easily. The OCF specification (as it relates to resource agents) can be found at http://www.opencf.org/cgi-bin/viewcvs.cgi/specs/ra/resource-agent-api.txt?rev=HEAD&content-type=text/vnd.viewcvs-markup. OCF specifications have strict definitions of which exit codes must be returned by actions, see Section 8.3, “OCF Return Codes and Failure Recovery” (page 121). The cluster follows these specifications exactly. For a detailed list of all available OCF RAs, refer to Chapter 19, *HA OCF Agents* (page 269).

All OCF Resource Agents are required to have at least the actions start, stop, status, monitor, and meta-data. The meta-data action retrieves information about how to configure the agent. For example, if you want to know more about the IPaddr agent by the provider heartbeat, use the following command:

```
OCF_ROOT=/usr/lib/ocf /usr/lib/ocf/resource.d/heartbeat/IPaddr meta-data
```

The output is information in XML format, including several sections (general description, available parameters, available actions for the agent).

STONITH Resource Agents

This class is used exclusively for fencing related resources. For more information, see Chapter 9, *Fencing and STONITH* (page 125).

The agents supplied with the High Availability Extension are written to OCF specifications.

4.2.3 Types of Resources

The following types of resources can be created:

Primitives

A primitive resource, the most basic type of a resource.

Learn how to create primitive resources with the GUI in Procedure 5.2, “Adding Primitive Resources” (page 62). If you prefer the command line approach, see Section 6.3.1, “Creating Cluster Resources” (page 97).

Groups

Groups contain a set of resources that need to be located together, started sequentially and stopped in the reverse order. For more information, refer to Section “Groups” (page 41).

Clones

Clones are resources that can be active on multiple hosts. Any resource can be cloned, provided the respective resource agent supports it. For more information, refer to Section “Clones” (page 42).

Masters

Masters are a special type of clone resources, they can have multiple modes. For more information, refer to Section “Masters” (page 43).

4.2.4 Advanced Resource Types

Whereas primitives are the simplest kind of resources and therefore easy to configure, you will probably also need more advanced resource types for cluster configuration, such as groups, clones or masters.

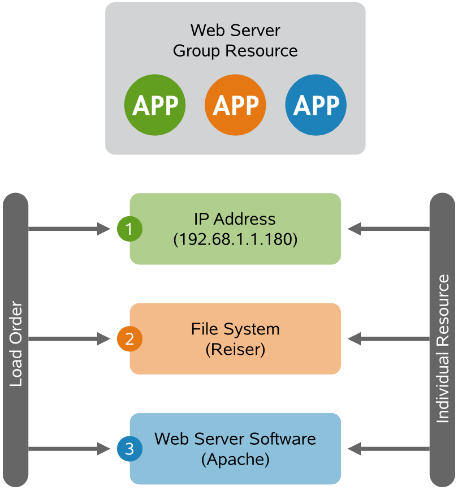
Groups

Some cluster resources are dependent on other components or resources, and require that each component or resource starts in a specific order and runs together on the same server. To simplify this configuration, you can use groups.

Example 4.1 *Resource Group for a Web Server*

An example of a resource group would be a Web server that requires an IP address and a file system. In this case, each component is a separate cluster resource that is combined into a cluster resource group. The resource group would then run on a server or servers, and in case of a software or hardware malfunction, fail over to another server in the cluster the same as an individual cluster resource.

Figure 4.1 *Group Resource*



Groups have the following properties:

Starting and Stopping
: Resources are started in the order they appear in and stopped in the reverse order.

Dependency
: If a resource in the group cannot run anywhere, then none of the resources located after that resource in the group is allowed to run.

Contents
: Groups may only contain a collection of primitive cluster resources. Groups must contain at least one resource, otherwise the configuration is not valid. To refer to the child of a group resource, use the child's ID instead of the group's ID.

Constraints
: Although it is possible to reference the group's children in constraints, it is usually preferable to use the group's name instead.

Stickiness
: Stickiness is additive in groups. Every *active* member of the group will contribute its stickiness value to the group's total. So if the default `resource-stickiness` is `100` and a group has seven members (five of which are active), then the group as a whole will prefer its current location with a score of `500`.

Resource Monitoring
: To enable resource monitoring for a group, you must configure monitoring separately for each resource in the group that you want monitored.

Learn how to create groups with the GUI in Procedure 5.12, "Adding a Resource Group" (page 78). If you prefer the command line approach, see Section 6.3.9, "Configuring a Cluster Resource Group" (page 108).

Clones

You may want certain resources to run simultaneously on multiple nodes in your cluster. To do this you must configure a resource as a clone. Examples of resources that might be configured as clones include STONITH and cluster file systems like OCFS2. You can clone any resource provided. This is supported by the resource's Resource Agent. Clone resources may even be configured differently depending on which nodes they are hosted.

There are three types of resource clones:

Anonymous Clones
: These are the simplest type of clones. They behave identically anywhere they are running. Because of this, there can only be one instance of an anonymous clone active per machine.

Globally Unique Clones

These resources are distinct entities. An instance of the clone running on one node is not equivalent to another instance on another node; nor would any two instances on the same node be equivalent.

Stateful Clones

Active instances of these resources are divided into two states, active and passive. These are also sometimes referred to as primary and secondary, or master and slave. Stateful clones can be either anonymous or globally unique. See also Section "Masters" (page 43).

Clones must contain exactly one group or one regular resource.

When configuring resource monitoring or constraints, masters have different requirements than simple resources. For details, see *Pacemaker 1.0—Configuration Explained*, available from `http://clusterlabs.org/wiki/Documentation`. Refer to section *Clones - Resources That Should be Active on Multiple Hosts*.

Learn how to create clones with the GUI in Procedure 5.14, "Adding or Modifying Clones" (page 82). If you prefer the command line approach, see Section 6.3.10, "Configuring a Clone Resource" (page 108).

Masters

Masters are a specialization of clones that allow the instances to be in one of two operating modes (`master` or `slave`). Masters must contain exactly one group or one regular resource.

When configuring resource monitoring or constraints, masters have different requirements than simple resources. For details, see *Pacemaker 1.0—Configuration Explained*, available from `http://clusterlabs.org/wiki/Documentation`. Refer to section *Multi-state - Resources That Have Multiple Modes*.

4.2.5 Resource Options (Meta Attributes)

For each resource you add, you can define options. Options are used by the cluster to decide how your resource should behave—they tell the CRM how to treat a specific resource. Resource options can be set with the `crm_resource --meta` command or with the GUI as described in Procedure 5.3, “Adding or Modifying Meta and Instance Attributes” (page 64).

Table 4.1 *Options for a Primitive Resource*

Option	Description
`priority`	If not all resources can be active, the cluster will stop lower priority resources in order to keep higher priority ones active.
`target-role`	In what state should the cluster attempt to keep this resource? Allowed values: `stopped`, `started`.
`is-managed`	Is the cluster allowed to start and stop the resource? Allowed values: `true`, `false`.
`resource-stickiness`	How much does the resource prefer to stay where it is? Defaults to the value of `default-resource-stickiness`.
`migration-threshold`	How many failures should occur for this resource on a node before making the node ineligible to host this resource? Default: `none`.
`multiple-active`	What should the cluster do if it ever finds the resource active on more than one node? Allowed values: `block` (mark the resource as unmanaged), `stop_only`, `stop_start`.
`failure-timeout`	How many seconds to wait before acting as if the failure had not occurred (and potentially allowing the resource back to the node on which it failed)? Default: `never`.
`allow-migrate`	Allow resource migration for resources which support `migrate_to`/`migrate_from` actions.

4.2.6 Instance Attributes

The scripts of all resource classes can be given parameters which determine how they behave and which instance of a service they control. If your resource agent supports parameters, you can add them with the `crm_resource` command or with the GUI as described in Procedure 5.3, “Adding or Modifying Meta and Instance Attributes” (page 64). In the `crm` command line utility, instance attributes are called `params`. The list of instance attributes supported by an OCF script can be found by executing the following command as `root`:

```
crm ra info [class:[provider:]]resource_agent
```

or, even shorter:

```
crm ra info resource_agent
```

The output lists all the supported attributes, their purpose and default values.

For example, the command

```
crm ra info Ipaddr
```

returns the following output:

```
Manages virtual IPv4 addresses (portable version) (ocf:heartbeat:IPaddr)

This script manages IP alias IP addresses
It can add an IP alias, or remove one.

Parameters (* denotes required, [] the default):

ip* (string): IPv4 address
The IPv4 address to be configured in dotted quad notation, for example
"192.168.1.1".

nic (string, [eth0]): Network interface
The base network interface on which the IP address will be brought
online.

If left empty, the script will try and determine this from the
routing table.

Do NOT specify an alias interface in the form eth0:1 or anything here;
rather, specify the base interface only.

cidr_netmask (string): Netmask
The netmask for the interface in CIDR format. (ie, 24), or in
dotted quad notation  255.255.255.0).
```

```
If unspecified, the script will also try to determine this from the
routing table.

broadcast (string): Broadcast address
Broadcast address associated with the IP. If left empty, the script will
determine this from the netmask.

iflabel (string): Interface label
You can specify an additional label for your IP address here.

lvs_support (boolean, [false]): Enable support for LVS DR
Enable support for LVS Direct Routing configurations. In case a IP
address is stopped, only move it to the loopback device to allow the
local node to continue to service requests, but no longer advertise it
on the network.

local_stop_script (string):
Script called when the IP is released

local_start_script (string):
Script called when the IP is added

ARP_INTERVAL_MS (integer, [500]): milliseconds between gratuitous ARPs
milliseconds between ARPs

ARP_REPEAT (integer, [10]): repeat count
How many gratuitous ARPs to send out when bringing up a new address

ARP_BACKGROUND (boolean, [yes]): run in background
run in background (no longer any reason to do this)

ARP_NETMASK (string, [ffffffffffff]): netmask for ARP
netmask for ARP - in nonstandard hexadecimal format.

Operations' defaults (advisory minimum):

start         timeout=90
stop          timeout=100
monitor_0     interval=5s timeout=20s
```

NOTE: Instance Attributes for Groups, Clones or Masters

Note that groups, clones and masters do not have instance attributes. However, any instance attributes set will be inherited by the group's, clone's or master's children.

4.2.7 Resource Operations

By default, the cluster will not ensure that your resources are still healthy. To instruct the cluster to do this, you need to add a monitor operation to the resource's definition. Monitor operations can be added for all classes or resource agents. For more information, refer to Section 4.3, "Resource Monitoring" (page 48).

***Table 4.2** Resource Operations*

Operation	Description
`id`	Your name for the action. Must be unique. (The ID is not shown).
`name`	The action to perform. Common values: `monitor`, `start`, `stop`.
`interval`	How frequently to perform the operation. Unit: seconds
`timeout`	How long to wait before declaring the action has failed.
`requires`	What conditions need to be satisfied before this action occurs. Allowed values: `nothing`, `quorum`, `fencing`. The default depends on whether fencing is enabled and if the resource's class is `stonith`. For STONITH resources, the default is `nothing`.
`on-fail`	The action to take if this action ever fails. Allowed values: • `ignore`: Pretend the resource did not fail. • `block`: Do not perform any further operations on the resource. • `stop`: Stop the resource and do not start it elsewhere. • `restart`: Stop the resource and start it again (possibly on a different node).

Operation	Description
	• fence: Bring down the node on which the resource failed (STONITH). • standby: Move *all* resources away from the node on which the resource failed.
enabled	If false, the operation is treated as if it does not exist. Allowed values: true, false.
role	Run the operation only if the resource has this role.
record-pending	Can be set either globally or for individual resources. Makes the CIB reflect the state of "in-flight" operations on resources.
description	Description of the operation.

4.3 Resource Monitoring

If you want to ensure that a resource is running, you must configure resource monitoring for it.

If the resource monitor detects a failure, the following takes place:

- Log file messages are generated, according to the configuration specified in the logging section of /etc/corosync/corosync.conf. By default, the logs are written to syslog, usually /var/log/messages.
- The failure is reflected in the cluster management tools (Pacemaker GUI, HA Web Konsole crm_mon), and in the CIB status section.
- The cluster initiates noticeable recovery actions which may include stopping the resource to repair the failed state and restarting the resource locally or on another node. The resource also may not be restarted at all, depending on the configuration and state of the cluster.

If you do not configure resource monitoring, resource failures after a successful start will not be communicated, and the cluster will always show the resource as healthy.

Learn how to add monitor operations to resources with the GUI in Procedure 5.11, “Adding or Modifying Monitor Operations” (page 76). If you prefer the command line approach, see Section 6.3.8, “Configuring Resource Monitoring” (page 107).

4.4 Resource Constraints

Having all the resources configured is only part of the job. Even if the cluster knows all needed resources, it might still not be able to handle them correctly. Resource constraints let you specify which cluster nodes resources can run on, what order resources will load, and what other resources a specific resource is dependent on.

4.4.1 Types of Constraints

There are three different kinds of constraints available:

Resource Location
: Locational constraints that define on which nodes a resource may be run, may not be run or is preferred to be run.

Resource Collocation
: Collocational constraints that tell the cluster which resources may or may not run together on a node.

Resource Order
: Ordering constraints to define the sequence of actions.

For more information on configuring constraints and detailed background information about the basic concepts of ordering and collocation, refer to the following documents available at `http://clusterlabs.org/wiki/Documentation`:

- *Pacemaker 1.0—Configuration Explained* , chapter *Resource Constraints*

- *Collocation Explained*

- *Ordering Explained*

Learn how to add the various kinds of constraints with the GUI in Section 5.3.3, “Configuring Resource Constraints” (page 66). If you prefer the command line approach, see Section 6.3.4, “Configuring Resource Constraints” (page 102).

4.4.2 Scores and Infinity

When defining constraints, you also need to deal with scores. Scores of all kinds are integral to how the cluster works. Practically everything from migrating a resource to deciding which resource to stop in a degraded cluster is achieved by manipulating scores in some way. Scores are calculated on a per-resource basis and any node with a negative score for a resource cannot run that resource. After calculating the scores for a resource, the cluster then chooses the node with the highest score.

`INFINITY` is currently defined as `1,000,000`. Additions or subtractions with it stick to the following three basic rules:

- Any value + INFINITY = INFINITY
- Any value - INFINITY = -INFINITY
- INFINITY - INFINITY = -INFINITY

When defining resource constraints, you specify a score for each constraint. The score indicates the value you are assigning to this resource constraint. Constraints with higher scores are applied before those with lower scores. By creating additional location constraints with different scores for a given resource, you can specify an order for the nodes that a resource will fail over to.

4.4.3 Failover Nodes

A resource will be automatically restarted if it fails. If that cannot be achieved on the current node, or it fails `N` times on the current node, it will try to fail over to another node. Each time the resource fails, its failcount is raised. You can define a number of failures for resources (a `migration-threshold`), after which they will migrate to a new node. If you have more than two nodes in your cluster, the node a particular resource fails over to is chosen by the High Availability software.

However, you can specify the node a resource will fail over to by configuring one or several location constraints and a `migration-threshold` for that resource. For detailed instructions how to achieve this with the GUI, refer to Section 5.3.4, "Specifying Resource Failover Nodes" (page 70). If you prefer the command line approach, see Section 6.3.5, "Specifying Resource Failover Nodes" (page 104).

Example 4.2 *Migration Threshold—Process Flow*

For example, let us assume you have configured a location constraint for resource `r1` to preferably run on `node1`. If it fails there, `migration-threshold` is checked and compared to the failcount. If failcount >= migration-threshold then the resource is migrated to the node with the next best preference.

By default, once the threshold has been reached, the node will no longer be allowed to run the failed resource until the resource's failcount is reset. This can be done manually by the cluster administrator or by setting a `failure-timeout` option for the resource.

For example, a setting of `migration-threshold=2` and `failure-timeout=60s` would cause the resource to migrate to a new node after two failures and potentially allow it to move back (depending on the stickiness and constraint scores) after one minute.

There are two exceptions to the migration threshold concept, occurring when a resource either fails to start or fails to stop:

- Start failures set the failcount to `INFINITY` and thus always cause an immediate migration.

- Stop failures cause fencing (when `stonith-enabled` is set to `true` which is the default).

  In case there is no STONITH resource defined (or `stonith-enabled` is set to `false`), the resource will not migrate at all.

For details on using migration thresholds and resetting failcounts, refer to Section 5.3.4, "Specifying Resource Failover Nodes" (page 70). If you prefer the command line approach, see Section 6.3.5, "Specifying Resource Failover Nodes" (page 104).

4.4.4 Failback Nodes

A resource might fail back to its original node when that node is back online and in the cluster. If you want to prevent a resource from failing back to the node it was running on prior to failover, or if you want to specify a different node for the resource to fail back to, you must change its `resource stickiness` value. You can either specify resource stickiness when you are creating a resource, or afterwards.

Consider the following implications when specifying resource stickiness values:

Value is `0`:
: This is the default. The resource will be placed optimally in the system. This may mean that it is moved when a “better” or less loaded node becomes available. This option is almost equivalent to automatic failback, except that the resource may be moved to a node that is not the one it was previously active on.

Value is greater than `0`:
: The resource will prefer to remain in its current location, but may be moved if a more suitable node is available. Higher values indicate a stronger preference for a resource to stay where it is.

Value is less than `0`:
: The resource prefers to move away from its current location. Higher absolute values indicate a stronger preference for a resource to be moved.

Value is `INFINITY`:
: The resource will always remain in its current location unless forced off because the node is no longer eligible to run the resource (node shutdown, node standby, reaching the `migration-threshold`, or configuration change). This option is almost equivalent to completely disabling automatic failback.

Value is `-INFINITY`:
: The resource will always move away from its current location.

4.4.5 Placing Resources Based on Their Load Impact

Not all resources are equal. Some, such as Xen guests, require that the node hosting them meets their capacity requirements. If resources are placed such that their combined need exceed the provided capacity, the resources diminish in performance (or even fail).

To take this into account, the High Availability Extension allows you to specify the following parameters:

1. The capacity a certain node *provides*.

2. The capacity a certain resource *requires*.

3. An overall strategy for placement of resources.

Currently, these settings are static and must be configured by the administrator—they are not dynamically discovered or adjusted.

Learn how to configure these settings with the GUI in Section 5.3.6, “Configuring Placement of Resources Based on Load Impact” (page 72). If you prefer the command line approach, see Section 6.3.7, “Configuring Placement of Resources Based on Load Impact” (page 105).

A node is considered eligible for a resource if it has sufficient free capacity to satisfy the resource's requirements. The nature of the required or provided capacities is completely irrelevant for the High Availability Extension, it just makes sure that all capacity requirements of a resource are satisfied before moving a resource to a node.

To configure the resource's requirements and the capacity a node provides, use utilization attributes. You can name the utilization attributes according to your preferences and define as many name/value pairs as your configuration needs. However, the attribute's values must be integers.

The placement strategy can be specified with the `placement-strategy` property (in the global cluster options). The following values are available:

`default` (default value)
: Utilization values are not considered at all. Resources are allocated according to location scoring. If scores are equal, resources are evenly distributed across nodes.

`utilization`
: Utilization values are considered when deciding if a node has enough free capacity to satisfy a resources's requirements. However, load-balancing is still done based on the number of resources allocated to a node.

`minimal`
: Utilization values are considered when deciding if a node has enough free capacity to satisfy a resource's requirements. An attempt is made to concentrate the resources on as few nodes as possible (in order to achieve power savings on the remaining nodes).

`balanced`
: Utilization values are considered when deciding if a node has enough free capacity to satisfy a resource's requirements. An attempt is made to distribute the resources evenly, thus optimizing resource performance.

NOTE: Configuring Resource Priorities

The available placement strategies are best-effort—they do not yet use complex heuristic solvers to always reach optimum allocation results. Thus, set your resource priorities in a way that makes sure that your most important resources are scheduled first.

Example 4.3 *Example Configuration for Load-Balanced Placing*

The following example demonstrates a three-node cluster of equal nodes, with four virtual machines.

```
node node1 utilization memory="4000"
node node2 utilization memory="4000"
node node3 utilization memory="4000"
primitive xenA ocf:heartbeat:Xen utilization memory="3500" \
     meta priority="10"
primitive xenB ocf:heartbeat:Xen utilization memory="2000" \
     meta priority="1"
primitive xenC ocf:heartbeat:Xen utilization memory="2000" \
     meta priority="1"
primitive xenD ocf:heartbeat:Xen utilization memory="1000" \
     meta priority="5"
property placement-strategy="minimal"
```

With all three nodes up, resource `xenA` will be placed onto a node first, followed by `xenD`. `xenB` and `xenC` would either be allocated together or one of them with `xenD`.

If one node failed, too little total memory would be available to host them all. `xenA` would be ensured to be allocated, as would `xenD`. However, only one of the remaining resources `xenB` or `xenC` could still be placed. Since their priority is equal, the result would still be open. To resolve this ambiguity as well, you would need to set a higher priority for either one.

4.5 For More Information

`http://clusterlabs.org/`
: Home page of Pacemaker, the cluster resource manager shipped with the High Availability Extension.

`http://linux-ha.org`
: Home page of the The High Availability Linux Project.

`http://clusterlabs.org/wiki/Documentation`
: *CRM Command Line Interface* : Introduction to the `crm` command line tool.

`http://clusterlabs.org/wiki/Documentation`
: *Pacemaker 1.0—Configuration Explained* : Explains the concepts used to configure Pacemaker. Contains comprehensive and very detailed information for reference.

Configuring and Managing Cluster Resources (GUI)

5

To configure and manage cluster resources, either use the graphical user interface (the Pacemaker GUI) or the `crm` command line utility. For the command line approach, refer to Chapter 6, *Configuring and Managing Cluster Resources (Command Line)* (page 89).

This chapter introduces the Pacemaker GUI and covers basic tasks needed when configuring and managing cluster resources: creating basic and advanced types of resources (groups and clones), configuring constraints, specifying failover nodes and failback nodes, configuring resource monitoring, starting, cleaning up or removing resources, and migrating resources manually.

Support for the GUI is provided by two packages: The `pacemaker-mgmt` package contains the back-end for the GUI (the `mgmtd` daemon). It must be installed on all cluster nodes you want to connect to with the GUI. On any machine where you want to run the GUI, install the `pacemaker-mgmt-client` package.

5.1 Pacemaker GUI—Overview

To start the Pacemaker GUI, enter `crm_gui` at the command line. To access the configuration and administration options, you need to log in to a cluster.

5.1.1 Connecting to a Cluster

NOTE: User Authentication

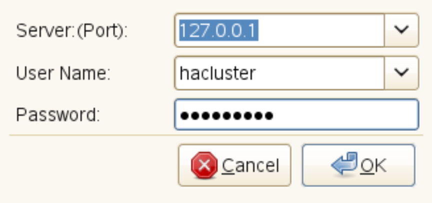
To log in to the cluster from the Pacemaker GUI, the respective user must be a member of the `haclient` group. The installation creates a linux user named `hacluster` which is member of the `haclient` group.

Before using the Pacemaker GUI, either set a password for the `hacluster` user or create a new user which is member of the `haclient` group.

Do this on every node you will connect to with the Pacemaker GUI.

To connect to the cluster, select *Connection* > *Login*. By default, the *Server* field shows the localhost's IP address and `hacluster` as *User Name*. Enter the user's password to continue.

Figure 5.1 *Connecting to the Cluster*



Server:(Port): 127.0.0.1

User Name: hacluster

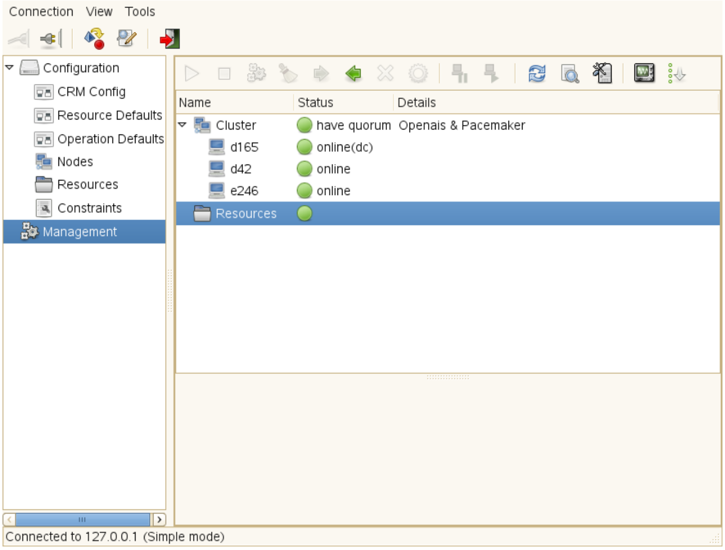
Password: ●●●●●●●●●●

If you are running the Pacemaker GUI remotely, enter the IP address of a cluster node as *Server*. As *User Name*, you can also use any other user belonging to the `haclient` group to connect to the cluster.

5.1.2 Main Window

After being connected, the main window opens:

Figure 5.2 *Pacemaker GUI - Main Window*



Name	Status	Details
Cluster	● have quorum	Openais & Pacemaker
d165	● online(dc)	
d42	● online	
e246	● online	
Resources	●	

Connected to 127.0.0.1 (Simple mode)

To view or modify cluster components like the CRM, resources, nodes or constraints, select the respective subentry of the *Configuration* category in the left pane and use the options that become available in the right pane. Additionally, the Pacemaker GUI lets you easily view, edit, import and export XML fragments of the CIB for the following subitems: *Resource Defaults*, *Operation Defaults*, *Nodes*, *Resources*, and *Constraints*. Select any of the *Configuration* subitems and select *Show* > *XML Mode* in the upper right corner of the window.

If you have already configured your resources, click the *Management* category in the left pane to show the status of your cluster and its resources. This view also allows you to set nodes to `standby` and to modify the management status of nodes (if they are currently managed by the cluster or not). To access the main functions for resources (starting, stopping, cleaning up or migrating resources), select the resource in the right pane and use the icons in the toolbar. Alternatively, right-click the resource and select the respective menu item from the context menu.

The Pacemaker GUI also allows you to switch between different view modes, influencing the behavior of the software and hiding or showing certain aspects:

Simple Mode
: Lets you add resources in a wizard-like mode. When creating and modifying resources, shows the frequently-used tabs for sub-objects, allowing you to directly add objects of that type via the tab.

  Allows you to view and change all available global cluster options by selecting the *CRM Config* entry in the left pane. The right pane then shows the values that are currently set. If no specific value is set for an option, it shows the default values instead.

Expert Mode
: Lets you add resources in either a wizard-like mode or via dialog windows. When creating and modifying resources, only shows the corresponding tab if a particular type of sub-object already exists in CIB. When adding a new sub-object, you will prompted to select the object type, thus allowing you to add all supported types of sub-objects.

  When selecting the *CRM Config* entry in the left pane, only shows the values of global cluster options that have been actually set. Hides all cluster options that will automatically use the defaults (because no values have been set). In this mode, the global cluster options can only be modified by using the individual configuration dialogs.

Hack Mode
: Has the same functions as the expert mode. Allows you to add additional attribute sets that include specific rules to make your configuration more dynamic. For example, you can make a resource have different instance attributes depending on the node it is hosted on. Furthermore, you can add a time-based rule for a meta attribute set to determine when the attributes take effect.

The window's status bar also shows the currently active mode.

The following sections guide you through the main tasks you need to execute when configuring cluster options and resources and show you how to administer the resources with the Pacemaker GUI. Where not stated otherwise, the step-by-step instructions reflect the procedure as executed in the simple mode.

5.2 Configuring Global Cluster Options

Global cluster options control how the cluster behaves when confronted with certain situations. They are grouped into sets and can be viewed and modified with the cluster management tools like Pacemaker GUI and `crm` shell. The predefined values can be kept in most cases. However, to make key functions of your cluster work correctly, you need to adjust the following parameters after basic cluster setup:

- Option `no-quorum-policy` (page 36)

- Option `stonith-enabled` (page 37)

***Procedure 5.1** Modifying Global Cluster Options*

1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Connecting to a Cluster” (page 58).

2 Select *View* > *Simple Mode*.

3 In the left pane, select *CRM Config* to view the global cluster options and their current values.

4 Depending on your cluster requirements, set *No Quorum Policy* to the appropriate value.

5 If you need to disable fencing for any reasons, deselect `Stonith Enabled`.

6 Confirm your changes with *Apply*.

You can at any time switch back to the default values for all options by selecting *CRM Config* in the left pane and clicking *Default*.

5.3 Configuring Cluster Resources

As a cluster administrator, you need to create cluster resources for every resource or application you run on servers in your cluster. Cluster resources can include Web sites, e-mail servers, databases, file systems, virtual machines, and any other server-based applications or services you want to make available to users at all times.

For an overview of resource types you can create, refer to Section 4.2.3, “Types of Resources” (page 40).

5.3.1 Creating Simple Cluster Resources

To create the most basic type of a resource, proceed as follows:

Procedure 5.2 *Adding Primitive Resources*

1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Connecting to a Cluster” (page 58).

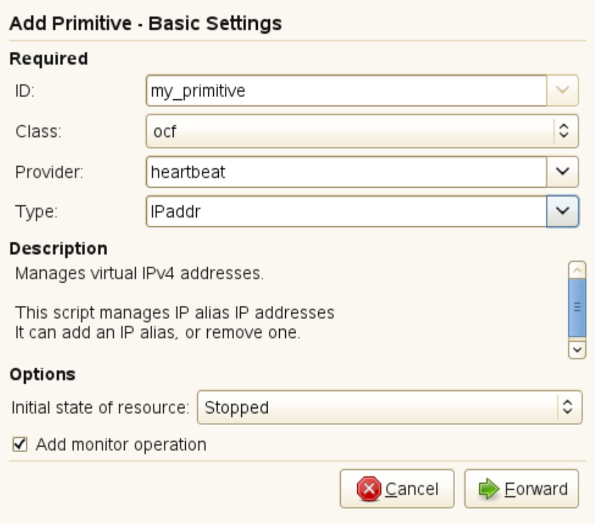
2 In the left pane, select *Resources* and click *Add* > *Primitive*.

3 In the next dialog, set the following parameters for the resource:

> **3a** Enter a unique *ID* for the resource.
>
> **3b** From the *Class* list, select the resource agent class you want to use for that resource: *heartbeat*, *lsb*, *ocf* or *stonith*. For more information, see Section 4.2.2, “Supported Resource Agent Classes” (page 38).
>
> **3c** If you selected *ocf* as class, specify also the *Provider* of your OCF resource agent. The OCF specification allows multiple vendors to supply the same resource agent.
>
> **3d** From the *Type* list, select the resource agent you want to use (for example, *IPaddr* or *Filesystem*). A short description for this resource agent is displayed below.
>
> The selection you get in the *Type* list depends on the *Class* (and for OCF resources also on the *Provider*) you have chosen.

3e Below *Options*, set the *Initial state of resource*.

3f Activate *Add monitor operation* if you want the cluster to monitor if the resource is still healthy.



Add Primitive - Basic Settings

Required

ID: my_primitive

Class: ocf

Provider: heartbeat

Type: IPaddr

Description

Manages virtual IPv4 addresses.

This script manages IP alias IP addresses
It can add an IP alias, or remove one.

Options

Initial state of resource: Stopped

☑ Add monitor operation

4 Click *Forward*. The next window shows a summary of the parameters that you have already defined for that resource. All required *Instance Attributes* for that resource are listed. You need to edit them in order to set them to appropriate values. You may also need to add more attributes, depending on your deployment and settings. For details how to do so, refer to Procedure 5.3, “Adding or Modifying Meta and Instance Attributes” (page 64).

5 If all parameters are set according to your wishes, click *Apply* to finish the configuration of that resource. The configuration dialog is closed and the main window shows the newly added resource.

During or after creation of a resource, you can add or modify the following parameters for resources:

- `Instance attributes`—they determine which instance of a service the resource controls. For more information, refer to Section 4.2.6, “Instance Attributes” (page 45).

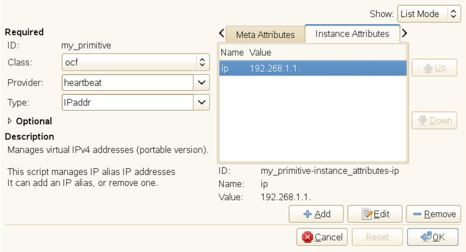
- `Meta attributes`—they tell the CRM how to treat a specific resource. For more information, refer to Section 4.2.5, “Resource Options (Meta Attributes)” (page 43).

- `Operations`—they are needed for resource monitoring. For more information, refer to Section 4.2.7, “Resource Operations” (page 47).

Procedure 5.3 *Adding or Modifying Meta and Instance Attributes*

1. In the Pacemaker GUI main window, click *Resources* in the left pane to see the resources already configured for the cluster.

2. In the right pane, select the resource to modify and click *Edit* (or double-click the resource). The next window shows the basic resource parameters and the *Meta Attributes*, *Instance Attributes* or *Operations* already defined for that resource.



Show: List Mode

Required
ID: my_primitive
Class: ocf
Provider: heartbeat
Type: IPaddr

Optional

Description
Manages virtual IPv4 addresses (portable version).

This script manages IP alias IP addresses
It can add an IP alias, or remove one.

Name	Value
ip	192.268.1.1.

ID: my_primitive-instance_attributes-ip
Name: ip
Value: 192.268.1.1.

3. To add a new meta attribute or instance attribute, select the respective tab and click *Add*.

4. Select the *Name* of the attribute you want to add. A short *Description* is displayed.

5. If needed, specify an attribute *Value*. Otherwise the default value of that attribute will be used.

6. Click *OK* to confirm your changes. The newly added or modified attribute appears on the tab.

7. If all parameters are set according to your wishes, click *OK* to finish the configuration of that resource. The configuration dialog is closed and the main window shows the modified resource.

TIP: XML Source Code for Resources

The Pacemaker GUI allows you to view the XML fragments that are generated from the parameters that you have defined. For individual resources, select *Show* > *XML Mode* in the top right corner of the resource configuration dialog.

To access the XML representation of all resources that you have configured, click *Resources* in the left pane and then select *Show* > *XML Mode* in the upper right corner of the main window.

The editor displaying the XML code allows you to *Import* or *Export* the XML elements or to manually edit the XML code.

5.3.2 Creating STONITH Resources

To configure fencing, you need to configure one or more STONITH resources.

Procedure 5.4 *Adding a STONITH Resource*

1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, "Connecting to a Cluster" (page 58).

2 In the left pane, select *Resources* and click *Add* > *Primitive*.

3 In the next dialog, set the following parameters for the resource:

3a Enter a unique *ID* for the resource.

3b From the *Class* list, select the resource agent class *stonith*.

3c From the *Type* list, select the STONITH plug-in for controlling your STONITH device. A short description for this plug-in is displayed below.

3d Below *Options*, set the *Initial state of resource*.

3e Activate *Add monitor operation* if you want the cluster to monitor the fencing device. For more information, refer to Section 9.4, "Monitoring Fencing Devices" (page 132).

4 Click *Forward*. The next window shows a summary of the parameters that you have already defined for that resource. All required *Instance Attributes* for the selected STONITH plug-in are listed. You need to edit them in order to set them to appropriate values. You may also need to add more attributes or monitor operations, depending on your deployment and settings. For details how to do so, refer to Procedure 5.3, “Adding or Modifying Meta and Instance Attributes” (page 64) and Section 5.3.7, “Configuring Resource Monitoring” (page 76).

5 If all parameters are set according to your wishes, click *Apply* to finish the configuration of that resource. The configuration dialog is closed and the main window shows the newly added resource.

To complete your fencing configuration add constraints, or use clones or both. For more details, refer to Chapter 9, *Fencing and STONITH* (page 125).

5.3.3 Configuring Resource Constraints

Having all the resources configured is only part of the job. Even if the cluster knows all needed resources, it might still not be able to handle them correctly. Resource constraints let you specify which cluster nodes resources can run on, what order resources will load, and what other resources a specific resource is dependent on.

For an overview which types of constraints are available, refer to Section 4.4.1, “Types of Constraints” (page 49). When defining constraints, you also need to specify scores. For more information about scores and their implications in the cluster, see Section 4.4.2, “Scores and Infinity” (page 50).

Learn how to create the different types of the constraints in the following procedures.

Procedure 5.5 *Adding or Modifying Locational Constraints*

1. Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Connecting to a Cluster” (page 58).

2. In the Pacemaker GUI main window, click *Constraints* in the left pane to see the constraints already configured for the cluster.

3. In the left pane, select *Constraints* and click *Add*.

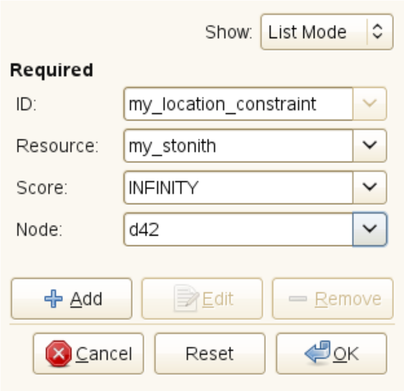
4. Select *Resource Location* and click *OK*.

5. Enter a unique *ID* for the constraint. When modifying existing constraints, the ID is already defined and is displayed in the configuration dialog.

6. Select the *Resource* for which to define the constraint. The list shows the IDs of all resources that have been configured for the cluster.

7. Set the *Score* for the constraint. Positive values indicate the resource can run on the *Node* you specify below. Negative values indicate the resource cannot run on this node. Values of `+/- INFINITY` change “can” to `must`.

8. Select the *Node* for the constraint.

   Show: List Mode

   Required

   ID: my_location_constraint

   Resource: my_stonith

   Score: INFINITY

   Node: d42

9. If you leave the *Node* and the *Score* field empty, you can also add rules by clicking *Add* > *Rule*. To add a lifetime, just click *Add* > *Lifetime*.

10 If all parameters are set according to your wishes, click *OK* to finish the configuration of the constraint. The configuration dialog is closed and the main window shows the newly added or modified constraint.

Procedure 5.6 *Adding or Modifying Collocational Constraints*

1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Connecting to a Cluster” (page 58).

2 In the Pacemaker GUI main window, click *Constraints* in the left pane to see the constraints already configured for the cluster.

3 In the left pane, select *Constraints* and click *Add*.

4 Select *Resource Collocation* and click *OK*.

5 Enter a unique *ID* for the constraint. When modifying existing constraints, the ID is already defined and is displayed in the configuration dialog.

6 Select the *Resource* which is the collocation source. The list shows the IDs of all resources that have been configured for the cluster.

If the constraint cannot be satisfied, the cluster may decide not to allow the resource to run at all.

7 If you leave both the *Resource* and the *With Resource* field empty, you can also add a resource set by clicking *Add* > *Resource Set*. To add a lifetime, just click *Add* > *Lifetime*.

8 In *With Resource*, define the collocation target. The cluster will decide where to put this resource first and then decide where to put the resource in the *Resource* field.

9 Define a *Score* to determine the location relationship between both resources. Positive values indicate the resources should run on the same node. Negative values indicate the resources should not run on the same node. Values of `+/-INFINITY` change `should` to `must`. The score will be combined with other factors to decide where to put the resource.

10 If needed, specify further parameters, like *Resource Role*.

Depending on the parameters and options you choose, a short *Description* explains the effect of the collocational constraint you are configuring.

11 If all parameters are set according to your wishes, click *OK* to finish the configuration of the constraint. The configuration dialog is closed and the main window shows the newly added or modified constraint.

Procedure 5.7 *Adding or Modifying Ordering Constraints*

1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Connecting to a Cluster” (page 58).

2 In the Pacemaker GUI main window, click *Constraints* in the left pane to see the constraints already configured for the cluster.

3 In the left pane, select *Constraints* and click *Add*.

4 Select *Resource Order* and click *OK*.

5 Enter a unique *ID* for the constraint. When modifying existing constraints, the ID is already defined and is displayed in the configuration dialog.

6 With *First*, define the resource that must be started before the resource specified with *Then* is allowed to.

7 With *Then* define the resource that will start after the *First* resource.

Depending on the parameters and options you choose, a short *Description* explains the effect of the ordering constraint you are configuring.

8 If needed, define further parameters, for example:

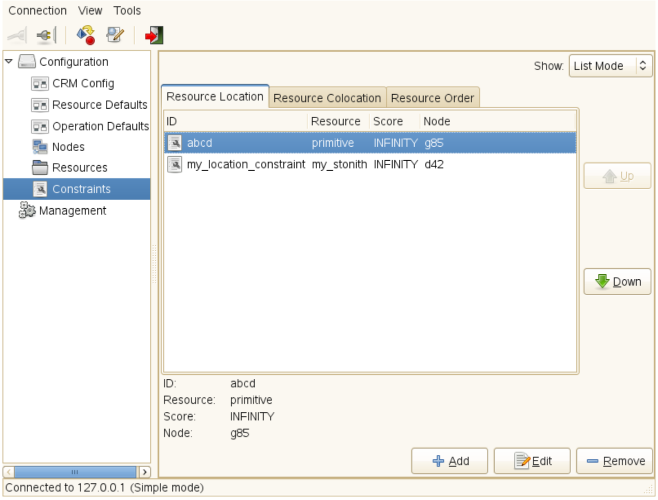
8a Specify a *Score*. If greater than zero, the constraint is mandatory, otherwise it is only a suggestion. The default value is `INFINITY`.

8b Specify a value for *Symmetrical*. If `true`, the resources are stopped in the reverse order. The default value is `true`.

9 If all parameters are set according to your wishes, click *OK* to finish the configuration of the constraint. The configuration dialog is closed and the main window shows the newly added or modified constraint.

You can access and modify all constraints that you have configured in the *Constraints* view of the Pacemaker GUI.

***Figure 5.3** Pacemaker GUI - Constraints*



ID	Resource	Score	Node
abcd	primitive	INFINITY	g85
my_location_constraint	my_stonith	INFINITY	d42

ID: abcd
Resource: primitive
Score: INFINITY
Node: g85

Connected to 127.0.0.1 (Simple mode)

5.3.4 Specifying Resource Failover Nodes

A resource will be automatically restarted if it fails. If that cannot be achieved on the current node, or it fails N times on the current node, it will try to fail over to another node. You can define a number of failures for resources (a `migration-threshold`), after which they will migrate to a new node. If you have more than two nodes in your cluster, the node a particular resource fails over to is chosen by the High Availability software.

However, you can specify the node a resource will fail over to by proceeding as follows:

1 Configure a location constraint for that resource as described in Procedure 5.5, “Adding or Modifying Locational Constraints” (page 67).

2 Add the `migration-threshold` meta attribute to that resource as described in Procedure 5.3, “Adding or Modifying Meta and Instance Attributes” (page 64) and enter a *Value* for the migration-threshold. The value should be positive and less that INFINITY.

3 If you want to automatically expire the failcount for a resource, add the `failure-timeout` meta attribute to that resource as described in Procedure 5.3, “Adding or Modifying Meta and Instance Attributes” (page 64) and enter a *Value* for the failure-timeout.

4 If you want to specify additional failover nodes with preferences for a resource, create additional location constraints.

For an example of the process flow in the cluster regarding migration thresholds and failcounts, see Example 4.2, “Migration Threshold—Process Flow” (page 51).

Instead of letting the failcount for a resource expire automatically, you can also clean up failcounts for a resource manually at any time. Refer to Section 5.4.2, “Cleaning Up Resources” (page 84) for the details.

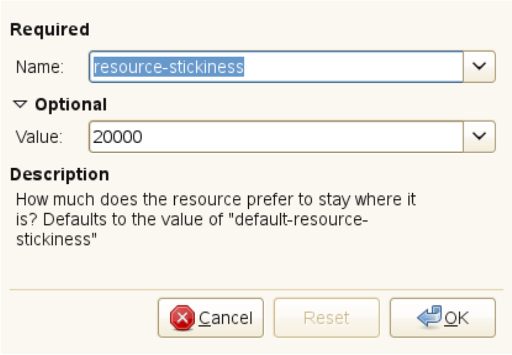
5.3.5 Specifying Resource Failback Nodes (Resource Stickiness)

A resource might fail back to its original node when that node is back online and in the cluster. If you want to prevent a resource from failing back to the node it was running on prior to failover, or if you want to specify a different node for the resource to fail back to, you must change its resource stickiness value. You can either specify resource stickiness when you are creating a resource, or afterwards.

For the implications of different resource stickiness values, refer to Section 4.4.4, “Failback Nodes” (page 52).

Procedure 5.8 *Specifying Resource Stickiness*

1 Add the `resource-stickiness` meta attribute to the resource as described in Procedure 5.3, “Adding or Modifying Meta and Instance Attributes” (page 64).



Required

Name: resource-stickiness

Optional

Value: 20000

Description

How much does the resource prefer to stay where it is? Defaults to the value of "default-resource-stickiness"

2 As *Value* for the resource-stickiness, specify a value between `-INFINITY` and `INFINITY`.

5.3.6 Configuring Placement of Resources Based on Load Impact

Not all resources are equal. Some, such as Xen guests, require that the node hosting them meets their capacity requirements. If resources are placed such that their combined need exceed the provided capacity, the resources diminish in performance (or even fail).

To take this into account, the High Availability Extension allows you to specify the following parameters:

1. The capacity a certain node *provides*.

2. The capacity a certain resource *requires*.

3. An overall strategy for placement of resources.

For detailed background information about the parameters and a configuration example, refer to Section 4.4.5, "Placing Resources Based on Their Load Impact" (page 53).

To configure the resource's requirements and the capacity a node provides, use utilization attributes as described in Procedure 5.9, "Adding Or Modifying Utilization Attributes" (page 73). You can name the utilization attributes according to your preferences and define as many name/value pairs as your configuration needs.

Procedure 5.9 *Adding Or Modifying Utilization Attributes*

In the following example, we assume that you already have a basic configuration of cluster nodes and resources and now additionally want to configure the capacities a certain node provides and the capacity a certain resource requires. The procedure of adding utilization attributes is basically the same and only differs in Step 2 (page 73) and Step 3 (page 73).

1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, "Connecting to a Cluster" (page 58).

2 To specify the capacity a node *provides*:

  2a In the left pane, click *Node*.

  2b In the right pane, select the node whose capacity you want to configure and click *Edit*.

3 To specify the capacity a resource *requires*:

  3a In the left pane, click *Resources*.

  3b In the right pane, select the resource whose capacity you want to configure and click *Edit*.

4 Select the *Utilization* tab and click *Add* to add an utilization attribute.

5 Enter a *Name* for the new attribute. You can name the utilization attributes according to your preferences.

6 Enter a *Value* for the attribute and click *OK*. The attribute value must be an integer.

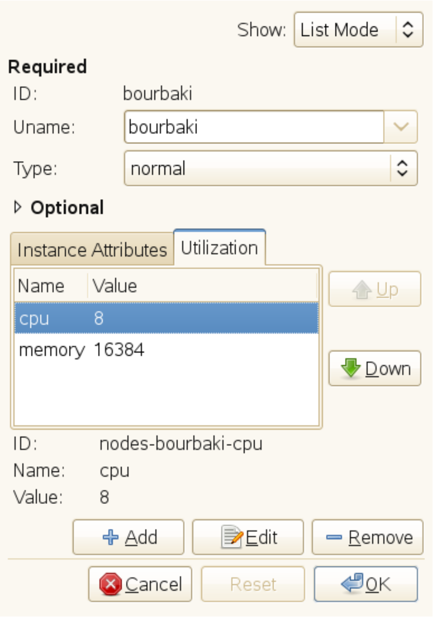
7 If you need more utilization attributes, repeat Step 5 (page 73) to Step 6 (page 73).

The *Utilization* tab shows a summary of the utilization attributes that you have already defined for that node or resource.

8 If all parameters are set according to your wishes, click *OK* to close the configuration dialog.

Figure 5.4, “Example Configuration for Node Capacity” (page 74) shows the configuration of a node which would provide 8 CPU units and 16 GB of memory to resources running on that node:

Figure 5.4 *Example Configuration for Node Capacity*



Show: List Mode

Required

ID: bourbaki

Uname: bourbaki

Type: normal

Optional

Instance Attributes | Utilization

Name	Value
cpu	8
memory	16384

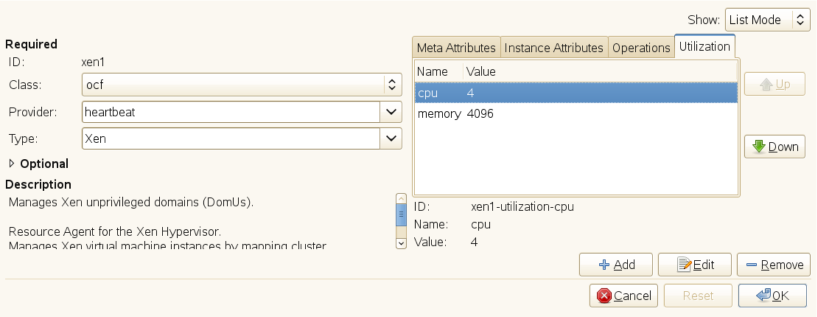
ID: nodes-bourbaki-cpu

Name: cpu

Value: 8

An example configuration for a resource requiring 4096 memory units and 4 of the CPU units of a node would look as follows:

Figure 5.5 *Example Configuration for Resource Capacity*



Show: List Mode

Required

ID: xen1

Class: ocf

Provider: heartbeat

Type: Xen

Optional

Description

Manages Xen unprivileged domains (DomUs).

Resource Agent for the Xen Hypervisor.
Manages Xen virtual machine instances by mapping cluster

Name	Value
cpu	4
memory	4096

ID: xen1-utilization-cpu

Name: cpu

Value: 4

After you have configured the capacities your nodes provide and the capacities your resources require, you need to set the placement strategy in the global cluster options, otherwise the capacity configurations have no effect. Several strategies are available to schedule the load: for example, you can concentrate it on as few nodes as possible, or balance it evenly over all available nodes. For more information, refer to Section 4.4.5, “Placing Resources Based on Their Load Impact” (page 53).

Procedure 5.10 *Setting the Placement Strategy*

1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Connecting to a Cluster” (page 58).

2 Select *View* > *Simple Mode*.

3 In the left pane, select *CRM Config* to view the global cluster options and their current values.

4 Depending on your requirements, set *Placement Strategy* to the appropriate value.

5 If you need to disable fencing for any reasons, deselect `Stonith Enabled`.

6 Confirm your changes with *Apply*.

5.3.7 Configuring Resource Monitoring

Although the High Availability Extension can detect a node failure, it also has the ability to detect when an individual resource on a node has failed. If you want to ensure that a resource is running, you must configure resource monitoring for it. Resource monitoring consists of specifying a timeout and/or start delay value, and an interval. The interval tells the CRM how often it should check the resource status. You can also set particular parameters, such as `Timeout` for `start` or `stop` operations.

Procedure 5.11 *Adding or Modifying Monitor Operations*

1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Connecting to a Cluster” (page 58).

2 In the Pacemaker GUI main window, click *Resources* in the left pane to see the resources already configured for the cluster.

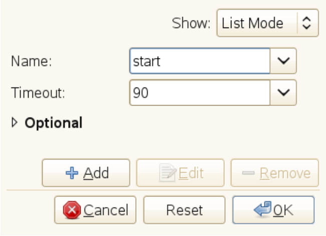
3 In the right pane, select the resource to modify and click *Edit*. The next window shows the basic resource parameters and the meta attributes, instance attributes and operations already defined for that resource.

4 To add a new monitor operation, select the respective tab and click *Add*.

To modify an existing operation, select the respective entry and click *Edit*.

5 In *Name*, select the action to perform, for example `monitor`, `start`, or *stop*.

The parameters shown below depend on the selection you make here.



Show: List Mode

Name: start

Timeout: 90

6. In the *Timeout* field, enter a value in seconds. After the specified timeout period, the operation will be treated as `failed`. The PE will decide what to do or execute what you specified in the *On Fail* field of the monitor operation.

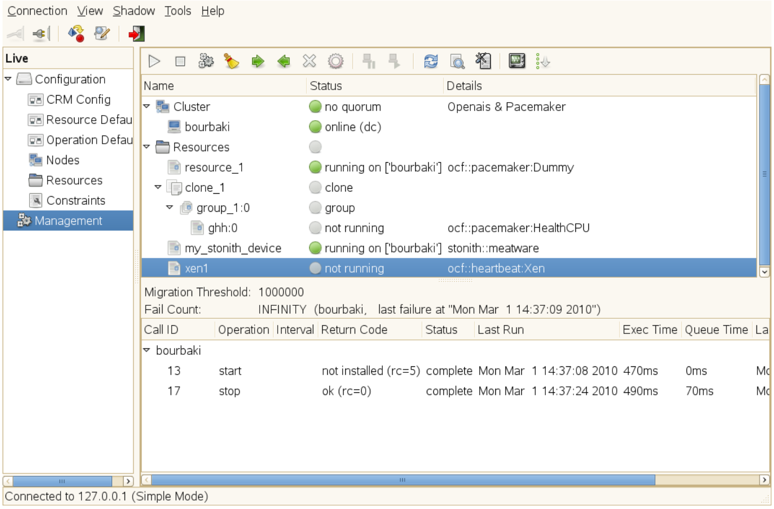
7. If needed, expand the *Optional* section and add parameters, like *On Fail* (what to do if this action ever fails?) or *Requires* (what conditions need to be satisfied before this action occurs?).

8. If all parameters are set according to your wishes, click *OK* to finish the configuration of that resource. The configuration dialog is closed and the main window shows the modified resource.

For the processes which take place if the resource monitor detects a failure, refer to Section 4.3, "Resource Monitoring" (page 48).

To view resource failures in the Pacemaker GUI, click *Management* in the left pane, then select the resource whose details you want to see in the right pane. For a resource that has failed, the *Fail Count* and last failure of the resource is shown in the middle of the right pane (below the *Migration threshold* entry).

***Figure 5.6** Viewing a Resource's Failcount*



Name	Status	Details
Cluster	● no quorum	Openais & Pacemaker
bourbaki	● online (dc)	
Resources	○	
resource_1	● running on ['bourbaki']	ocf::pacemaker:Dummy
clone_1	○ clone	
group_1:0	○ group	
ghh:0	○ not running	ocf::pacemaker:HealthCPU
my_stonith_device	● running on ['bourbaki']	stonith::meatware
xen1	○ not running	ocf::heartbeat:Xen

Migration Threshold: 1000000
Fail Count: INFINITY (bourbaki, last failure at "Mon Mar 1 14:37:09 2010")

Call ID	Operation	Interval	Return Code	Status	Last Run	Exec Time	Queue Time	La
bourbaki								
13	start		not installed (rc=5)	complete	Mon Mar 1 14:37:08 2010	470ms	0ms	Mo
17	stop		ok (rc=0)	complete	Mon Mar 1 14:37:24 2010	490ms	70ms	Mo

Connected to 127.0.0.1 (Simple Mode)

5.3.8 Configuring a Cluster Resource Group

Some cluster resources are dependent on other components or resources, and require that each component or resource starts in a specific order and runs together on the same server. To simplify this configuration we support the concept of groups.

For an example of a resource group and more information about groups and their properties, refer to Section "Groups" (page 41).

NOTE: Empty Groups

Groups must contain at least one resource, otherwise the configuration is not valid.

Procedure 5.12 *Adding a Resource Group*

1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, "Connecting to a Cluster" (page 58).

2 In the left pane, select *Resources* and click *Add* > *Group*.

3 Enter a unique *ID* for the group.

4 Below *Options*, set the *Initial state of resource* and click *Forward*.

5 In the next step, you can add primitives as sub-resources for the group. These are created similar as described in Procedure 5.2, "Adding Primitive Resources" (page 62).

6 If all parameters are set according to your wishes, click *Apply* to finish the configuration of the primitive.

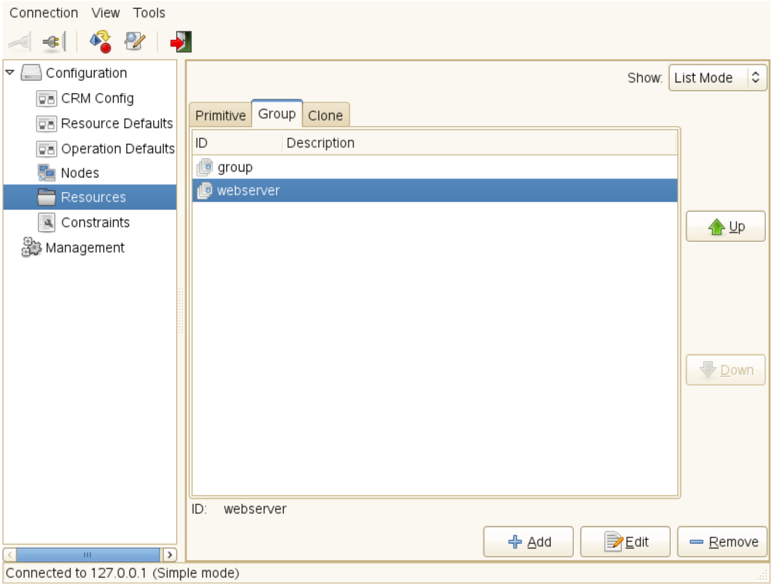
7 In the next window, you can continue adding sub-resources for the group by choosing *Primitive* again and clicking *OK*.

   When you do not want to add more primitives to the group, click *Cancel* instead. The next window shows a summary of the parameters that you have already defined for that group. The *Meta Attributes* and *Primitives* of the group are listed. The position of the resources in the *Primitive* tab represents the order in which the resources are started in the cluster.

8 As the order of resources in a group is important, use the *Up* and *Down* buttons to sort the *Primitives* in the group.

9 If all parameters are set according to your wishes, click *OK* to finish the configuration of that group. The configuration dialog is closed and the main window shows the newly created or modified group.

Figure 5.7 *Pacemaker GUI - Groups*



Show: List Mode

ID	Description
group	
webserver	

ID: webserver

Connected to 127.0.0.1 (Simple mode)

Let us assume you already have created a resource group as explained in Procedure 5.12, “Adding a Resource Group” (page 78). The following procedure shows you how to modify the group to match Example 4.1, “Resource Group for a Web Server” (page 41).

Procedure 5.13 *Adding Resources to an Existing Group*

1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Connecting to a Cluster” (page 58).

2 In the left pane, switch to the *Resources* view and in the right pane, select the group to modify and click *Edit*. The next window shows the basic group parameters and the meta attributes and primitives already defined for that resource.

3 Click the *Primitives* tab and click *Add*.

4 In the next dialog, set the following parameters to add an IP address as sub-resource of the group:

  4a Enter a unique *ID* (for example, `my_ipaddress`).

  4b From the *Class* list, select *ocf* as resource agent class.

  4c As *Provider* of your OCF resource agent, select *heartbeat*.

  4d From the *Type* list, select *IPaddr* as resource agent.

  4e Click *Forward*.

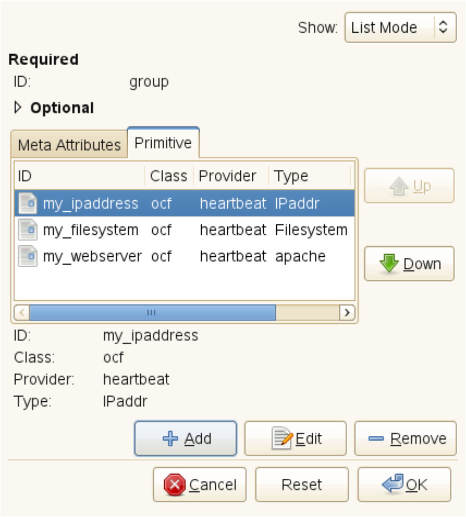
  4f In the *Instance Attribute* tab, select the *IP* entry and click *Edit* (or double-click the *IP* entry).

  4g As *Value*, enter the desired IP address, for example, `192.168.1.1`.

  4h Click *OK* and *Apply*. The group configuration dialog shows the newly added primitive.

5 Add the next sub-resources (file system and Web server) by clicking *Add* again.

6 Set the respective parameters for each of the sub-resources similar to steps Step 4a (page 80) to Step 4h (page 80), until you have configured all sub-resources for the group.


Show: List Mode

Required

ID: group

Optional

ID	Class	Provider	Type
my_ipaddress	ocf	heartbeat	IPaddr
my_filesystem	ocf	heartbeat	Filesystem
my_webserver	ocf	heartbeat	apache

ID: my_ipaddress
Class: ocf
Provider: heartbeat
Type: IPaddr

As we configured the sub-resources already in the order in that they need to be started in the cluster, the order on the *Primitives* tab is already correct.

7 In case you need to change the resource order for a group, use the *Up* and *Down* buttons to sort the resources on the *Primitive* tab.

8 To remove a resource from the group, select the resource on the *Primitives* tab and click *Remove*.

9 Click *OK* to finish the configuration of that group. The configuration dialog is closed and the main window shows the modified group.

5.3.9 Configuring a Clone Resource

You may want certain resources to run simultaneously on multiple nodes in your cluster. To do this you must configure a resource as a clone. Examples of resources that might be configured as clones include STONITH and cluster file systems like OCFS2. You can clone any resource provided. This is supported by the resource's Resource Agent. Clone resources may even be configured differently depending on which nodes they are hosted.

For an overview which types of resource clones are available, refer to Section "Clones" (page 42).

Procedure 5.14 *Adding or Modifying Clones*

1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Connecting to a Cluster” (page 58).

2 In the left pane, select *Resources* and click *Add* > *Clone*.

3 Enter a unique *ID* for the clone.

4 Below *Options*, set the *Initial state of resource*.

5 Activate the respective options you want to set for your clone and click *Forward*.

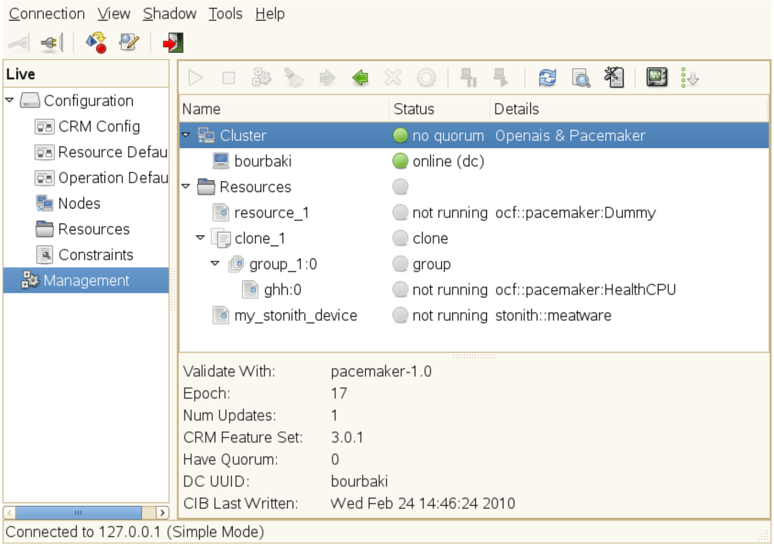
6 In the next step, you can either add a *Primitive* or a *Group* as sub-resources for the clone. These are created similar as described in Procedure 5.2, “Adding Primitive Resources” (page 62) or Procedure 5.12, “Adding a Resource Group” (page 78).

7 If all parameters in the clone configuration dialog are set according to your wishes, click *Apply* to finish the configuration of the clone.

5.4 Managing Cluster Resources

Apart from the possibility to configure your cluster resources, the Pacemaker GUI also allows you to manage existing resources. To switch to a management view and to access the available options, click *Management* in the left pane.

Figure 5.8 Pacemaker GUI - Management



Name	Status	Details
Cluster	no quorum	Openais & Pacemaker
bourbaki	online (dc)	
Resources		
resource_1	not running	ocf::pacemaker:Dummy
clone_1	clone	
group_1:0	group	
ghh:0	not running	ocf::pacemaker:HealthCPU
my_stonith_device	not running	stonith::meatware

```
Validate With:       pacemaker-1.0
Epoch:               17
Num Updates:         1
CRM Feature Set:     3.0.1
Have Quorum:         0
DC UUID:             bourbaki
CIB Last Written:    Wed Feb 24 14:46:24 2010
```

Connected to 127.0.0.1 (Simple Mode)

5.4.1 Starting Resources

Before you start a cluster resource, make sure it is set up correctly. For example, if you want to use an Apache server as a cluster resource, set up the Apache server first and complete the Apache configuration before starting the respective resource in your cluster.

NOTE: Do Not Touch Services Managed by the Cluster

When managing a resource with the High Availability Extension, the same resource must not be started or stopped otherwise (outside of the cluster, for example manually or on boot or reboot). The High Availability Extension software is responsible for all service start or stop actions.

However, if you want to check if the service is configured properly, start it manually, but make sure that it is stopped again before High Availability takes over.

For interventions in resources that are currently managed by the cluster, set the resource to `unmanaged mode` first as described in Section 5.4.5, "Changing Management Mode of Resources" (page 88).

During creation of a resource with the Pacemaker GUI, you can set the resource's initial state with the `target-role` meta attribute. If its value has been set to `stopped`, the resource does not start automatically after being created.

Procedure 5.15 *Starting A New Resource*

1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, "Connecting to a Cluster" (page 58).

2 Click *Management* in the left pane.

3 In the right pane, right-click the resource and select *Start* from the context menu (or use the *Start Resource* icon in the toolbar).

5.4.2 Cleaning Up Resources

A resource will be automatically restarted if it fails, but each failure raises the resource's failcount. View a resource's failcount with the Pacemaker GUI my clicking *Management* in the left pane, then selecting the resource in the right pane. If a resource has failed, its *Fail Count* is shown in the middle of the right pane (below the *Migration Threshold* entry).

If a `migration-threshold` has been set for that resource, the node will no longer be allowed to run the resource as soon as the number of failures has reached the migration threshold.

A resource's failcount can either be reset automatically (by setting a `failure-timeout` option for the resource) or you can reset it manually as described below.

Procedure 5.16 *Cleaning Up A Resource*

1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, "Connecting to a Cluster" (page 58).

2 Click *Management* in the left pane.

3 In the right pane, right-click the respective resource and select *Cleanup Resource* from the context menu (or use the *Cleanup Resource* icon in the toolbar).

   This executes the commands `crm_resource -C` and `crm_failcount -D` for the specified resource on the specified node.

For more information, see also crm_resource(8) (page 242) and crm_failcount(8) (page 233).

5.4.3 Removing Cluster Resources

If you need to remove a resource from the cluster, follow the procedure below to avoid configuration errors:

NOTE: Removing Referenced Resources

Cluster resources cannot be removed if their ID is referenced by any constraint. If you cannot delete a resource, check where the resource ID is referenced and remove the resource from the constraint first.

Procedure 5.17 *Removing a Cluster Resource*

1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Connecting to a Cluster” (page 58).

2 Click *Management* in the left pane.

3 Select the respective resource in the right pane.

4 Clean up the resource on all nodes as described in Procedure 5.16, “Cleaning Up A Resource” (page 84).

5 *Stop* the resource.

6 Remove all constraints that relate to the resource, otherwise removing the resource will not be possible.

5.4.4 Migrating Cluster Resources

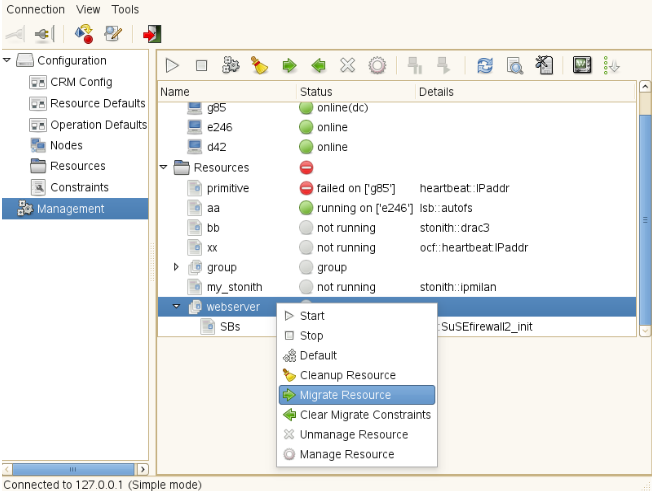
As mentioned in Section 5.3.4, “Specifying Resource Failover Nodes” (page 70), the cluster will fail over (migrate) resources automatically in case of software or hardware failures—according to certain parameters you can define (for example, migration threshold or resource stickiness). Apart from that, you can also manually migrate a resource to another node in the cluster resources manually.

***Procedure 5.18** Manually Migrating a Resource*

1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Connecting to a Cluster” (page 58).

2 Click *Management* in the left pane.

3 Right-click the respective resource in the right pane and select *Migrate Resource*.



Name	Status	Details
g85	online(dc)	
e246	online	
d42	online	
Resources		
primitive	failed on ['g85']	heartbeat::IPaddr
aa	running on ['e246']	lsb::autofs
bb	not running	stonith::drac3
xx	not running	ocf::heartbeat:IPaddr
group	group	
my_stonith	not running	stonith::ipmilan
webserver		
SBs		:SuSEfirewall2_init

4 In the new window, select the node to which to move the resource to in *To Node*. This creates a location constraint with an `INFINITY` score for the destination node.

5 If you want to migrate the resource only temporarily, activate *Duration* and enter the time frame for which the resource should migrate to the new node. After the expiration of the duration, the resource *can* move back to its original location or it may stay where it is (depending on resource stickiness).

6 In cases where the resource cannot be migrated (if the resource's stickiness and constraint scores total more than `INFINITY` on the current node), activate the *Force* option. This forces the resource to move by creating a rule for the current location and a score of `-INFINITY`.

> **NOTE**
>
> This prevents the resource from running on this node until the constraint is removed with *Clear Migrate Constraints* or the duration expires.

7 Click *OK* to confirm the migration.

To allow a resource to move back again, proceed as follows:

Procedure 5.19 *Clearing a Migration Constraint*

1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Connecting to a Cluster” (page 58).

2 Click *Management* in the left pane.

3 Right-click the respective resource in the right pane and select *Clear Migrate Constraints*.

This uses the `crm_resource -U` command. The resource *can* move back to its original location or it may stay where it is (depending on resource stickiness).

For more information, see crm_resource(8) (page 242) or *Pacemaker 1.0—Configuration Explained*, available from `http://clusterlabs.org/wiki/Documentation`. Refer to section *Resource Migration*.

5.4.5 Changing Management Mode of Resources

When a resource is being managed by the cluster, it must not be touched otherwise (outside of the cluster). For maintenance of individual resources, you can set the respective resources to an `unmanaged mode` that allows you to modify the resource outside of the cluster.

Procedure 5.20 *Changing Management Mode of Resources*

1. Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Connecting to a Cluster” (page 58).

2. Click *Management* in the left pane.

3. Right-click the respective resource in the right pane and from the context menu, select *Unmanage Resource*.

4. After you have finished the maintenance task for that resource, right-click the respective resource again in the right pane and select *Manage Resource*.

   From this point on, the resource will be managed by the High Availability Extension software again.

Configuring and Managing Cluster Resources (Command Line)

6

To configure and manage cluster resources, either use the graphical user interface (the Pacemaker GUI) or the `crm` command line utility. For the GUI approach, refer to Chapter 5, *Configuring and Managing Cluster Resources (GUI)* (page 57).

This chapter introduces `crm`, the command line tool and covers an overview of this tool, how to use templates, and mainly configuring and managing cluster resources: creating basic and advanced types of resources (groups and clones), configuring constraints, specifying failover nodes and failback nodes, configuring resource monitoring, starting, cleaning up or removing resources, and migrating resources manually.

NOTE: User Privileges

Sufficient privileges are necessary in order to manage a cluster. The `crm` commands and its subcommands have to be run either as `root` user or as the CRM owner user (typically the user `hacluster`).

However, the `user` option allows you to run the `crm` and its subcommands as a regular (unprivileged) user and to change its ID using `sudo` whenever necessary. For example, with the following command `crm` will use `hacluster` as the privileged user ID:

```
crm options user hacluster
```

Note that you need to set up `/etc/sudoers` in such a way that `sudo` does not ask for a password.

6.1 crm Command Line Tool—Overview

After installation you usually need the `crm` command only. This command has several subcommands which manage resources, CIBs, nodes, resource agents, and others. Run `crm help` to get an overview of all available commands. It offers a thorough help system with embedded examples.

The `crm` command can be used in the following ways:

- **Directly** Add all subcommands to `crm`, press Enter and you see the output immediately. For example, enter `crm help ra` to get information about the `ra` subcommand (resource agents).

- **As Shell Script** Use `crm` and a script with `crm` commands. This can be done in two ways:

  ```
  crm -f script.cli
  crm < script.cli
  ```

  The script can contain any command from `crm`. For example:

  ```
  # A small example
  status
  node list
  ```

  Any line starting with the hash symbol (#) is a comment and is ignored. If a line is too long, insert a backslash (\) at the end and continue in the next line.

- **Interactive As Internal Shell** Type `crm` to enter the internal shell. The prompt changes to `crm(live)#`. With `help` you can get an overview of the available subcommands. As the internal shell has different levels of subcommands, you can “enter” one by just typing this subcommand and press Enter.

  For example, if you type `resource` you enter the resource management level. Your prompt changes to `crm(live)resource#`. If you want to leave the internal shell, use the commands `quit`, `bye`, or `exit`. If you need to go one level back, use `up`, `end`, or `cd`.

It is possible to enter the level directly, if you type `crm`, the respective subcommand and no options.

The internal shell supports also tab completion for subcommands and resources. Type the beginning of a command, press →| and `crm` completes the respective object.

NOTE: Differentiate Between Management and Configuration Subcommands

The `crm` tool has management capability (the subcommands `resource` and `node`) and can be used for configuration (`cib`, `configure`).

The following subsections give you an overview about some important aspects of the `crm` tool.

6.1.1 Displaying Information about OCF Resource Agents

As you have to deal with resource agents in your cluster configuration all the time, the `crm` tool contains the `ra` command to get information about resource agents and to manage them (for additional information, see also Section 4.2.2, “Supported Resource Agent Classes” (page 38)):

```
# crm ra
crm(live)ra#
```

The command `classes` gives you a list of all classes and providers:

```
crm(live)ra# classes
heartbeat
lsb
ocf / heartbeat linbit lvm2 ocfs2 pacemaker
stonith
```

To get an overview about all available resource agents for a class (and provider) use the `list` command:

```
crm(live)ra# list ocf
AoEtarget           AudibleAlarm        CTDB                ClusterMon
Delay               Dummy               EvmsSCC             Evmsd
Filesystem          HealthCPU           HealthSMART         ICP
IPaddr              IPaddr2             IPsrcaddr           IPv6addr
```

```
LVM                 LinuxSCSI           MailTo              ManageRAID
ManageVE            Pure-FTPd           Raid1               Route
SAPDatabase         SAPInstance         SendArp             ServeRAID
...
```

An overview about a resource agent can be viewed with `info`:

```
crm(live)ra# info ocf:drbd:linbit
This resource agent manages a DRBD resource
as a master/slave resource. DRBD is a shared-nothing replicated storage
device. (ocf:linbit:drbd)

Master/Slave OCF Resource Agent for DRBD

Parameters (* denotes required, [] the default):

drbd_resource* (string): drbd resource name
    The name of the drbd resource from the drbd.conf file.

drbdconf (string, [/etc/drbd.conf]): Path to drbd.conf
    Full path to the drbd.conf file.

Operations' defaults (advisory minimum):

    start         timeout=240
    promote       timeout=90
    demote        timeout=90
    notify        timeout=90
    stop          timeout=100
    monitor_Slave_0 interval=20 timeout=20 start-delay=1m
    monitor_Master_0 interval=10 timeout=20 start-delay=1m
```

Leave the viewer by pressing Q. Find a configuration example at Appendix A, *Example of Setting Up a Simple Testing Resource* (page 367).

TIP: Use `crm` Directly

In the former example we used the internal shell of the `crm` command. However, you do not necessarily have to use it. You get the same results, if you add the respective subcommands to `crm`. For example, you can list all the OCF resource agents by entering `crm ra list ocf` in your shell.

6.1.2 Using Templates

Templates are ready-made cluster configurations. They require minimum effort to be tailored to the particular user's needs. Whenever a template creates a configuration, warning messages give hints which can be edited later for further customization.

The following procedure shows how to create a simple yet functional Apache configuration:

1 Log in as `root`.

2 Start the `crm` tool:

```
# crm configure
```

3 Create a new configuration from a template:

3a Switch to the `template` subcommand:

```
crm(live)configure# template
```

3b List the available templates:

```
crm(live)configure template#  list templates
gfs2-base   filesystem  virtual-ip  apache   clvm     ocfs2    gfs2
```

3c Decide which template you need. As we need an Apache configuration, we choose the `apache` template:

```
crm(live)configure template#  new intranet apache
INFO: pulling in template apache
INFO: pulling in template virtual-ip
```

4 Define your parameters:

4a List the just created configuration:

```
crm(live)configure template#  list
intranet
```

4b Display the minimum of required changes which have to be filled out by you:

```
crm(live)configure template#  show
ERROR: 23: required parameter ip not set
ERROR: 61: required parameter id not set
ERROR: 65: required parameter configfile not set
```

4c Invoke your preferred text editor and fill out all lines that have been displayed as errors in Step 4b (page 93):

```
crm(live)configure template#  edit
```

5 Show the configuration and check whether it is valid (bold text depends on the configuration you have entered in Step 4c (page 94)):

```
crm(live)configure template#  show
primitive virtual-ip ocf:heartbeat:IPaddr \
    params ip="192.168.1.101"
primitive apache ocf:heartbeat:apache \
    params configfile="/etc/apache2/httpd.conf"
monitor apache 120s:60s
group intranet \
    apache virtual-ip
```

6 Apply the configuration:

```
crm(live)configure template#  apply
crm(live)configure#  cd ..
crm(live)configure#  show
```

7 Submit your changes to the CIB:

```
crm(live)configure#  commit
```

It is possible to simplify the commands even more, if you know the details. The above procedure can be summarized with the following command on the shell:

```
crm configure template \
   new intranet apache params \
   configfile="/etc/apache2/httpd.conf" ip="192.168.1.101"
```

If you are inside your internal `crm` shell, use the following command:

```
crm(live)configure template# new intranet apache params \
   configfile="/etc/apache2/httpd.conf" ip="192.168.1.101"
```

However, the previous command only creates its configuration from the template. It does not apply nor commit it to the CIB.

6.1.3 Testing with Shadow Configuration

A shadow configuration is used to test different configuration scenarios. If you have created several shadow configurations, you can test them one by one to see the effects of your changes.

The usual process looks like this:

1 Open a shell and become `root`.

2 Start the crm shell with the following command:

```
crm configure
```

3 Create a new shadow configuration:

```
crm(live)configure# cib new myNewConfig
INFO: myNewConfig shadow CIB created
```

4 If you want to copy the current live configuration into your shadow configuration, use the following command, otherwise skip this step:

```
crm(myNewConfig)# cib reset myNewConfig
```

The previous command makes it easier to modify any existing resources later.

5 Make your changes as usual. After you have created the shadow configuration, all changes go there. To save all your changes, use the following command:

```
crm(myNewConfig)#
```

6 If you need the live cluster configuration again, switch back with the following command:

```
crm(myNewConfig)configure# cib use live
crm(live)#
```

6.1.4 Debugging Your Configuration Changes

Before loading your configuration changes back into the cluster, it is recommended to review your changes with `ptest`. The `ptest` can show a diagram of actions that will be induced by committing the changes. You need the `graphviz` package to display the diagrams. The following example is a transcript, adding a monitor operation:

```
# crm configure
crm(live)configure# show fence-node2
primitive fence-node2 stonith:apcsmart \
        params hostlist="node2"
crm(live)configure# monitor fence-node2 120m:60s
crm(live)configure# show changed
primitive fence-node2 stonith:apcsmart \
        params hostlist="node2" \
        op monitor interval="120m" timeout="60s"
crm(live)configure# ptest
crm(live)configure# commit
```

6.2 Configuring Global Cluster Options

Global cluster options control how the cluster behaves when confronted with certain situations. They can be viewed and modified with the `crm` tool. The predefined values can be kept in most cases. However, to make key functions of your cluster work correctly, you need to adjust the following parameters after basic cluster setup:

- Option `no-quorum-policy` (page 36)

- Option `stonith-enabled` (page 37)

***Procedure 6.1** Modifying Global Cluster Options With crm*

1 Open a shell and become `root`.

2 Enter `crm configure` to open the internal shell.

3 Use the following commands to set the options for a two-node clusters only:

```
crm(live)configure# property no-quorum-policy=ignore
crm(live)configure# property stonith-enabled=false
```

4 Show your changes:

```
crm(live)configure# show
property $id="cib-bootstrap-options" \
   dc-version="1.1.1-530add2a3721a0ecccb24660a97dbfdaa3e68f51" \
   cluster-infrastructure="openais" \
   expected-quorum-votes="2" \
   no-quorum-policy="ignore" \
   stonith-enabled="false"
```

5 Commit your changes and exit:

```
crm(live)configure# commit
crm(live)configure# exit
```

6.3 Configuring Cluster Resources

As a cluster administrator, you need to create cluster resources for every resource or application you run on servers in your cluster. Cluster resources can include Web sites, e-mail servers, databases, file systems, virtual machines, and any other server-based applications or services you want to make available to users at all times.

For an overview of resource types you can create, refer to Section 4.2.3, “Types of Resources” (page 40).

6.3.1 Creating Cluster Resources

There are three types of RAs (Resource Agents) available with the cluster (for background information, see Section 4.2.2, “Supported Resource Agent Classes” (page 38)). To create a cluster resource use the `crm` tool. To add a new resource to the cluster, proceed as follows:

1 Open a shell and become `root`.

2 Enter `crm configure` to open the internal shell

3 Configure a primitive IP address:

```
crm(live)configure# primitive myIP ocf:heartbeat:IPaddr \
     params ip=127.0.0.99 op monitor interval=60s
```

The previous command configures a “primitive” with the name `myIP`. You need to choose a class (here `ocf`), provider (`heartbeat`), and type (`IPaddr`). Furthermore, this primitive expects other parameters like the IP address. Change the address to your setup.

4 Display and review the changes you have made:

```
crm(live)configure# show
```

5 Commit your changes to take effect:

```
crm(live)configure# commit
```

6.3.2 Example Configuration for an NFS Server

To set up the NFS server, you need to complete the following:

1 Configure DRBD.

2 Set up a file system Resource.

3 Set up the NFS server and configure the IP address.

Learn how to achieve this in the following subsection.

Configuring DRBD

Before starting with the DRBD High Availability configuration, set up a DRBD device manually. Basically this is configuring DRBD and letting it synchronize. The exact procedure is described in Chapter 13, *Distributed Replicated Block Device (DRBD)* (page 165). For now, assume that you configured a resource `r0` that may be accessed at the device `/dev/drbd_r0` on both of your cluster nodes.

The DRBD resource is an OCF master/slave resource. This can be found in the description of the metadata of the DRBD resource agent. However, it is important that the actions `promote` and `demote` exist in the `actions` section of the metadata. These are mandatory for master/slave resources and commonly not available to other resources.

For High Availability, master/slave resources may have multiple masters on different nodes. It is even possible to have a master and slave on the same node. Therefore, configure this resource in a way that there is exactly one master and one slave, each running on different nodes. Do this with the meta attributes of the `master` resource. Master/slave resources are special types of clone resources in High Availability. Every master and every slave counts as a clone.

Proceed as follows to configure a DRBD resource:

1 Open a shell and become `root`.

2 Enter `crm configure` to open the internal shell.

3 If you have a two-node cluster, set the following properties per `ms` resource:

```
crm(live)configure# primitive my-stonith stonith:external/ipmi ...
crm(live)configure# ms ms_drbd_r0 res_drbd_r0 meta \
   globally-unique=false ...
crm(live)configure# property no-quorum-policy=ignore
crm(live)configure# property stonith-enabled=true
```

4 Create a primitive DRBD resource:

```
crm(live)configure# primitive drbd_r0 ocf:linbit:drbd params \
 drbd drbd_resource=r0 op monitor interval="30s"
```

5 Create a master/slave resource:

```
crm(live)configure# ms ms_drbd_r0 res_drbd_r0 meta master-max=1 \
 master-node-max=1 clone-max=2 clone-node-max=1 notify=true
```

6 Specify an colocation and order constraint:

```
crm(live)configure# colocation fs_on_drbd_r0 inf: res_fs_r0 ms_drbd_r0:Master
crm(live)configure# order fs_after_drbd_r0 inf: ms_drbd_r0:promote
res_fs_r0:start
```

7 Display your changes with the `show` command.

8 Commit your changes with the `commit` command.

Setting Up a File System Resource

The filesystem resource is configured as an OCF primitive resource with DRBD. It has the task of mounting and unmounting a device to a directory on start and stop requests. In this case, the device is /dev/drbd_r0 and the directory to use as mount point is /srv/failover. The file system used is xfs.

Use the following commands in the crm shell to configure a file system resource:

```
crm(live)# configure
crm(live)configure# primitive filesystem_resource \
    ocf:linbit:drbd \
    params device=/dev/drbd_r0 directory=/srv/failover fstype=xfs
```

NFS Server and IP Address

To make the NFS server always available at the same IP address, use an additional IP address as well as the ones the machines use for their normal operations. This IP address is then assigned to the active NFS server in addition to the system's IP address.

The NFS server and the IP address of the NFS server should always be active on the same machine. In this case, the start sequence is not very important. They may even be started at the same time. These are the typical requirements for a group resource.

Before starting the High Availability RA configuration, configure the NFS server with YaST. Do not let the system start the NFS server. Just set up the configuration file. If you want to do that manually, see the manual page exports(5) (man 5 exports). The configuration file is /etc/exports. The NFS server is configured as an LSB resource.

Configure the IP address completely with the High Availability RA configuration. No additional modification is necessary in the system. The IP address RA is an OCF RA.

```
crm(live)# configure
crm(live)configure# primitive nfs_resource ocf:nfsserver \
     params nfs_ip=10.10.0.1  nfs_shared_infodir=/shared
crm(live)configure# primitive ip_resource ocf:heartbeat:IPaddr \
     params ip=10.10.0.1
crm(live)configure# group nfs_group nfs_resource ip_resource
crm(live)configure# show
primitive ip_res ocf:heartbeat:IPaddr \
        params ip="192.168.1.10"
primitive nfs_res ocf:heartbeat:nfsserver \
        params nfs_ip="192.168.1.10" nfs_shared_infodir="/shared"
```

```
group nfs_group nfs_res ip_res
crm(live)configure# commit
crm(live)configure# end
crm(live)# quit
```

6.3.3 Creating a STONITH Resource

From the `crm` perspective, a STONITH device is just another resource. To create a STONITH resource, proceed as follows:

1. Open a shell and become `root`.

2. Enter `crm` to open the internal shell.

3. Get a list of all STONITH types with the following command:

```
crm(live)# ra list stonith
apcmaster               apcsmart                baytech
cyclades                drac3                   external/drac5
external/hmchttp        external/ibmrsa         external/ibmrsa-telnet
external/ipmi           external/kdumpcheck     external/rackpdu
external/riloe          external/sbd            external/ssh
external/vmware         external/xen0           external/xen0-ha
ibmhmc                  ipmilan                 meatware
null                    nw_rpc100s              rcd_serial
rps10                   ssh                     suicide
```

4. Choose a STONITH type from the above list and view the list of possible options. Use the following command:

```
crm(live)# ra info stonith:external/ipmi
IPMI STONITH external device (stonith:external/ipmi)

ipmitool based power management. Apparently, the power off
method of ipmitool is intercepted by ACPI which then makes
a regular shutdown. If case of a split brain on a two-node
it may happen that no node survives. For two-node clusters
use only the reset method.

Parameters (* denotes required, [] the default):

hostname (string): Hostname
    The name of the host to be managed by this STONITH device.
...
```

5 Create the STONITH resource with the `stonith` class, the type you have chosen in Step 4, and the respective parameters if needed, for example:

```
crm(live)# configure
crm(live)configure# primitive my-stonith stonith:external/ipmi \
    params hostname="node1"
    ipaddr="192.168.1.221" \
    userid="admin" passwd="secret" \
    op monitor interval=60m timeout=120s
```

6.3.4 Configuring Resource Constraints

Having all the resources configured is only one part of the job. Even if the cluster knows all needed resources, it might still not be able to handle them correctly. For example, try not to mount the file system on the slave node of drbd (in fact, this would fail with drbd). Define constraints to make these kind of information available to the cluster.

For more information about constraints, see Section 4.4, “Resource Constraints” (page 49).

Locational Constraints

This type of constraint may be added multiple times for each resource. All `location` constraints are evaluated for a given resource. A simple example that expresses a preference to run the resource `fs1` on the node with the name `earth` to 100 would be the following:

```
crm(live)configure# location fs1-loc fs1 100: earth
```

Another example is a location with pingd:

```
crm(live)configure# primitive pingd pingd \
    params name=pingd dampen=5s multiplier=100 host_list="r1 r2"
crm(live)configure#  location node_pref internal_www \
    rule 50: #uname eq node1 \
    rule pingd: defined pingd
```

Collocational Constraints

The `collocation` command is used to define what resources should run on the same or on different hosts.

It is only possible to set a score of either +inf or -inf, defining resources that must always or must never run on the same node. It is also possible to use non-infinite scores. In that case the collocation is called *advisory* and the cluster may decide not to follow them in favor of not stopping other resources if there is a conflict.

For example, to the resources with the IDs `filesystem_resource` and `nfs_group` always on the same host, use the following constraint:

```
crm(live)configure# colocation nfs_on_filesystem inf: nfs_group
filesystem_resource
```

For a master slave configuration, it is necessary to know if the current node is a master in addition to running the resource locally.

Ordering Constraints

Sometimes it is necessary to provide an order of resource actions or operations. For example, you cannot mount a file system before the device is available to a system. Ordering constraints can be used to start or stop a service right before or after a different resource meets a special condition, such as being started, stopped, or promoted to master. Use the following commands in the `crm` shell to configure an ordering constraint:

```
crm(live)configure# order nfs_after_filesystem mandatory: filesystem_resource
 nfs_group
```

Constraints for the Example Configuration

The example used for this chapter would not work without additional constraints. It is essential that all resources run on the same machine as the master of the drbd resource. The drbd resource must be master before any other resource starts. Trying to mount the DRBD device when it is not the master simply fails. The following constraints must be fulfilled:

- The file system must always be on the same node as the master of the DRBD resource.

  ```
  crm(live)configure# colocation filesystem_on_master inf: \
      filesystem_resource drbd_resource:Master
  ```

- The NFS server as well as the IP address must be on the same node as the file system.

  ```
  crm(live)configure# colocation nfs_with_fs inf: \
     nfs_group filesystem_resource
  ```

- The NFS server as well as the IP address start after the file system is mounted:

  ```
  crm(live)configure# order nfs_second mandatory: \
     filesystem_resource:start nfs_group
  ```

- The file system must be mounted on a node after the DRBD resource is promoted to master on this node.

  ```
  crm(live)configure# order drbd_first inf: \
      drbd_resource:promote filesystem_resource
  ```

6.3.5 Specifying Resource Failover Nodes

To determine a resource failover, use the meta attribute migration-threshold. For example:

```
crm(live)configure# location r1-node1 r1 100: node1
```

Normally, r1 prefers to run on node1. If it fails there, migration-threshold is checked and compared to the failcount. If failcount >= migration-threshold then it is migrated to the node with the next best preference.

Start failures set the failcount to inf depend on the `start-failure-is-fatal` option. Stop failures cause fencing. If there is no STONITH defined, the resource will not migrate at all.

For an overview, refer to Section 4.4.3, “Failover Nodes” (page 50).

6.3.6 Specifying Resource Failback Nodes (Resource Stickiness)

A resource might fail back to its original node when that node is back online and in the cluster. If you want to prevent a resource from failing back to the node it was running on prior to failover, or if you want to specify a different node for the resource to fail back to, you must change its resource stickiness value. You can either specify resource stickiness when you are creating a resource, or afterwards.

For an overview, refer to Section 4.4.4, “Failback Nodes” (page 52).

6.3.7 Configuring Placement of Resources Based on Load Impact

Configuring Placement of Resources Based on Load Impact

Not all resources are equal. Some, such as Xen guests, require that the node hosting them meets their capacity requirements. If resources are placed such that their combined need exceed the provided capacity, the resources diminish in performance (or even fail).

To take this into account, the High Availability Extension allows you to specify the following parameters:

1. The capacity a certain node *provides*.

2. The capacity a certain resource *requires*.

3. An overall strategy for placement of resources.

For detailed background information about the parameters and a configuration example, refer to Section 4.4.5, “Placing Resources Based on Their Load Impact” (page 53).

To configure the resource's requirements and the capacity a node provides, use utilization attributes as described in Procedure 5.9, “Adding Or Modifying Utilization Attributes” (page 73). You can name the utilization attributes according to your preferences and define as many name/value pairs as your configuration needs.

In the following example, we assume that you already have a basic configuration of cluster nodes and resources and now additionally want to configure the capacities a certain node provides and the capacity a certain resource requires.

Procedure 6.2 *Adding Or Modifying Utilization Attributes With crm*

1 Start the `crm` shell with the following command:

```
crm configure
```

2 To specify the capacity a node *provides*, use the following command and replace the placeholder *`NODE_1`* with the name of your node:

```
crm(live)configure# node NODE_1 utilization memory=16384 cpu=8
```

With these values, *NODE_1* would be assumed to provide 16GB of memory and 8 CPU cores to resources.

3 To specify the capacity a resource *requires*, use:

```
crm(live)configure# primitive xen1 ocf:heartbeat:Xen ... \
     utilization memory=4096 cpu=4
```

This would make the resource consume 4096 of those memory units from nodeA, and 4 of the cpu units.

4 Configure the placement strategy with the `property` command:

```
crm(live)configure# property ...
```

Four values are available for the placement strategy:

`propertyplacement-strategy=default`
: Utilization values are not taken into account at all, per default. Resources are allocated according to location scoring. If scores are equal, resources are evenly distributed across nodes.

`propertyplacement-strategy=utilization`
: Utilization values are taken into account when deciding whether a node is considered eligible if it has sufficient free capacity to satisfy the resource's requirements. However, load-balancing is still done based on the number of resources allocated to a node.

`propertyplacement-strategy=minimal`
: Utilization values are taken into account when deciding whether a node is eligible to serve a resource; an attempt is made to concentrate the resources on as few nodes as possible, thereby enabling possible power savings on the remaining nodes.

`propertyplacement-strategy=balanced`
: Utilization values are taken into account when deciding whether a node is eligible to serve a resource; an attempt is made to spread the resources evenly, optimizing resource performance.

The placing strategies are best-effort, and do not yet utilize complex heuristic solvers to always reach an optimum allocation result. Ensure that resource priorities are properly set so that your most important resources are scheduled first.

5 Commit your changes before leaving the crm shell:

```
crm(live)configure# commit
```

The following example demonstrates a three node cluster of equal nodes, with 4 virtual machines:

```
crm(live)configure# node node1 utilization memory="4000"
crm(live)configure# node node2 utilization memory="4000"
crm(live)configure# node node3 utilization memory="4000"
crm(live)configure# primitive xenA ocf:heartbeat:Xen \
    utilization memory="3500" meta priority="10"
crm(live)configure# primitive xenB ocf:heartbeat:Xen \
    utilization memory="2000" meta priority="1"
crm(live)configure# primitive xenC ocf:heartbeat:Xen \
    utilization memory="2000" meta priority="1"
crm(live)configure# primitive xenD ocf:heartbeat:Xen \
    utilization memory="1000" meta priority="5"
crm(live)configure# property placement-strategy="minimal"
```

With all three nodes up, xenA will be placed onto a node first, followed by xenD. xenB and xenC would either be allocated together or one of them with xenD.

If one node failed, too little total memory would be available to host them all. xenA would be ensured to be allocated, as would xenD; however, only one of xenB or xenC could still be placed, and since their priority is equal, the result is not defined yet. To resolve this ambiguity as well, you would need to set a higher priority for either one.

6.3.8 Configuring Resource Monitoring

To monitor a resource, there are two possibilities: either define a monitor operation with the `op` keyword or use the `monitor` command. The following example configures an Apache resource and monitors it every 60 seconds with the `op` keyword:

```
crm(live)configure# primitive apache apache \
  params ... \
  op monitor interval=60s timeout=30s
```

The same can be done with:

```
crm(live)configure# primitive apache apache \
   params ...
crm(live)configure# monitor apache 60s:30s
```

For an overview, refer to Section 4.3, “Resource Monitoring” (page 48).

6.3.9 Configuring a Cluster Resource Group

One of the most common elements of a cluster is a set of resources that needs to be located together. Start sequentially and stop in the reverse order. To simplify this configuration we support the concept of groups. The following example creates two primitives (an IP address and an e-mail resource):

1 Run the `crm` command as system administrator. The prompt changes to `crm(live)`.

2 Configure the primitives:

```
crm(live)# configure
crm(live)configure# primitive Public-IP ocf:IPaddr:heartbeat \
   params ip=1.2.3.4
crm(live)configure# primitive Email lsb:exim
```

3 Group the primitives with their relevant identifiers in the correct order:

```
crm(live)configure# group shortcut Public-IP Email
```

For an overview, refer to Section “Groups” (page 41).

6.3.10 Configuring a Clone Resource

Clones were initially conceived as a convenient way to start N instances of an IP resource and have them distributed throughout the cluster for load balancing. They have turned out to quite useful for a number of other purposes, including integrating with DLM, the fencing subsystem and OCFS2. You can clone any resource, provided the resource agent supports it.

Learn more about cloned resources in Section “Clones” (page 42).

Creating Anonymous Clone Resources

To create an anonymous clone resource, first create a primitive resource and then refer to it with the `clone` command. Do the following:

1 Open a shell and become `root`.

2 Enter `crm configure` to open the internal shell.

3 Configure the primitive, for example:

```
crm(live)configure# primitive Apache lsb:apache
```

4 Clone the primitive:

```
crm(live)configure# clone apache-clone Apache
```

Creating Stateful/Multi-State Clone Resources

To create an stateful clone resource, first create a primitive resource and then the master/slave resource.

1 Open a shell and become `root`.

2 Enter `crm configure` to open the internal shell.

3 Configure the primitive. Change the intervals if needed:

```
crm(live)configure# primitive myRSC ocf:myCorp:myAppl \
    op monitor interval=60 \
    op monitor interval=61 role=Master
```

4 Create the master slave resource:

```
crm(live)configure# clone apache-clone Apache
```

6.4 Managing Cluster Resources

Apart from the possibility to configure your cluster resources, the `crm` tool also allows you to manage existing resources. The following subsections gives you an overview.

6.4.1 Starting a New Cluster Resource

To start a new cluster resource you need the respective identifier. Proceed as follows:

1 Open a shell to become `root`.

2 Enter `crm` to open the internal shell.

3 Switch to the resource level:

```
crm(live)# resource
```

4 Start the resource with `start` and press the →| key to show all known resources:

```
crm(live)resource# start start ID
```

6.4.2 Cleaning Up Resources

A resource will be automatically restarted if it fails, but each failure raises the resource's failcount. If a `migration-threshold` has been set for that resource, the node will no longer be allowed to run the resource as soon as the number of failures has reached the migration threshold.

1 Open a shell and log in as user `root`.

2 Get a list of all your resources:

```
crm resource list
  ...
Resource Group: dlm-clvm:1
         dlm:1  (ocf::pacemaker:controld) Started
         clvm:1 (ocf::lvm2:clvmd) Started
         cmirrord:1     (ocf::lvm2:cmirrord) Started
```

3 If the resource is running, it has to be stopped first. Replace *`RSC`* with the name of the resource.

```
crm resource stop RSC
```

For example, if you want to stop the DLM resource, from the `dlm-clvm` resource group, replace *`RSC`* with `dlm`.

4 Delete the resource itself:

```
crm configure delete ID
```

6.4.3 Removing a Cluster Resource

To remove a cluster resource you need the relevant identifier. Proceed as follows:

1 Open a shell and become `root`.

2 Run the following command to get a list of your resources:

```
crm(live)# resource status
```

For example, the output can look like this (whereas myIP is the relevant identifier of your resource):

```
myIP    (ocf::IPaddr:heartbeat) ...
```

3 Delete the resource with the relevant identifier (which implies a `commit` too):

```
crm(live)# configure delete YOUR_ID
```

4 Commit the changes:

```
crm(live)# configure commit
```

6.4.4 Migrating a Cluster Resource

Although resources are configured to automatically fail over (or migrate) to other nodes of the cluster in the event of a hardware or software failure, you can also manually migrate a resource to another node in the cluster using either the Pacemaker GUI or the command line.

1 Open a shell to become `root`.

2 Enter `crm` to open the internal shell.

3 To migrate a resource named `ipaddress1` to a cluster node named `node2`, enter:

```
crm(live)# resource
crm(live)resource# migrate ipaddress1 node2
```

Managing Cluster Resources with the Web Interface

7

In addition to the crm command line tool and the Pacemaker GUI the High Availability Extension also comes with the HA Web Konsole, a Web-based user interface for management tasks. It allows you to monitor and administer your Linux cluster also from non-Linux machines. Furthermore, it is an ideal solution in case your system does not provide or allow a graphical user interface.

The Web interface is included in the hawk package. It must be installed on all cluster nodes you want to connect to with the HA Web Konsole. On the machine from which you want to access a cluster node using the HA Web Konsole, you only need a (graphical) Web browser with JavaScript and cookies enabled to establish the connection.

NOTE: User Authentication

To log in to the cluster from the HA Web Konsole, the respective user must be a member of the haclient group. The installation creates a Linux user named hacluster which is member of the haclient group.

Before using the HA Web Konsole, either set a password for the hacluster user or create a new user which is member of the haclient group.

Do this on every node you will connect to with the HA Web Konsole.

7.1 Starting the HA Web Konsole and Logging In

Procedure 7.1 *Starting the HA Web Konsole*

To use HA Web Konsole, the respective Web service must be started on the node that you want to connect to with the Web interface. For communication, the standard HTTP(s) protocol and port `7630` is used.

1. On the node you want to connect to, open a shell and log in as `root`.

2. Check the status of the service by entering

   ```
   rchawk status
   ```

3. If the service is not running, start it with

   ```
   rchawk start
   ```

   If you want the HA Web Konsole to start automatically at boot time, execute the following command:

   ```
   chkconfig hawk on
   ```

4. On any machine, start a Web browser and make sure that JavaScript and cookies are enabled.

5. Point it at the IP address or hostname of any cluster node, or the address of any `IPaddr(2)` resource that you may have configured:

   ```
   https://IPaddress:7630/
   ```

   NOTE: Certificate Warning

   Depending on your browser and browser options, you may get a certificate warning when trying to access the URL for the first time. This is because the HA Web Konsole uses a self-signed certificate that is not considered trustworthy per default.

   To proceed anyway, you can add an exception in the browser to bypass the warning. To avoid the warning in the first place, the self-signed certificate can also be replaced with a certificate signed by an official Certificate Authority. For information on how to do so, refer to Replacing the Self-Signed Certificate (page 117).

6 On the HA Web Konsole login screen, enter the *Username* and *Password* of the hacluster user (or of any other user that is member of the haclient group) and click *Log In*.

The *Cluster Status* screen appears, displaying the status of your cluster nodes and resources similar to the output of the crm_mon.

7.2 Using HA Web Konsole

After logging in, HA Web Konsole displays the most important global cluster parameters and the status of your cluster nodes and resources. The following color code is used for status display:

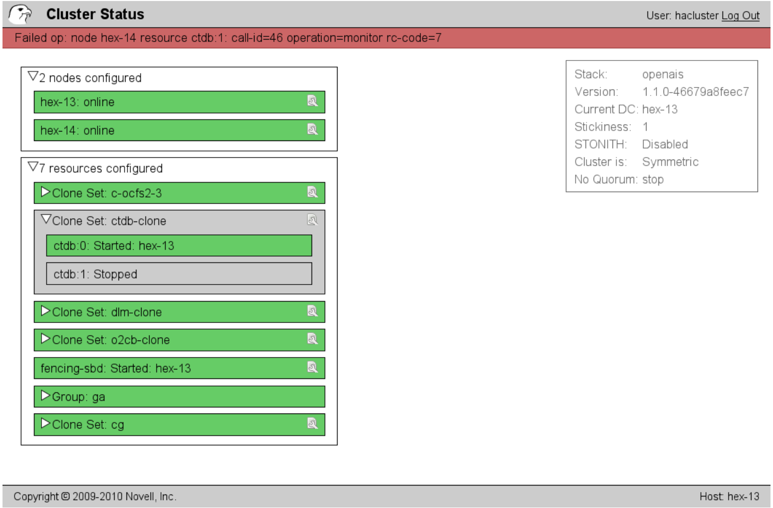
- Green: OK. For example, the resource is running or the node is online.

- Red: Bad, unclean. For example, the resource has failed or the node was not shut down cleanly.

- Yellow: In transition. For example, the node is currently being shut down.

- Grey: Not running, but the cluster expects it to be running. For example, nodes that the administrator has stopped or put into standby mode. Also nodes that are offline are displayed in grey (if they have been shut down cleanly).

Figure 7.1 *HA Web Konsole—Cluster Status*



Cluster Status

User: hacluster Log Out

Failed op: node hex-14 resource ctdb:1: call-id=46 operation=monitor rc-code=7

▽2 nodes configured

hex-13: online

hex-14: online

▽7 resources configured

▷Clone Set: c-ocfs2-3

▽Clone Set: ctdb-clone

ctdb:0: Started: hex-13

ctdb:1: Stopped

▷Clone Set: dlm-clone

▷Clone Set: o2cb-clone

fencing-sbd: Started: hex-13

▷Group: ga

▷Clone Set: cg

Stack: openais
Version: 1.1.0-46679a8feec7
Current DC: hex-13
Stickiness: 1
STONITH: Disabled
Cluster is: Symmetric
No Quorum: stop

Copyright © 2009-2010 Novell, Inc.

Host: hex-13

Click the arrow symbols in the *Nodes* and *Resources* groups to expand and collapse the tree view.

If a resource has failed, a failure message with the details is shown in red at the top of the screen.

Click the wrench icon at the right side of a node or resource to access a context menu that allows some actions, like starting, stopping or cleaning up a resource (or putting a node into `online` or `standby` mode or to fence a node).

Currently, the HA Web Konsole only allows basic operator tasks but more functions will be added in the future, for example, the ability to configure resources and nodes.

7.3 Troubleshooting

HA Web Konsole Log Files

> Find the HA Web Konsole log files in `/srv/www/hawk/log`. It is useful to check them in case you cannot access the HA Web Konsole at all for some reason.
>
> If you have trouble starting or stopping a resource with the HA Web Konsole, check the log files that Pacemaker logs to—by default, `/var/log/messages`) .

Authentication Fails

If you cannot log in to HA Web Konsole with a new user you added to the `haclient` group (or if you experience delays until HA Web Konsole accepts logins from this user), stop the `rcnscd` daemon with `rcnscd stop` and try again.

Replacing the Self-Signed Certificate

To avoid the warning about the self-signed certificate on first startup of the HA Web Konsole, replace the automatically created certificate with your own certificate or a certificate that was signed by an official Certificate Authority (CA).

The certificate is stored in `/etc/lighttpd/certs/hawk-combined.pem` and contains both key and certificate. After you have created or received your new key and certificate, combine them by executing the following command:

```
cat keyfile certificationfile > /etc/lighttpd/certs/hawk-combined.pem
```

Change the permissions to make the file only accessible by `root`:

```
chown root.root /etc/lighttpd/certs/hawk-combined.pem
chmod 600 /etc/lighttpd/certs/hawk-combined.pem
```

Adding or Modifying Resource Agents

8

All tasks that need to be managed by a cluster must be available as a resource. There are two major groups here to consider: resource agents and STONITH agents. For both categories, you can add your own agents, extending the abilities of the cluster to your own needs.

8.1 STONITH Agents

A cluster sometimes detects that one of the nodes is behaving strangely and needs to remove it. This is called *fencing* and is commonly done with a STONITH resource. All STONITH resources reside in `/usr/lib/stonith/plugins` on each node.

WARNING: SSH and STONITH Are Not Supported

It is impossible to know how SSH might react to other system problems. For this reason, SSH and STONITH agent are not supported for production environments.

To get a list of all currently available STONITH devices (from the software side), use the command `stonith -L`.

As of yet there is no documentation about writing STONITH agents. If you want to write new STONITH agents, consult the examples available in the source of the `heartbeat-common` package.

8.2 Writing OCF Resource Agents

All OCF resource agents (RAs) are available in `/usr/lib/ocf/resource.d/`, see Section 4.2.2, “Supported Resource Agent Classes” (page 38) for more information. Each resource agent must supported the following operations to control it:

`start`
: start or enable the resource

`stop`
: stop or disable the resource

`status`
: returns the status of the resource

`monitor`
: similar to `status`, but checks also for unexpected states

`validate`
: validate the resource's configuraton

`meta-data`
: returns information about the resource agent in XML

The general procedure of how to create a OCF RA is like the following:

1 Load the file `/usr/lib/ocf/resource.d/pacemaker/Dummy` as a template.

2 Create a new subdirectory for each new resource agents to avoid naming contradictions. For example, if you have a resource group `kitchen` with the resource `coffee_machine`, add this resource to the directory `/usr/lib/ocf/resource.d/kitchen/`. To access this RA, execute the command `crm`:

```
configure
primitive coffee_1 ocf:coffee_machine:kitchen ...
```

3 Implement the different shell functions and save your file under a different name.

More details about writing OCF resource agents can be found at http://linux-ha.org/wiki/Resource_Agents. Find special information about several concepts at Chapter 1, *Product Overview* (page 3).

8.3 OCF Return Codes and Failure Recovery

According to the OCF specification, there are strict definitions of the exit codes an action must return. The cluster always checks the return code against the expected result. If the result does not match the expected value, then the operation is considered to have failed and a recovery action is initiated. There are three types of failure recovery:

Table 8.1 *Failure Recovery Types*

Recovery Type	Description	Action Taken by the Cluster
soft	A transient error occurred.	Restart the resource or move it to a new location.
hard	A non-transient error occurred. The error may be specific to the current node.	Move the resource elsewhere and prevent it from being retried on the current node.
fatal	A non-transient error occurred that will be common to all cluster nodes. This means a bad configuration was specified.	Stop the resource and prevent it from being started on any cluster node.

Assuming an action is considered to have failed, the following table outlines the different OCF return codes and the type of recovery the cluster will initiate when the respective error code is received.

Table 8.2 OCF Return Codes

OCF Return Code	OCF Alias	Description	Recovery Type
0	OCF_SUCCESS	Success. The command completed successfully. This is the expected result for all start, stop, promote and demote commands.	soft
1	OCF_ERR_GENERIC	Generic “there was a problem” error code.	soft
2	OCF_ERR_ARGS	The resource’s configuration is not valid on this machine (for example, it refers to a location/tool not found on the node).	hard
3	OCF_ERR_UNIMPLEMENTED	The requested action is not implemented.	hard
4	OCF_ERR_PERM	The resource agent does not have sufficient privileges to complete the task.	hard
5	OCF_ERR_INSTALLED	The tools required by the resource are not installed on this machine.	hard
6	OCF_ERR_CONFIGURED	The resource’s configuration is invalid (for example, required parameters are missing).	fatal
7	OCF_NOT_RUNNING	The resource is not running. The cluster will not attempt to stop a resource that returns this for any action. This OCF return code may or may not require resource recovery—it depends on what is the expected resource status. If unexpected, then `soft` recovery.	N/A

OCF Return Code	OCF Alias	Description	Recovery Type
8	OCF_RUNNING_MASTER	The resource is running in Master mode.	soft
9	OCF_FAILED_MASTER	The resource is in Master mode but has failed. The resource will be demoted, stopped and then started (and possibly promoted) again.	soft
other	N/A	Custom error code.	soft

9 Fencing and STONITH

Fencing is a very important concept in computer clusters for HA (High Availability). A cluster sometimes detects that one of the nodes is behaving strangely and needs to remove it. This is called *fencing* and is commonly done with a STONITH resource. Fencing may be defined as a method to bring an HA cluster to a known state.

Every resource in a cluster has a state attached. For example: “resource r1 is started on node1”. In an HA cluster, such a state implies that “resource r1 is stopped on all nodes but node1”, because an HA cluster must make sure that every resource may be started on at most one node. Every node must report every change that happens to a resource. The cluster state is thus a collection of resource states and node states.

If, for whatever reason, a state of some node or resource cannot be established with certainty, fencing comes in. Even when the cluster is not aware of what is happening on a given node, fencing can ensure that the node does not run any important resources.

9.1 Classes of Fencing

There are two classes of fencing: resource level and node level fencing. The latter is the primary subject of this chapter.

Resource Level Fencing

> Using resource level fencing the cluster can ensure that a node cannot access one or more resources. One typical example is a SAN, where a fencing operation changes rules on a SAN switch to deny access from the node.

The resource level fencing may be achieved using normal resources on which the resource you want to protect depends. Such a resource would simply refuse to start on this node and therefore resources which depend on will not run on the same node.

Node Level Fencing
: Node level fencing ensures that a node does not run any resources at all. This is usually done in a very simple, yet abrupt way: the node is reset using a power switch. This is necessary when the node becomes unresponsive.

9.2 Node Level Fencing

In SUSE® Linux Enterprise High Availability Extension, the fencing implementation is STONITH (Shoot The Other Node in the Head). It provides the node level fencing. The High Availability Extension includes the `stonith` command line tool, an extensible interface for remotely powering down a node in the cluster. For an overview of the available options, run `stonith --help` or refer to the man page of `stonith` for more information.

9.2.1 STONITH Devices

To use node level fencing, you first need to have a fencing device. To get a list of STONITH devices which are supported by the High Availability Extension, run the following command as `root` on any of the nodes:

```
stonith -L
```

STONITH devices may be classified into the following categories:

Power Distribution Units (PDU)
: Power Distribution Units are an essential element in managing power capacity and functionality for critical network, server and data center equipment. They can provide remote load monitoring of connected equipment and individual outlet power control for remote power recycling.

Uninterruptible Power Supplies (UPS)
: A stable power supply provides emergency power to connected equipment by supplying power from a separate source in the event of utility power failure.

Blade Power Control Devices

  If you are running a cluster on a set of blades, then the power control device in the blade enclosure is the only candidate for fencing. Of course, this device must be capable of managing single blade computers.

Lights-out Devices

  Lights-out devices (IBM RSA, HP iLO, Dell DRAC) are becoming increasingly popular, and in the future they may even become standard on off-the-shelf computers. However, they are inferior to UPS devices, because they share a power supply with their host (a cluster node). If a node stays without power, the device supposed to control it would be just as useless. In that case, the CRM would continue its attempts to fence the node indefinitely, as all other resource operations would wait for the fencing/STONITH operation to complete.

Testing Devices

  Testing devices are used exclusively for testing purposes. They are usually more gentle on the hardware. Once the cluster goes into production, they must be replaced with real fencing devices.

The choice of the STONITH device depends mainly on your budget and the kind of hardware you use.

9.2.2 STONITH Implementation

The STONITH implementation of SUSE® Linux Enterprise High Availability Extension consists of two components:

stonithd

  stonithd is a daemon which can be accessed by local processes or over the network. It accepts the commands which correspond to fencing operations: reset, power-off, and power-on. It can also check the status of the fencing device.

  The stonithd daemon runs on every node in the CRM HA cluster. The stonithd instance running on the DC node receives a fencing request from the CRM. It is up to this and other stonithd programs to carry out the desired fencing operation.

STONITH Plug-ins

  For every supported fencing device there is a STONITH plug-in which is capable of controlling said device. A STONITH plug-in is the interface to the fencing device.

On each node, all STONITH plug-ins reside in /usr/lib/stonith/plugins (or in /usr/lib64/stonith/plugins, respectively for 64-bit architectures). All STONITH plug-ins look the same to stonithd, but are quite different on the other side reflecting the nature of the fencing device.

Some plug-ins support more than one device. A typical example is ipmilan (or external/ipmi) which implements the IPMI protocol and can control any device which supports this protocol.

9.3 STONITH Configuration

To set up fencing, you need to configure one or more STONITH resources—the stonithd daemon requires no configuration. All configuration is stored in the CIB. A STONITH resource is a resource of class stonith (see Section 4.2.2, “Supported Resource Agent Classes” (page 38)). STONITH resources are a representation of STONITH plug-ins in the CIB. Apart from the fencing operations, the STONITH resources can be started, stopped and monitored, just like any other resource. Starting or stopping STONITH resources means enabling and disabling STONITH in this case. Starting and stopping are thus only administrative operations, and do not translate to any operation on the fencing device itself. However, monitoring does translate to device status.

STONITH resources can be configured just like any other resource. For more information about configuring resources, see Section 5.3.2, “Creating STONITH Resources” (page 65) or Section 6.3.3, “Creating a STONITH Resource” (page 101).

The list of parameters (attributes) depends on the respective STONITH type. To view a list of parameters for a specific device, use the stonith command:

```
stonith -t stonith-device-type -n
```

For example, to view the parameters for the ibmhmc device type, enter the following:

```
stonith -t ibmhmc -n
```

To get a short help text for the device, use the -h option:

```
stonith -t stonith-device-type -h
```

9.3.1 Example STONITH Resource Configurations

In the following, find some example configurations written in the syntax of the `crm` command line tool. To apply them, put the sample in a text file (for example, `sample.txt`) and run:

```
crm < sample.txt
```

For more information about configuring resources with the `crm` command line tool, refer to Chapter 6, *Configuring and Managing Cluster Resources (Command Line)* (page 89).

WARNING: Testing Configurations

Some of the examples below are for demonstration and testing purposes only. Do not use any of the `Testing Configuration` examples in real-life cluster scenarios.

Example 9.1 *Testing Configuration*

```
configure
primitive st-null stonith:null \
params hostlist="node1 node2"
clone fencing st-null
commit
```

Example 9.2 *Testing Configuration*

An alternative configuration:

```
configure
 primitive st-node1 stonith:null \
 params hostlist="node1"
 primitive st-node2 stonith:null \
 params hostlist="node2"
 location l-st-node1 st-node1 -inf: node1
 location l-st-node2 st-node2 -inf: node2
 commit
```

This configuration example is perfectly alright as far as the cluster software is concerned. The only difference to a real world configuration is that no fencing operation takes place.

Example 9.3 *Testing Configuration*

A more realistic example (but still only for testing) is the following external/ssh configuration:

```
configure
 primitive st-ssh stonith:external/ssh \
 params hostlist="node1 node2"
 clone fencing st-ssh
 commit
```

This one can also reset nodes. The configuration is remarkably similar to the first one which features the null STONITH device. In this example, clones are used. They are a CRM/Pacemaker feature. A clone is basically a shortcut: instead of defining `n` identical, yet differently-named resources, a single cloned resource suffices. By far the most common use of clones is with STONITH resources, as long as the STONITH device is accessible from all nodes.

Example 9.4 *Configuration of an IBM RSA Lights-out Device*

The real device configuration is not much different, though some devices may require more attributes. An IBM RSA lights-out device might be configured like this:

```
configure
primitive st-ibmrsa-1 stonith:external/ibmrsa-telnet \
params nodename=node1 ipaddr=192.168.0.101 \
userid=USERID passwd=PASSW0RD
primitive st-ibmrsa-2 stonith:external/ibmrsa-telnet \
params nodename=node2 ipaddr=192.168.0.102 \
userid=USERID passwd=PASSW0RD
location l-st-node1 st-ibmrsa-1 -inf: node1
location l-st-node2 st-ibmrsa-2 -inf: node2
commit
```

In this example, location constraints are used because of the following reason: There is always a certain probability that the STONITH operation is going to fail. Therefore, a STONITH operation (on the node which is the executioner, as well) is not reliable. If the node is reset, then it cannot send the notification about the fencing operation outcome. The only way to do that is to assume that the operation is going to succeed and send the notification beforehand. But if the operation fails, problems could arise. Therefore, by convention, stonithd refuses to kill its host.

Example 9.5 *Configuration of an UPS Fencing Device*

The configuration of a UPS type of fencing device is similar to the examples above. The details are left (as an exercise) to the reader. All UPS devices employ the same mechanics for fencing, but how the device itself is accessed varies. Old UPS devices used to have just a serial port, in most cases connected at 1200baud using a special serial cable. Many new ones still have a serial port, but often they also utilize a USB or ethernet interface. The kind of connection you can use is dependent on what the plug-in supports.

For example, compare the `apcmaster` with the `apcsmart` device by using the `stonith -t` *`stonith-device-type`* `-n` command:

```
stonith -t apcmaster -h
```

returns the following information:

```
STONITH Device: apcmaster - APC MasterSwitch (via telnet)
NOTE: The APC MasterSwitch accepts only one (telnet)
connection/session a time. When one session is active,
subsequent attempts to connect to the MasterSwitch will fail.
For more information see http://www.apc.com/
List of valid parameter names for apcmaster STONITH device:
ipaddr
login
 password
```

With

```
stonith -t apcsmart -h
```

you get the following output:

```
STONITH Device: apcsmart - APC Smart UPS
(via serial port - NOT USB!).
Works with higher-end APC UPSes, like
Back-UPS Pro, Smart-UPS, Matrix-UPS, etc.
(Smart-UPS may have to be >= Smart-UPS 700?).
See http://www.networkupstools.org/protocols/apcsmart.html
for protocol compatibility details.
For more information see http://www.apc.com/
List of valid parameter names for apcsmart STONITH device:
ttydev
hostlist
```

The first plug-in supports APC UPS with a network port and telnet protocol. The second plug-in uses the APC SMART protocol over the serial line, which is supported by many different APC UPS product lines.

9.3.2 Constraints Versus Clones

In Section 9.3.1, “Example STONITH Resource Configurations” (page 129) you learned that there are several ways to configure a STONITH resource: using constraints, clones, or both. The choice of which construct to use for configuration depends on several factors (nature of the fencing device, number of hosts managed by the device, number of cluster nodes, or personal preference).

In short: if clones are safe to use with your configuration and if they reduce the configuration, then use cloned STONITH resources.

9.4 Monitoring Fencing Devices

Just like any other resource, the STONITH class agents also support the monitoring operation which is used for checking status.

NOTE: Monitoring STONITH Resources

Monitoring STONITH resources is strongly recommended. Monitor them regularly, yet sparingly.

Fencing devices are an indispensable part of an HA cluster, but the less you need to utilize them, the better. Power management equipment is known to be rather fragile on the communication side. Some devices give up if there is too much broadcast traffic. Some cannot handle more than ten or so connections per minute. Some get confused if two clients try to connect at the same time. Most cannot handle more than one session at a time.

Checking the fencing devices once every couple of hours should be enough in most cases. The probability that within those few hours there will be a need for a fencing operation and that the power switch would fail is usually low.

For detailed information on how to configure monitor operations, refer to Procedure 5.3, “Adding or Modifying Meta and Instance Attributes” (page 64) for the GUI approach or to Section 6.3.8, “Configuring Resource Monitoring” (page 107) for the command line approach.

9.5 Special Fencing Devices

Apart from plug-ins which handle real STONITH devices, some STONITH plug-ins require additional explanation.

WARNING: For Testing Only

Some of the STONITH plug-ins mentioned below are for demonstration and testing purposes only. Do not use any of following devices in real-life scenarios because this may lead to data corruption and unpredictable results:

- `external/ssh`
- `ssh`
- `null`

`external/kdumpcheck`
This plug-in is useful for checking if a kernel dump is in progress on a node. If that is the case, it will return `true`, as if the node has been fenced (it cannot run any resources at that time). This avoids fencing a node that is already down but doing a dump, which takes some time. The plug-in must be used in concert with another, real STONITH device. For more details, see `/usr/share/doc/packages/cluster-glue/README_kdumpcheck.txt`.

`external/sbd`
This is a self-fencing device. It reacts to a so-called “poison pill” which can be inserted into a shared disk. On shared-storage connection loss, it also makes the node cease to operate. Learn how to use this STONITH agent to implement storage based fencing in Chapter 15, *Storage Protection* (page 189). See also `http://www.linux-ha.org/wiki/SBD_Fencing` for more details.

`external/ssh`
Another software-based “fencing” mechanism. The nodes must be able to log in to each other as `root` without passwords. It takes a single parameter, `hostlist`, specifying the nodes that it will target. As it is not able to reset a truly failed node, it must not be used for real-life clusters—for testing and demonstration purposes only. Using it for shared storage would result in data corruption.

`meatware`

`meatware` requires help from a human to operate. Whenever invoked, `meatware` logs a CRIT severity message which shows up on the node's console. The operator then confirms that the node is down and issue a `meatclient(8)` command. This tells `meatware` that it can inform the cluster that it may consider the node dead. See `/usr/share/doc/packages/cluster-glue/README.meatware` for more information.

`null`

This is an imaginary device used in various testing scenarios. It always behaves as if and claims that it has shot a node, but never does anything. Do not use it unless you know what you are doing.

`suicide`

This is a software-only device, which can reboot a node it is running on, using the `reboot` command. This requires action by the node's operating system and can fail under certain circumstances. Therefore avoid using this device whenever possible. However, it is safe to use on one-node clusters.

`suicide` and `null` are the only exceptions to the “do not shoot my host” rule.

9.6 For More Information

`/usr/share/doc/packages/cluster-glue`

In your installed system, this directory holds README files for many STONITH plug-ins and devices.

http://www.linux-ha.org/wiki/STONITH

Information about STONITH on the home page of the The High Availability Linux Project.

http://www.clusterlabs.org/doc/crm_fencing.html

Information about fencing on the home page of the Pacemaker Project.

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

Explains the concepts used to configure Pacemaker. Contains comprehensive and very detailed information for reference.

`http://techthoughts.typepad.com/managing_computers/2007/10/split-brain-quo.html`

Article explaining the concepts of split brain, quorum and fencing in HA clusters.

Load Balancing with Linux Virtual Server 10

The goal of Linux Virtual Server (LVS) is to provide a basic framework that directs network connections to multiple servers that share their workload. Linux Virtual Server is a cluster of servers (one or more load balancers and several real servers for running services) which appears to be one large, fast server to an outside client. This apparent single server is called a *virtual server*. The Linux Virtual Server can be used to build highly scalable and highly available network services, such as Web, cache, mail, FTP, media and VoIP services.

The real servers and the load balancers may be interconnected by either high-speed LAN or by geographically dispersed WAN. The load balancers can dispatch requests to the different servers. They make parallel services of the cluster appear as a virtual service on a single IP address (the virtual IP address or VIP). Request dispatching can use IP load balancing technologies or application-level load balancing technologies. Scalability of the system is achieved by transparently adding or removing nodes in the cluster. High availability is provided by detecting node or daemon failures and reconfiguring the system appropriately.

10.1 Conceptual Overview

The following sections give an overview of the main LVS components and concepts.

10.1.1 Director

The main component of LVS is the ip_vs (or IPVS) kernel code. It implements transport-layer load balancing inside the Linux kernel (layer-4 switching). The node that runs a Linux kernel including the IPVS code is called *director*. The IPVS code running on the director is the essential feature of LVS.

When clients connect to the director, the incoming requests are load-balanced across all cluster nodes: The director forwards packets to the real servers, using a modified set of routing rules that make the LVS work. For example, connections do not originate or terminate on the director, it does not send acknowledgments. The director acts as a specialized router that forwards packets from end-users to real servers (the hosts that run the applications that process the requests).

By default, the kernel does not have the IPVS module installed. The IPVS kernel module is included in the `cluster-network-kmp-default` package.

10.1.2 User Space Controller and Daemons

The `ldirectord` daemon is a user-space daemon for managing Linux Virtual Server and monitoring the real servers in an LVS cluster of load balanced virtual servers. A configuration file, `/etc/ha.d/ldirectord.cf`, specifies the virtual services and their associated real servers and tells `ldirecord` how to configure the server as a LVS redirector. When the daemon is initialized, it creates the virtual services for the cluster.

By periodically requesting a known URL and checking the responses, the `ldirectord` daemon monitors the health of the real servers. If a real server fails, it will be removed from the available server list at the load balancer. When the service monitor detects that the dead server has recovered and is working again, it will add the server back to the available server list. For the case that all real servers should be down, a fall-back server can be specified to which to redirect a Web service. Typically the fall-back server is localhost, presenting an emergency page about the Web service being temporarily unavailable.

10.1.3 Packet Forwarding

There are three different methods of how the director can send packets from the client to the real servers:

Network Address Translation (NAT)
: Incoming requests arrive at the virtual IP and are forwarded to the real servers by changing the destination IP address and port to that of the chosen real server. The real server sends the response to the load balancer which in turn changes the destination IP address and forwards the response back to the client, so that the end user receives the replies from the expected source. As all traffic goes through the load balancer, it usually becomes a bottleneck for the cluster.

IP Tunneling (IP-IP Encapsulation)
: IP tunneling enables packets addressed to an IP address to be redirected to another address, possibly on a different network. The LVS sends requests to real servers through an IP tunnel (redirecting to a different IP address) and the real servers reply directly to the client using their own routing tables. Cluster members can be in different subnets.

Direct Routing
: Packets from end users are forwarded directly to the real server. The IP packet is not modified, so the real servers must be configured to accept traffic for the virtual server's IP address. The response from the real server is sent directly to the client. The real servers and load balancers have to be in the same physical network segment.

10.1.4 Scheduling Algorithms

Deciding which real server to use for a new connection requested by a client is implemented using different algorithms. They are available as modules and can be adapted to specific needs. For an overview of available modules, refer to the `ipvsadm(8)` man page. Upon receiving a connect request from a client, the director assigns a real server to the client based on a *schedule*. The scheduler is the part of the IPVS kernel code which decides which real server will get the next new connection.

10.2 Configuring IP Load Balancing with YaST

You can configure kernel-based IP load balancing with the YaST iplb module. It is a front-end for `ldirectord`.

To access the IP Load Balancing dialog, start YaST as `root` and select *High Availability* > *IP Load Balancing*. Alternatively, start the YaST cluster module as `root` on a command line with `yast2 iplb`.

The YaST module writes its configuration to `/etc/ha.d/ldirectord.cf`. The tabs available in the YaST module correspond to the structure of the `/etc/ha.d/ldirectord.cf` configuration file, defining global options and defining the options for the virtual services.

For an example configuration and the resulting processes between load balancers and real servers, refer to Example 10.1, “Simple ldirectord Configuration” (page 145).

NOTE: Global Parameters and Virtual Server Parameters

If a certain parameter is specified in both the virtual server section and in the global section, the value defined in the virtual server section overrides the value defined in the global section.

Procedure 10.1 *Configuring Global Parameters*

The following procedure describes how to configure the most important global parameters. For more details about the individual parameters (and the parameters not covered here), click *Help* or refer to the `ldirectord` man page.

1. With *Check Interval*, define the interval in which `ldirectord` will connect to each of the real servers to check if they are still online.

2. With *Check Timeout*, set the time in which the real server should have responded after the last check.

3. With *Check Count* you can define how many times `ldirectord` will attempt to request the real servers until the check is considered as failed.

4. With *Negotiate Timeout* define a timeout in seconds for negotiate checks.

5. In *Fallback*, enter the hostname or IP address of the Web server onto which to redirect a Web service in case all real servers are down.

6. If you want to use an alternative path for logging, specify a path for the logs in *Log File*. By default, `ldirectord` writes its logs to `/var/log/ldirectord.log`.

7. If you want the system to send alerts in case the connection status to any real server changes, enter a valid e-mail address in *Email Alert*.

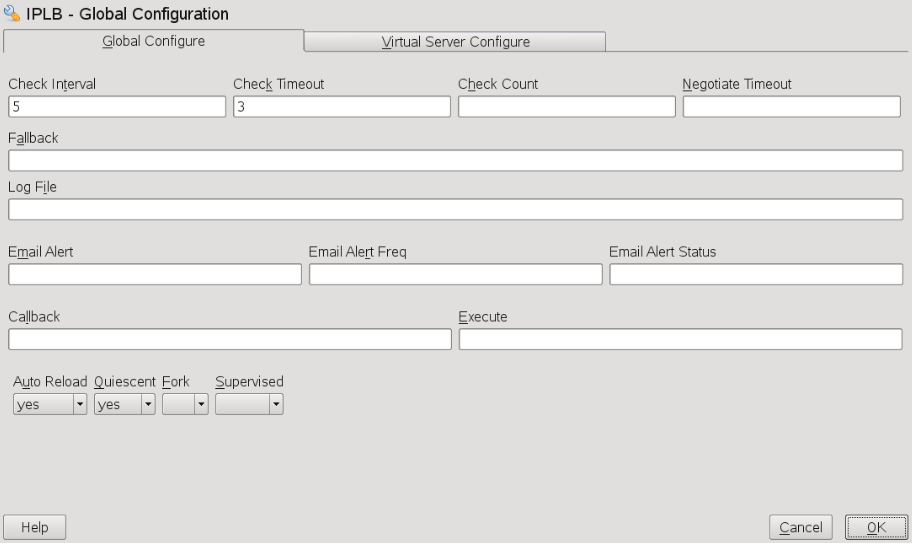
8. With *Email Alert Frequency*, define after how many seconds the e-mail alert should be repeated if any of the real servers remains inaccessible.

9. In *Email Alert Status* specify the server states for which email alerts should be sent. If you want to define more than one state, use a comma-separated list.

10. With *Auto Reload* define, if `ldirectord` should continuously monitor the configuration file for modification. If set to `yes`, the configuration is automatically reloaded upon changes.

11. With the *Quiescent* switch, define if to remove failed real servers from the kernel's LVS table or not. If set to *Yes*, failed servers are not removed. Instead their weight is set to `0` which means that no new connections will be accepted. Already established connections will persist until they time out.

Figure 10.1 YaST IP Load Balancing—Global Parameters



Check Interval: 5

Check Timeout: 3

Auto Reload: yes

Quiescent: yes

Procedure 10.2 Configuring Virtual Services

You can configure one or more virtual services by defining a couple of parameters for each. The following procedure describes how to configure the most important parameters for a virtual service. For more details about the individual parameters (and the parameters not covered here), click *Help* or refer to the `ldirectord` man page.

1. In the YaST iplb module, switch to the *Virtual Server Configuration* tab.

2. *Add* a new virtual server or *Edit* an existing virtual server. A new dialog shows the available options.

3. In *Virtual Server* enter the shared virtual IP address and port under which the load balancers and the real servers are accessible as LVS. Instead of IP address and port name, you can also specify a hostname and a service. Alternatively, you can also use a firewall mark. A firewall mark is a way of aggregating an arbitrary collection of `VIP:port` services into one virtual service.

4. To specify the *Real Servers*, you need to enter the IP address (or hostnames) of the servers, the ports (or service names) and the forwarding method. The forward-

ing method must either be `gate`, `ipip` or `masq`, see Section 10.1.3, "Packet Forwarding" (page 139).

Click the *Add* button and enter the required arguments for each real server.

5 As *Check Type*, select the type of check that should be performed to test if the real servers are still alive. For example, to send a request and check if the response contains an expected string, select `Negotiate`.

6 If you have set the *Check Type* to `Negotiate`, you also need to define the type of service to monitor. Select it from the *Service* drop-down list.

7 In *Request*, enter the URI to the object that is requested on each real server during the check intervals.

8 If you want to check if the response from the real servers contains a certain string ("I'm alive" message), define a regular expression that needs to be matched. Enter the regular expression into *Receive*. If the response from a real server contains this expression, the real server is considered to be alive.

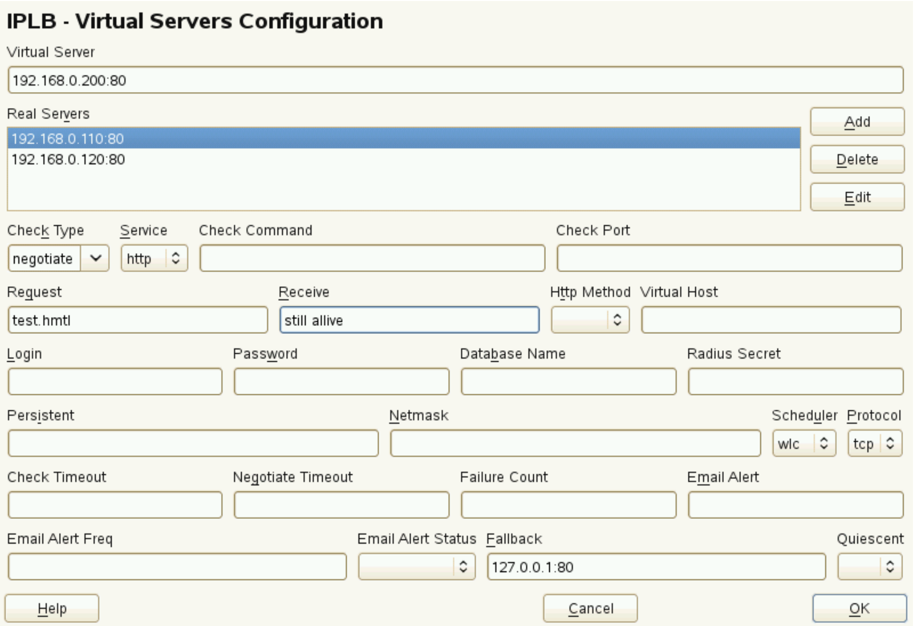
9 Depending on the type of *Service* you have selected in Step 6 (page 143), you also need to specify further parameters like *Login*, *Password*, *Database*, or *Secret*. For more information, refer to the YaST help text or to the `ldirectord` man page.

10 Select the *Scheduler* to be used for load balancing. For information on the available schedulers, refer to the `ipvsadm(8)` man page.

11 Select the *Protocol* to be used. If the virtual service is specified as an IP address and port, it must be either `tcp` or `udp`. If the virtual service is specified as a firewall mark, the protocol must be `fwm`.

12 Define further parameters, if needed. Confirm your configuration with *OK*. YaST writes the configuration to `/etc/ha.d/ldirectord.cf`.

Figure 10.2 *YaST IP Load Balancing—Virtual Services*



IPLB - Virtual Servers Configuration

Virtual Server: 192.168.0.200:80

Real Servers:
192.168.0.110:80
192.168.0.120:80

Check Type: negotiate

Service: http

Request: test.hmtl

Receive: still allive

Scheduler: wlc

Protocol: tcp

Fallback: 127.0.0.1:80

Example 10.1 *Simple ldirectord Configuration*

The values shown in Figure 10.1, “YaST IP Load Balancing—Global Parameters” (page 142) and Figure 10.2, “YaST IP Load Balancing—Virtual Services” (page 144), would lead to the following configuration, defined in `/etc/ha.d/ldirectord.cf`:

```
autoreload = yes ❶
checkinterval = 5 ❷
checktimeout = 3 ❸
quiescent = yes ❹
    virtual = 192.168.0.200:80 ❺
    checktype = negotiate ❻
    fallback = 127.0.0.1:80 ❼
    protocol = tcp ❽
    real = 192.168.0.110:80 gate ❾
    real = 192.168.0.120:80 gate ❾
    receive = "still alive" ❿
    request = "test.html"
    scheduler = wlc
    service = http
```

❶ Defines that `ldirectord` should continuously check the configuration file for modification.

❷ Interval in which `ldirectord` will connect to each of the real servers to check if they are still online.

❸ Time in which the real server should have responded after the last check.

❹ Defines not to remove failed real servers from the kernel's LVS table, but to set their weight to `0` instead.

❺ Virtual IP address (VIP) of the LVS. The LVS is available at port `80`.

❻ Type of check that should be performed to test if the real servers are still alive.

❼ Server onto which to redirect a Web service all real servers for this service are down.

❽ Protocol to be used.

❾ Two real servers defined, both available at port `80`. The packet forwarding method is `gate`, meaning that direct routing is used.

❿ Regular expression that needs to be matched in the response string from the real server.

URI to the object that is requested on each real server during the check intervals.

Selected scheduler to be used for load balancing.

Type of service to monitor.

This configuration would lead to the following process flow: The `ldirectord` will connect to each real server once every 5 seconds ❷ and request `192.168.0.110:80/test.html` or `192.168.0.120:80/test.html` as specified in ❾ and . If it does not receive the expected `still alive` string ❿ from a real server within 3 seconds ❸ of the last check, it will remove the real server from the available pool. However, because of the `quiescent=yes` setting ❹, the real server will not be removed from the LVS table, but its weight will be set to `0` so that no new connections to this real server will be accepted. Already established connections will be persistent until they time out.

10.3 Further Setup

Apart from the configuration of `ldirectord` with YaST, you need to make sure the following conditions are fulfilled to complete the LVS setup:

- The real servers are set up correctly to provide the needed services.

- The load balancing server (or servers) must be able to route traffic to the real servers using IP forwarding. The network configuration of the real servers depends on which packet forwarding method you have chosen.

- To prevent the load balancing server (or servers) from becoming a single point of failure for the whole system, you need to set up one or several backups of the load balancer. In the cluster configuration, configure a primitive resource for `ldirectord`, so that `ldirectord` can fail over to other servers in case of hardware failure.

- As the backup of the load balancer also needs the `ldirectord` configuration file to fulfill its task, make sure the `/etc/ha.d/ldirectord.cf` is available on all servers that you want to use as backup for the load balancer. You can synchronize the configuration file with Csync2 as described in Section 3.2.3, “Transferring the Configuration to All Nodes” (page 26).

10.4 For More Information

To learn more about Linux Virtual Server, refer to the project home page available at `http://www.linuxvirtualserver.org/`.

For more information about `ldirectord`, refer to its comprehensive man page.

Network Device Bonding 11

For many systems, it is desirable to implement network connections that comply to more than the standard data security or availability requirements of a typical ethernet device. In these cases, several ethernet devices can be aggregated to a single bonding device.

The configuration of the bonding device is done by means of bonding module options. The behavior is determined through the mode of the bonding device. By default, this is `mode=active-backup`, which means that a different slave device will become active if the active slave fails.

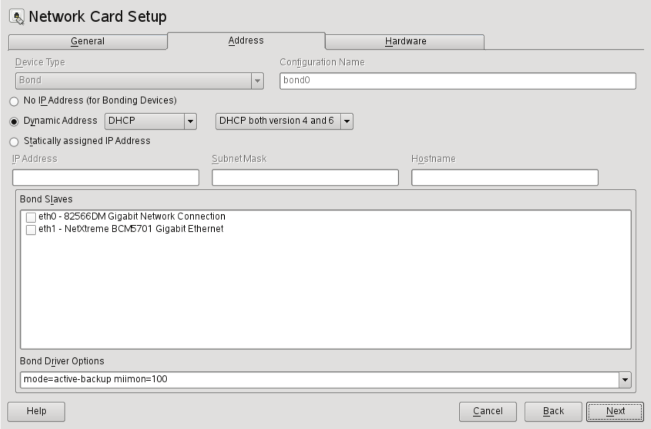
When using OpenAIS, the bonding device is not managed by the cluster software. Therefore, the bonding device must be configured on each cluster node that might possibly need to access the bonding device.

11.1 Configuring Bonding Devices with YaST

To configure a bonding device, use the following procedure:

1 Start YaST as `root` and select *Network Devices* > *Network Settings*.

2 Click *Add* to configure a new network card and change the *Device Type* to *Bond*. Proceed with *Next*.



Device Type: Bond

Configuration Name: bond0

○ No IP Address (for Bonding Devices)

● Dynamic Address DHCP — DHCP both version 4 and 6

○ Statically assigned IP Address

IP Address | Subnet Mask | Hostname

Bond Slaves

□ eth0 - 82566DM Gigabit Network Connection

□ eth1 - NetXtreme BCM5701 Gigabit Ethernet

Bond Driver Options

mode=active-backup miimon=100

3 Select how to assign the IP address to the bonding device. Three methods are at your disposal:

- No IP Address
- Dynamic Address (with DHCP or Zeroconf)
- Statically assigned IP Address

Use the method that is appropriate for your environment. If OpenAIS manages virtual IP addresses, select *Statically assigned IP Address* and assign a basic IP address to the interface.

4 Switch to the *Bond Slaves* tab.

5 To select the ethernet devices that need to be included into the bond, activate the check box in front of the relevant *Bond Slave*.

6 Edit the *Bond Driver Options*. The following modes are available:

`balance-rr`
: Provides load balancing and fault tolerance.

`active-backup`
: Provides fault tolerance.

`balance-xor`
: Provides load balancing and fault tolerance.

`broadcast`
: Provides fault tolerance

`802.3ad`
: Provides dynamic link aggregation if supported by the connected switch.

`balance-tlb`
: Provides load balancing for outgoing traffic.

`balance-alb`
: Provides load balancing for incoming and outgoing traffic, if the network devices used allow the modifying of the network device's hardware address while in use.

7 Make sure that the parameter `miimon=100` is added to *Bond Driver Options*. Without this parameter, the data integrity is not checked regularly.

8 Click *Next* and leave YaST with *Ok* to create the device.

11.2 For More Information

All modes, as well as many other options, are explained in detail in the *Linux Ethernet Bonding Driver HOWTO*, which can be found at `/usr/src/linux/Documentation/networking/bonding.txt` once you have installed the package `kernel-source`.

Part III. Storage and Data Replication

Oracle Cluster File System 2 12

Oracle Cluster File System 2 (OCFS2) is a general-purpose journaling file system that has been fully integrated since the Linux 2.6 kernel. OCFS2 allows you to store application binary files, data files, and databases on devices on shared storage. All nodes in a cluster have concurrent read and write access to the file system. A user-space control daemon, managed via a clone resource, provides the integration with the HA stack, in particular with OpenAIS/Corosync and the Distributed Lock Manager (DLM).

12.1 Features and Benefits

OCFS2 can be used for the following storage solutions for example:

- General applications and workloads.

- Xen image store in a cluster. Xen virtual machines and virtual servers can be stored on OCFS2 volumes that are mounted by cluster servers. This provides quick and easy portability of Xen virtual machines between servers.

- LAMP (Linux, Apache, MySQL, and PHP | Perl | Python) stacks.

As a high-performance, symmetric and parallel cluster file system, OCFS2 supports the following functions:

- An application's files are available to all nodes in the cluster. Users simply install it once on an OCFS2 volume in the cluster.

- All nodes can concurrently read and write directly to storage via the standard file system interface, enabling easy management of applications that run across the cluster.

- File access is coordinated through DLM. DLM control is good for most cases, but an application's design might limit scalability if it contends with the DLM to coordinate file access.

- Storage backup functionality is available on all back-end storage. An image of the shared application files can be easily created, which can help provide effective disaster recovery.

OCFS2 also provides the following capabilities:

- Metadata caching.

- Metadata journaling.

- Cross-node file data consistency.

- Support for multiple-block sizes up to 4 KB, cluster sizes up to 1 MB, for a maximum volume size of 4 PB (Petabyte).

- Support for up to 32 cluster nodes.

- Asynchronous and direct I/O support for database files for improved database performance.

12.2 OCFS2 Packages and Management Utilities

The OCFS2 kernel module (`ocfs2`) is installed automatically in the High Availability Extension on SUSE® Linux Enterprise Server 11 SP1. To use OCFS2, make sure the following packages are installed on each node in the cluster: `ocfs2-tools` and the matching `ocfs2-kmp-*` packages for your kernel.

The `ocfs2-tools` package provides the following utilities for management of OFS2 volumes. For syntax information, see their man pages.

Table 12.1 OCFS2 Utilities

OCFS2 Utility	Description
debugfs.ocfs2	Examines the state of the OCFS file system for the purpose of debugging.
fsck.ocfs2	Checks the file system for errors and optionally repairs errors.
mkfs.ocfs2	Creates an OCFS2 file system on a device, usually a partition on a shared physical or logical disk.
mounted.ocfs2	Detects and lists all OCFS2 volumes on a clustered system. Detects and lists all nodes on the system that have mounted an OCFS2 device or lists all OCFS2 devices.
tunefs.ocfs2	Changes OCFS2 file system parameters, including the volume label, number of node slots, journal size for all node slots, and volume size.

12.3 Configuring OCFS2 Services and a STONITH Resource

Before you can create OCFS2 volumes, you must configure the following resources as services in the cluster: DLM, O2CB and a STONITH resource. OCFS2 uses the cluster membership services from Pacemaker which run in user space. Therefore, DLM and O2CB need to be configured as clone resources that are present on each node in the cluster.

The following procedure uses the `crm` shell to configure the cluster resources. Alternatively, you can also use the Pacemaker GUI to configure the resources.

NOTE: DLM Resource for Both cLVM and OCFS2

Both cLVM and OCFS2 need a DLM resource that runs on all nodes in the cluster and therefore usually is configured as a clone. If you have a setup that includes both OCFS2 and cLVM, configuring *one* DLM resource for both OCFS2 and cLVM is enough.

Procedure 12.1 *Configuring DLM and O2CB Resources*

The configuration consists of a base group that includes several primitives and a base clone. Both base group and base clone can be used in various scenarios afterwards (for both OCFS2 and cLVM, for example)—you only need to extended the base group with the respective primitives as needed. As the base group has internal collocation and ordering, this facilitates the overall setup as you do not have to specify lots of individual groups, clones and their dependencies.

Follow the steps below for one node in the cluster:

1 Start a shell and log in as `root` or equivalent.

2 Run `crm configure`.

3 Enter the following to create the primitive resources for DLM and O2CB:

```
primitive dlm ocf:pacemaker:controld \
      op monitor interval="60" timeout="60"
primitive o2cb ocf:ocfs2:o2cb \
      op monitor interval="60" timeout="60"
```

The `dlm` clone resource controls the distributed lock manager service, and makes sure this service is started on all nodes in the cluster. Due to the base group's internal collocation and ordering, the `o2cb` service is only started on nodes that also have a copy of the `dlm` service already running.

4 Enter the following to create a base group and a base clone:

```
group base-group dlm o2cb
clone base-clone base-group \
      meta interleave="true"
```

5 Review your changes with `show`.

6 If everything is correct, submit your changes with `commit` and leave the crm live configuration with `exit`.

Procedure 12.2 *Configuring a STONITH Resource*

NOTE: STONITH Device Needed

You need to configure a fencing device. Without a STONITH mechanism (like `external/sbd`) in place the configuration will fail.

1 Start a shell and log in as `root` or equivalent.

2 Run `crm configure`.

3 Configure `external/sdb` as fencing device with `/dev/sdb2` being a dedicated partition on the shared storage for heartbeating and fencing:

```
primitive sbd_stonith stonith:external/sbd \
      meta target-role="Started" \
      op monitor interval="15" timeout="15" start-delay="15" \
      params sbd_device="/dev/sdb2"
```

4 Review your changes with `show`.

5 If everything is correct, submit your changes with `commit` and leave the crm live configuration with `exit`.

12.4 Creating OCFS2 Volumes

After you have configured DLM and O2CB as cluster resources as described in Section 12.3, “Configuring OCFS2 Services and a STONITH Resource” (page 157), configure your system to use OCFS2 and create OCFs2 volumes.

NOTE: OCFS2 Volumes for Application and Data Files

We recommend that you generally store application files and data files on different OCFS2 volumes. If your application volumes and data volumes have different requirements for mounting, it is mandatory to store them on different volumes.

Before you begin, prepare the block devices you plan to use for your OCFS2 volumes. Leave the devices as free space.

Then create and format the OCFS2 volume with the `mkfs.ocfs2` as described in Procedure 12.3, "Creating and Formatting an OCFS2 Volume" (page 161). The most important parameters for the command are listed in Table 12.2, "Important OCFS2 Parameters" (page 160). For more information and the command syntax, refer to the `mkfs.ocfs2` man page.

Table 12.2 *Important OCFS2 Parameters*

OCFS2 Parameter	Description and Recommendation
Volume Label (`-L`)	A descriptive name for the volume to make it uniquely identifiable when it is mounted on different nodes. Use the `tunefs.ocfs2` utility to modify the label as needed.
Cluster Size (`-C`)	Cluster size is the smallest unit of space allocated to a file to hold the data. For the available options and recommendations, refer to the `mkfs.ocfs2` man page.
Number of Node Slots (`-N`)	The maximum number of nodes that can concurrently mount a volume. For each of the nodes, OCFS2 creates separate system files, such as the journals, for each of the nodes. Nodes that access the volume can be a combination of little-endian architectures (such as x86, x86-64, and ia64) and big-endian architectures (such as ppc64 and s390x). Node-specific files are referred to as local files. A node slot number is appended to the local file. For example: `journal:0000` belongs to whatever node is assigned to slot number `0`. Set each volume's maximum number of node slots when you create it, according to how many nodes that you expect to concurrently mount the volume. Use the `tunefs.ocfs2` utility to increase the number of node slots as needed. Note that the value cannot be decreased.

OCFS2 Parameter	Description and Recommendation
Block Size (-b)	The smallest unit of space addressable by the file system. Specify the block size when you create the volume. For the available options and recommendations, refer to the mkfs.ocfs2 man page.
Specific Features On/Off (--fs-features)	A comma separated list of feature flags can be provided, and mkfs.ocfs2 will try to create the file system with those features set according to the list. To turn a feature on, include it in the list. To turn a feature off, prepend no to the name. For on overview of all available flags, refer to the mkfs.ocfs2 man page.
Pre-Defined Features (--fs-feature-level)	Allows you to choose from a set of pre-determined file system features. For the available options, refer to the mkfs.ocfs2 man page.

If you do not specify any specific features when creating and formatting the volume with mkfs.ocfs2, the following features are enabled by default: backup-super, sparse, inline-data, unwritten, metaecc, indexed-dirs, and xattr.

Procedure 12.3 *Creating and Formatting an OCFS2 Volume*

Execute the following steps only on *one* of the cluster nodes.

1 Open a terminal window and log in as root.

2 Check if the cluster is online with the command crm_mon.

3 Create and format the volume using the mkfs.ocfs2 utility. For information about the syntax for this command, refer to the mkfs.ocfs2 man page.

For example, to create a new OCFS2 file system on /dev/sdb1 that supports up to 32 cluster nodes, use the following command:

```
mkfs.ocfs2 -N 32 /dev/sdb1
```

12.5 Mounting OCFS2 Volumes

You can either mount an OCFS2 volume manually or with the cluster manager, as described in Procedure 12.5, “Mounting an OCFS2 Volume with the Cluster Manager” (page 162).

Procedure 12.4 *Manually Mounting an OCFS2 Volume*

1 Open a terminal window and log in as `root`.

2 Check if the cluster is online with the command `crm_mon`.

3 Mount the volume from the command line, using the `mount` command.

WARNING: Manually Mounted OCFS2 Devices

If you mount the OCFS2 file system manually for testing purposes, make sure to unmount it again before starting to use it by means of OpenAIS.

Procedure 12.5 *Mounting an OCFS2 Volume with the Cluster Manager*

To mount an OCFS2 volume with the High Availability software, configure an ocf file system resource in the cluster. The following procedure uses the `crm` shell to configure the cluster resources. Alternatively, you can also use the Pacemaker GUI to configure the resources.

1 Start a shell and log in as `root` or equivalent.

2 Run `crm configure`.

3 Configure Pacemaker to mount the OCFS2 file system on every node in the cluster:

```
primitive ocfs2-1 ocf:heartbeat:Filesystem \
      params device="/dev/sdb1" directory="/mnt/shared" fstype="ocfs2"
options="acl" \
      op monitor interval="20" timeout="40"
```

4 Add the file system primitive to the base group you have configured in Procedure 12.1, “Configuring DLM and O2CB Resources” (page 158):

4a Enter

```
edit base-group
```

4b In the vi editor that opens, modify the group as follows and save your changes:

```
group base-group dlm o2cb ocfs2-1
```

Due to the base group's internal collocation and ordering, Pacemaker will only start the `ocfs2-1` resource on nodes that also have an o2cb resource already running.

5 Review your changes with `show`. To check if you have configured all needed resources, also refer to Appendix B, *Example Configuration for OCFS2 and cLVM* (page 371).

6 If everything is correct, submit your changes with `commit` and leave the crm live configuration with `exit`.

12.6 For More Information

For more information about OCFS2, see the following links:

`http://oss.oracle.com/projects/ocfs2/`
OCFS2 project home page at Oracle.

`http://oss.oracle.com/projects/ocfs2/documentation`
OCFS2 User's Guide, available from the project documentation home page.

Distributed Replicated Block Device (DRBD) 13

DRBD allows you to create a mirror of two block devices that are located at two different sites across an IP network. When used with OpenAIS, DRBD supports distributed high-availability Linux clusters. This chapter shows you how to install and set up DRBD.

13.1 Conceptual Overview

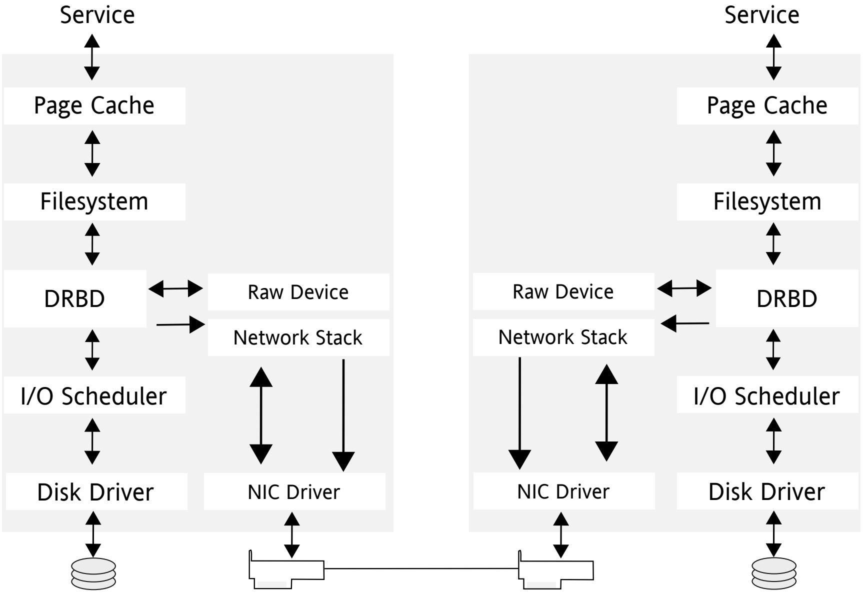
DRBD replicates data on the primary device to the secondary device in a way that ensures that both copies of the data remain identical. Think of it as a networked RAID 1. It mirrors data in real-time, so its replication occurs continuously. Applications do not need to know that in fact their data is stored on different disks.

IMPORTANT: Unencrypted Data

The data traffic between mirrors is not encrypted. For secure data exchange, you should deploy a Virtual Private Network (VPN) solution for the connection.

DRBD is a Linux kernel module and sits between the I/O scheduler at the lower end and the file system at the upper end, see Figure 13.1, “Position of DRBD within Linux” (page 166). To communicate with DRBD, users use the high-level command `drbdadm`. For maximum flexibility DRBD comes with the low-level tool `drbdsetup`.

Figure 13.1 *Position of DRBD within Linux*



DRBD allows you to use any block device supported by Linux, usually:

- partition or complete hard disk
- software RAID
- Logical Volume Manager (LVM)
- Enterprise Volume Management System (EVMS)

By default, DRBD uses the TCP ports `7788` and higher for communication between DRBD nodes. Make sure that your firewall does not prevent communication on this port.

You must set up the DRBD devices before creating file systems on them. Everything pertaining to user data should be done solely via the `/dev/drbd_`*`R`* device and not on the raw device, as DRBD uses the last 128 MB of the raw device for metadata. Make sure to create file systems only on the `/dev/drbd<n>` device and not on the raw device.

For example, if the raw device is 1024 MB in size, the DRBD device has only 896 MB available for data, with 128 MB hidden and reserved for the metadata. Any attempt to access the space between 896 MB and 1024 MB fails because it is not available for user data.

13.2 Installing DRBD Services

To install the needed packages for DRBD, install the High Availability Extension Add-On product on both SUSE Linux Enterprise Server machines in your networked cluster as described in Part I, “Installation and Setup” (page 1). Installing High Availability Extension also installs the DRBD program files.

If you do not need the complete cluster stack but just want to use DRBD, Table 13.1, “DRBD RPM Packages” (page 167) contains a list of all RPM packages for DRBD. During the last version the `drbd` package was split into separate packages.

Table 13.1 *DRBD RPM Packages*

Filename	Explanation
`drbd`	Convenience package, split into other
`drbd-bash-completion`	Programmable bash completion support for drbdadm
`drbd-heartbeat`	Heartbeat resource agent for DRBD (only needed for Heartbeat)
`drbd-kmp-default`	Kernel module for DRBD (needed)
`drbd-kmp-xen`	Xen kernel module for DRBD
`drbd-udev`	udev integration scripts for DRBD, managing symlinks to DRBD devices in `/dev/drbd/by-res` and `/dev/drbd/by-disk`
`drbd-utils`	Management utilities for DRBD (needed)

Filename	Explanation
drbd-pacemaker	Pacemaker resource agent for DRBD
drbd-xen	Xen block device management script for DRBD
yast2-drbd	YaST DRBD Configuration (recommended)

To simplify the work with drbdadm, use the Bash completion support in the RPM package drbd-bash-completion. If you want to enable it in your current shell session, insert the following command:

```
source /etc/bash_completion.d/drbdadm.sh
```

To use it permanently for root, create a file /root/.bashrc and insert the previous line.

13.3 Configuring the DRBD Service

NOTE

The following procedure uses the server names jupiter and venus, and the cluster resource name r0. It sets up jupiter as the primary node. Make sure to modify the instructions to use your own nodes and filenames.

Before you start configuring DRBD, make sure the block devices in your Linux nodes are ready and partitioned (if needed). The following procedure assumes you have two nodes, jupiter and venus, and they use the TCP port 7788. Make sure this port is open in your firewall.

To set up DRBD manually, proceed as follows:

Procedure 13.1 *Manually Configuring DRBD*

1. Log in as user root.

2. Change DRBD's configuration files:

2a Open the file `/etc/drbd.conf` and insert the following lines, if not available:

```
include "drbd.d/global_common.conf";
include "drbd.d/*.res";
```

Beginning with DRBD 8.3 the configuration file is split into separate files, located under the directory `/etc/drbd.d`.

2b Open the file `/etc/drbd.d/global_common.conf`. It contains already some pre-defined values. Go to the `startup` section and insert these lines:

```
startup {
    # wfc-timeout degr-wfc-timeout outdated-wfc-timeout
    # wait-after-sb;
    wfc-timeout 1;
    degr-wfc-timeout 1;
}
```

These options are used to reduce the timeouts when booting, see http://www.drbd.org/users-guide-emb/re-drbdconf.html for more details.

2c Create the file `/etc/drbd.d/r0.res`, change the lines according to your situation, and save it:

```
resource r0 { ❶
  device /dev/drbd_r0 minor 0; ❷
  disk /dev/sda1; ❸
  meta-disk internal; ❹
  on jupiter { ❺
    address  192.168.1.10:7788; ❻
  }
  on venus { ❺
    address 192.168.1.11:7788; ❻
  }
  syncer {
    rate  7M; ❼
  }
}
```

❶ Name of the resource. It is recommended to use resource names like `r0`, `r1`, etc.

❷ The device name for DRBD and its minor number.

In the example above, the device node name as created with udev is referenced (`/dev/drbd_r0`, with `r0` representing the resource name). For this usage, you need to have the `drbd-udev` package installed. Alternatively, omit the device node name in the configuration and use the following line instead:

```
device minor 0
```

❸ The device that is replicated between nodes. Note, in this example the devices are the same on both nodes. If you need different devices, move the `disk` parameter into the `on` section.

❹ The meta-disk parameter usually contains the value `internal`, but it is possible to specify an explicit device to hold the meta data. See http://www.drbd.org/users-guide-emb/ch-internals.html#s-metadata for more information.

❺ The `on` section contains the hostname of the node

❻ The IP address and port number of the respective node. Each resource needs an individual port, usually starting with `7788`.

❼ The synchronization rate. Set it to one third of your bandwith. It only limits the resynchronization, not the mirroring.

3 Check the syntax of your configuration file(s). If the following command returns an error, verify your files:

```
drbdadm dump all
```

4 Copy the DRBD configuration files to the other node:

```
scp /etc/drbd.conf venus:/etc/
scp /etc/drbd.d/*  venus:/etc/drbd.d/
```

5 Initialize the meta data on *both* systems by entering the following on each node.

```
drbdadm -- --ignore-sanity-checks create-md r0
rcdrbd start
```

If your disk contains already a filesystem that you do not need anymore, destroy the filesystem structure with the following command and repeat this step:

```
dd if=/dev/zero of=/dev/sdb1 count=10000
```

6 Watch the DRBD status by entering the following on each node:

```
rcdrbd status
```

You should get something like this:

```
drbd driver loaded OK; device status:
version: 8.3.7 (api:88/proto:86-91)
GIT-hash: ea9e28dbff98e331a62bcbcc63a6135808fe2917 build by phil@fat-tyre, 2010-01-13 17:17:27
m:res  cs         ro                   ds                         p  mounted  fstype
0:r0   Connected  Secondary/Secondary  Inconsistent/Inconsistent  C
```

7 Start the resync process on your intended primary node (jupiter in this case):

```
drbdadm -- --overwrite-data-of-peer primary r0
```

8 Check the status again with `rcdrbd status` and you get:

```
...
m:res  cs         ro                 ds                 p  mounted  fstype
0:r0   Connected  Primary/Secondary  UpToDate/UpToDate  C
```

The status in the `ds` row (disk status) must be UpToDate on both nodes.

9 Set jupiter as primary node:

```
drbdadm primary r0
```

10 Create your filesystem on top of your DRBD device, for example:

```
mkfs.ext3 /dev/drbd_r0
```

11 Mount the file system and use it:

```
mount /dev/drbd_r0 /mnt/
```

13.4 Testing the DRBD Service

If the install and configuration procedures worked as expected, you are ready to run a basic test of the DRBD functionality. This test also helps with understanding how the software works.

1 Test the DRBD service on jupiter.

1a Open a terminal console, then log in as `root`.

1b Create a mount point on jupiter, such as `/srv/r0mount`:

```
mkdir -p /srv/r0mount
```

1c Mount the `drbd` device:

```
mount -o rw /dev/drbd0 /srv/r0mount
```

1d Create a file from the primary node:

```
touch /srv/r0mount/from_node1
```

2 Test the DRBD service on venus.

2a Open a terminal console, then log in as `root`.

2b Unmount the disk on jupiter:

```
umount /srv/r0mount
```

2c Downgrade the DRBD service on jupiter by typing the following command on jupiter:

```
drbdadm secondary r0
```

2d On venus, promote the DRBD service to primary:

```
drbdadm primary r0
```

2e On venus, check to see if venus is primary:

```
rcdrbd status
```

2f On venus, create a mount point such as `/srv/r0mount`:

```
mkdir /srv/r0mount
```

2g On venus, mount the DRBD device:

```
mount -o rw /dev/drbd_r0 /srv/r0mount
```

2h Verify that the file you created on jupiter is viewable.

```
ls /srv/r0mount
```

The `/srv/r0mount/from_node1` file should be listed.

3 If the service is working on both nodes, the DRBD setup is complete.

4 Set up jupiter as the primary again.

4a Dismount the disk on venus by typing the following command on venus:

```
umount /srv/r0mount
```

4b Downgrade the DRBD service on venus by typing the following command on venus:

```
drbdadm secondary r0
```

4c On jupiter, promote the DRBD service to primary:

```
drbdadm primary r0
```

4d On jupiter, check to see if jupiter is primary:

```
rcdrbd status
```

5 To get the service to automatically start and fail over if the server has a problem, you can set up DRBD as a high availability service with OpenAIS. For information about installing and configuring OpenAIS for SUSE Linux Enterprise 11 see Part II, “Configuration and Administration” (page 33).

13.5 Tuning DRBD

There are several ways to tune DRBD:

1. Use an external disk for your metadata. This speeds up your connection.

2. Create a udev rule to change the read ahead of the DRBD device. Save the following line in the file /etc/udev/rules.d/82-dm-ra.rules and change the read_ahead_kb value to your workload:

   ```
   ACTION=="add", KERNEL=="dm-*", ATTR{bdi/read_ahead_kb}="4100"
   ```

   This line works only, when you use LVM.

3. Activate bmbv on Linux software RAID systems. Use the following line in the common disk section of your DRBD configuration, usually in /etc/drbd.d/global_common.conf:

   ```
   disk {
     use-bmbv;
   }
   ```

13.6 Troubleshooting DRBD

The drbd setup involves many different components and problems may arise from different sources. The following sections cover several common scenarios and recommends various solutions.

13.6.1 Configuration

If the initial drbd setup does not work as expected, there is probably something wrong with your configuration.

To get information about the configuration:

1 Open a terminal console, then log in as root.

2 Test the configuration file by running drbdadm with the -d option. Enter the following command:

```
drbdadm -d adjust r0
```

In a dry run of the adjust option, drbdadm compares the actual configuration of the DRBD resource with your DRBD configuration file, but it does not execute the calls. Review the output to make sure you know the source and cause of any errors.

3. If there are errors in the /etc/drbd.d/* and drbd.conf files, correct them before continuing.

4. If the partitions and settings are correct, run drbdadm again without the -d option.

   ```
   drbdadm adjust r0
   ```

   This applies the configuration file to the DRBD resource.

13.6.2 Hostnames

For DRBD, hostnames are case sensitive (Node0 would be a different host than node0).

If you have several network devices and want to use a dedicated network device, the hostname will likely not resolve to the used IP address. In this case, use the parameter disable-ip-verification.

13.6.3 TCP Port 7788

If your system is unable to connect to the peer, this might be a problem with your local firewall. By default, DRBD uses the TCP port 7788 to access the other node. Make sure that this port is accessible on both nodes.

13.6.4 DRBD Devices Broken after Reboot

In cases when DRBD does not know which of the real devices holds the latest data, it changes to a split brain condition. In this case, the respective DRBD subsystems come up as secondary and do not connect to each other. In this case, the following message is written to /var/log/messages:

```
Split-Brain detected, dropping connection!
```

To resolve this situation, enter the following on the node which has data to be discarded:

```
drbdadm secondary r0
drbdadm -- --discard-my-data connect r0
```

On the node which has the latest data enter the following:

```
drbdadm connect r0
```

13.7 For More Information

The following open source resources are available for DRBD:

- The project home page http://www.drbd.org.
- http://clusterlabs.org/wiki/DRBD_HowTo_1.0 by the Linux Pacemaker Cluster Stack Project.
- The following man pages for DRBD are available in the distribution: drbd(8), drbddisk(8), drbdsetup(8), drbdsetup(8), drbdadm(8), drbd.conf(5).
- Find a commented example configuration for DRBD at /usr/share/doc/packages/drbd/drbd.conf

Cluster LVM 14

When managing shared storage on a cluster, every node must be informed about changes that are done to the storage subsystem. The Linux Volume Manager 2 (LVM2), which is widely used to manage local storage, has been extended to support transparent management of volume groups across the whole cluster. Clustered volume groups can be managed using the same commands as local storage.

14.1 Conceptual Overview

Clustered LVM is coordinated with different tools:

Distributed Lock Manager (DLM)
: Coordinates disk access for cLVM.

Logical Volume Manager2 (LVM2)
: Enables flexible distribution of one file system over several disks. LVM provides a virtual pool of disk space.

Clustered Logical Volume Manager (cLVM)
: Coordinates access to the LVM2 metadata so every node knows about changes. cLVM does not coordinate access to the shared data itself; to enable cLVM to do so, you must configure OCFS2 or other cluster-aware applications on top of the cLVM-managed storage.

14.2 Configuration of cLVM

Depending on your scenario it is possible to create a RAID 1 device with cLVM with the following layers:

- **LVM** This is a very flexible solution if you want to increase or decrease your file system size, add more physical storage, or create snapshots of your file systems. This method is described in Section 14.2.2, “Scenario: cLVM With iSCSI on SANs” (page 181).

- **DRBD** This solution only provides RAID 0 (striping) and RAID 1 (mirroring). The last method is described in Section 14.2.3, “Scenario: cLVM With DRBD” (page 185).

- **MD Devices (Linux Software RAID or `mdadm`)** Although this solution provides all RAID levels, it does not support clusters yet.

Make sure you have fulfilled the following prerequisites:

- A shared storage device is available, such as provided by a Fibre Channel, FCoE, SCSI, iSCSI SAN, or DRBD.

- In case of DRBD, both nodes must be primary (as described in the following procedure).

- Check if the locking type of LVM2 is cluster-aware. The keyword `locking_type` in `/etc/lvm/lvm.conf` must contain the value 3 (should be the default). Copy the configuration to all nodes, if necessary.

NOTE: Create Cluster Resources First

First create your cluster resources as described in Section 14.2.1, “Creating the Cluster Resources” (page 178), and then your LVM volumes. Otherwise it is impossible to remove the volumes later.

14.2.1 Creating the Cluster Resources

Preparing the cluster for use of cLVM includes the following basic steps:

- Creating a DLM Resource (page 179)

- Creating LVM and cLVM Resources (page 179)

Procedure 14.1 *Creating a DLM Resource*

NOTE: DLM Resource for Both cLVM and OCFS2

Both cLVM and OCFS2 need a DLM resource that runs on all nodes in the cluster and therefore is usually configured as a clone. If you have a setup that includes both OCFS2 and cLVM, configuring *one* DLM resource for both OCFS2 and cLVM is enough.

1 Start a shell and log in as `root`.

2 Run `crm configure`.

3 Check the current configuration of the cluster resources with `show`.

4 If you have already configured a DLM resource (and a corresponding base group and base clone), continue with Procedure 14.2, “Creating LVM and cLVM Resources” (page 179).

Otherwise, configure a DLM resource and a corresponding base group and base clone as described in Procedure 12.1, “Configuring DLM and O2CB Resources” (page 158).

5 Leave the crm live configuration with `exit`.

Procedure 14.2 *Creating LVM and cLVM Resources*

1 Start a shell and log in as `root`.

2 Run `crm configure`.

3 Configure a cLVM resource as follows:

```
primitive clvm ocf:lvm2:clvmd \
      params daemon_timeout="30" \
      op monitor interval="60" timeout="60"
```

4 Configure an LVM resource for the volume group as follows:

```
primitive vg1 ocf:heartbeat:LVM \
      params volgrpname="cluster-vg" \
      op monitor interval="60" timeout="60"
```

5 However, if you want the volume group to be activated exclusively on *one* node, configure the LVM resource as described below and omit Step 6 (page 180):

```
primitive vg1 ocf:heartbeat:LVM \
      params volgrpname="cluster-vg" exclusive="yes" \
      op monitor interval="60" timeout="60"
```

In this case, cLVM will protect all logical volumes within the VG from being activated on multiple nodes, as an additional measure of protection for non-clustered applications.

6 To ensure that the cLVM and LVM resources are activated cluster-wide, add both primitives to the base group you have created in Procedure 12.1, “Configuring DLM and O2CB Resources” (page 158):

6a Enter

```
edit base-group
```

6b In the vi editor that opens, modify the group as follows and save your changes:

```
group base-group dlm o2cb clvm vg1 ocfs2-1
```

IMPORTANT: Setup Without OCFS2

If your setup does not include OCFS2, omit the `ocfs2-1` primitive from the base group. The oc2cb primitive can be configured and included in the group anyway, regardless of whether you use OCFS2 or not.

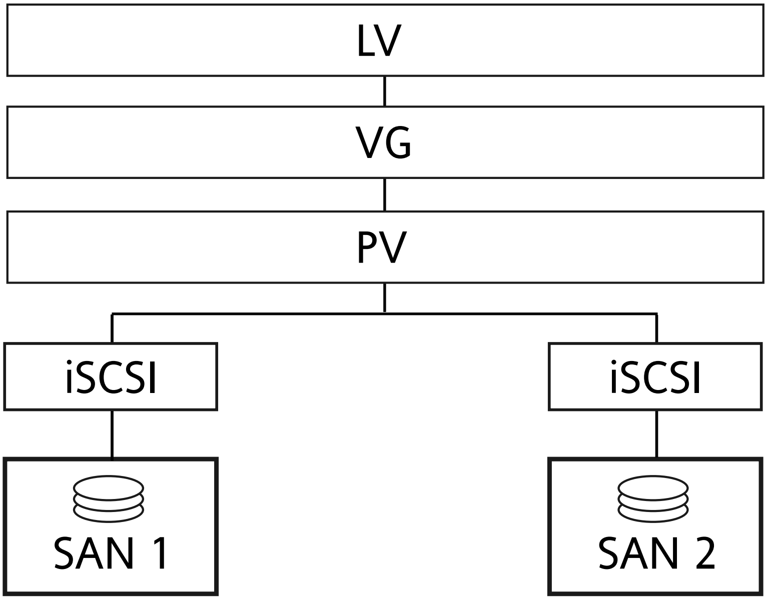
7 Review your changes with `show`. To check if you have configured all needed resources, also refer to Appendix B, *Example Configuration for OCFS2 and cLVM* (page 371).

8 If everything is correct, submit your changes with `commit` and leave the crm live configuration with `exit`.

14.2.2 Scenario: cLVM With iSCSI on SANs

The following scenario uses two SAN boxes which export their iSCSI targets to several clients. The general idea is displayed in Figure 14.1, “Setup of iSCSI with cLVM” (page 181).

Figure 14.1 *Setup of iSCSI with cLVM*



WARNING: Data Loss

The following procedures will destroy any data on your disks!

Configure only one SAN box first. Each SAN box has to export its own iSCSI target. Proceed as follows:

Procedure 14.3 *Configuring iSCSI Targets (SAN)*

1 Run YaST and click *Network Services* > *iSCSI Target* to start the iSCSI Server module.

2 If you want to start the iSCSI target whenever your computer is booted, choose *When Booting*, otherwise choose *Manually*.

3 If you have a firewall running, enable *Open Port in Firewall*.

4 Switch to the *Global* tab. If you need authentication enable incoming or outgoing authentication or both. In this example, we select *No Authentication*.

5 Add a new iSCSI target:

  5a Switch to the *Targets* tab.

  5b Click *Add*.

  5c Enter a target name. The name has to be formatted like this:

  ```
  iqn.DATE.DOMAIN
  ```

  For more information about the format, refer to , *Section 3.2.6.3.1. Type "iqn." (iSCSI Qualified Name)* [`http://www.ietf.org/rfc/rfc3720.txt`].

  5d If you want a more descriptive name, you can change it as long as your identifier is unique between your different targets.

  5e Click *Add*.

  5f Enter the device name in *Path* and use a *Scsiid*.

  5g Click *Next* two times.

6 Confirm the warning box with *Yes*.

7 Open the configuration file `/etc/iscsi/iscsi.conf` and change the parameter `node.startup` to `automatic`.

Now set up your iSCSI initiators as follows:

Procedure 14.4 *Configuring iSCSI Initiators*

1. Run YaST and click *Network Services* > *iSCSI Initiator*.

2. If you want to start the iSCSI initiator whenever your computer is booted, choose *When Booting*, otherwise set *Manually*.

3. Change to the *Discovery* tab and click the *Discovery* button.

4. Add your IP address and your port of your iSCSI target (see Procedure 14.3, “Configuring iSCSI Targets (SAN)” (page 181)). Normally, you can leave the port as it is and use the default value.

5. If you use authentication, insert the incoming and outgoing username and password, otherwise activate *No Authentication*.

6. Select *Next*. The found connections are displayed in the list.

7. Proceed with *Finish*.

8. Open a shell, log in as `root`.

9. Test if the iSCSI initiator has been started successfully:

   ```
   iscsiadm -m discovery -t st -p 192.168.3.100
   192.168.3.100:3260,1 iqn.2010-03.de.jupiter:san1
   ```

10. Establish a session:

    ```
    iscsiadm -m node -l
    Logging in to [iface: default, target: iqn.2010-03.de.jupiter:san2,
    portal: 192.168.3.100,3260]
    Logging in to [iface: default, target: iqn.2010-03.de.venus:san1, portal:
     192.168.3.101,3260]
    Login to [iface: default, target: iqn.2010-03.de.jupiter:san2, portal:
    192.168.3.100,3260]: successful
    Login to [iface: default, target: iqn.2010-03.de.venus:san1, portal:
    192.168.3.101,3260]: successful
    ```

    See the device names with `lsscsi`:

    ```
    ...
    [4:0:0:2]    disk    IET      ...     0     /dev/sdd
    [5:0:0:1]    disk    IET      ...     0     /dev/sde
    ```

Look for entries with `IET` in their third column. In this case, the devices are `/dev/sdd` and `/dev/sde`.

Procedure 14.5 *Creating the LVM Volume Groups*

1 Open a `root` shell on one of the nodes you have run the iSCSI initiator from Procedure 14.4, "Configuring iSCSI Initiators" (page 183).

2 Prepare the physical volume for LVM with the command `pvcreate` on the disks `/dev/sdd` and `/dev/sde`:

```
pvcreate /dev/sdd
pvcreate /dev/sde
```

3 Create the cluster-aware volume group on both disks:

```
vgcreate --clustered y clustervg /dev/sdd /dev/sde
```

4 Create logical volumes as needed:

```
lvcreate --name clusterlv --size 500M clustervg
```

5 Check the physical volume with `pvdisplay`:

```
  --- Physical volume ---
      PV Name               /dev/sdd
      VG Name               clustervg
      PV Size               509,88 MB / not usable 1,88 MB
      Allocatable           yes
      PE Size (KByte)       4096
      Total PE              127
      Free PE               127
      Allocated PE          0
      PV UUID               52okH4-nv3z-2AUL-GhAN-8DAZ-GMtU-Xrn9Kh

      --- Physical volume ---
      PV Name               /dev/sde
      VG Name               clustervg
      PV Size               509,84 MB / not usable 1,84 MB
      Allocatable           yes
      PE Size (KByte)       4096
      Total PE              127
      Free PE               127
      Allocated PE          0
      PV UUID               Ouj3Xm-AI58-lxB1-mWm2-xn51-agM2-0UuHFC
```

6 Check the volume group with `vgdisplay`:

```
--- Volume group ---
    VG Name               clustervg
    System ID
    Format                lvm2
    Metadata Areas        2
    Metadata Sequence No  1
    VG Access             read/write
    VG Status             resizable
    Clustered             yes
    Shared                no
    MAX LV                0
    Cur LV                0
    Open LV               0
    Max PV                0
    Cur PV                2
    Act PV                2
    VG Size               1016,00 MB
    PE Size               4,00 MB
    Total PE              254
    Alloc PE / Size       0 / 0
    Free  PE / Size       254 / 1016,00 MB
    VG UUID               UCyWw8-2jqV-enuT-KH4d-NXQI-JhH3-J24anD
```

After you have created the volumes and started your resources you should have a new device named `/dev/dm-0` . It is recommended to use a clustered file system on top of your LVM resource, for example OCFS. For more information, see Chapter 12, *Oracle Cluster File System 2* (page 155)

14.2.3 Scenario: cLVM With DRBD

The following scenarios can be used if you have data centers located in different parts of your city, country, or continent.

Procedure 14.6 *Creating a Cluster-Aware Volume Group With DRBD*

1 Create a primary/primary DRBD resource:

1a First, set up a DRBD device as primary/secondary as described in Procedure 13.1, "Manually Configuring DRBD" (page 168). Make sure the disk state is `up-to-date` on both nodes. Check this with `cat /proc/drbd` or with `rcdrbd status`.

1b Add the following options to your configuration file (usually something like /etc/drbd.d/r0.res):

```
resource r0 {
  startup {
    become-primary-on both;
  }

  net {
     allow-two-primaries;
  }
  ...
}
```

1c Copy the changed configuration file to the other node, for example:

```
scp /etc/drbd.d/r0.res venus:/etc/drbd.d/
```

1d Run the following commands on *both* nodes:

```
drbdadm disconnect r0
drbdadm connect r0
drbdadm primary r0
```

1e Check the status of your nodes:

```
cat /proc/drbd
...
 0: cs:Connected ro:Primary/Primary ds:UpToDate/UpToDate C r----
```

2 Include the clvmd resource as a clone in the pacemaker configuration, and make it depend on the DLM clone resource. See Procedure 14.1, “Creating a DLM Resource” (page 179) for detailed instructions. Before proceeding, confirm that these resources have started successfully on your cluster. You may use crm_mon or the GUI to check the running services.

3 Prepare the physical volume for LVM with the command pvcreate. For example, on the device /dev/drbd_r0 the command would look like this:

```
pvcreate /dev/drbd_r0
```

4 Create a cluster-aware volume group:

```
vgcreate --clustered y myclusterfs /dev/drbd_r0
```

5 Create logical volumes as needed. You may probably want to change the size of the logical volume. For example, create a 4 GB logcial volume with the following command:

```
lvcreate --name testlv -L 4G myclusterfs
```

6 The logical volumes within the VG are now available as file system mounts or raw usage. Ensure that services using them have proper dependencies to collocate them with and order them after the VG has been activated.

After finishing these configuration steps, the LVM2 configuration can be done just like on any standalone workstation.

14.3 Configuring Eligible LVM2 Devices Explicitly

When several devices seemingly share the same physical volume signature (as can be the case for multipath devices or DRBD), it is recommended to explicitly configure the devices which LVM2 scans for PVs.

For example, if the command `vgcreate` uses the physical device instead of using the mirrored block device, DRBD will be confused which may result in a split brain condition for DRBD.

To deactivate a single device for LVM2, do the following:

1 Edit the file `/etc/lvm/lvm.conf` and search for the line starting with `filter`.

2 The patterns there are handled as regular expressions. A leading “a” means to accept a device pattern to the scan, a leading “r” rejects the devices that follow the device pattern.

3 To remove a device named `/dev/sdb1`, add the following expression to the filter rule:

```
"r|^/dev/sdb1$|"
```

The complete filter line will look like the following:

```
filter = [ "r|^/dev/sdb1$|", "r|/dev/.*/by-path/.*|", "r|/dev/.*/by-id/.*|",
 "a/.*/" ]
```

A filter line, that accepts DRBD and MPIO devices but rejects all other devices would look like this:

```
filter = [ "a|/dev/drbd.*|", "a|/dev/.*/by-id/dm-uuid-mpath-.*|", "r/.*/"
]
```

4 Write the configuration file and copy it to all cluster nodes.

14.4 For More Information

Thorough information is available from the pacemaker mailing list, available at `http://www.clusterlabs.org/wiki/Help:Contents`.

The official cLVM FAQ can be found at `http://sources.redhat.com/cluster/wiki/FAQ/CLVM`.

Storage Protection 15

The High Availability cluster stack's highest priority is protecting the integrity of data. This is achieved by preventing uncoordinated concurrent access to data storage: For example, ext3 file systems are only mounted once in the cluster, OCFS2 volumes will not be mounted unless coordination with other cluster nodes is available. In a well-functioning cluster Pacemaker will detect if resources are active beyond their concurrency limits and initiate recovery. Furthermore, its policy engine will never exceed these limitations.

However, network partitioning or software malfunction could potentially cause scenarios where several coordinators are elected. If this so-called split brain scenarios were allowed to unfold, data corruption might occur. Hence, several layers of protection have been added to the cluster stack to mitigate this.

The primary component contributing to this goal is IO fencing/STONITH since it ensures that all other access prior to storage activation is terminated. Other mechanisms are cLVM2 exclusive activation or OCFS2 file locking support to protect your system against administrative or application faults. Combined appropriately for your setup, these can reliably prevent split-brain scenarios from causing harm.

This chapter describes an IO fencing mechanism that leverages the storage itself, followed by the description of an additional layer of protection to ensure exclusive storage access. These two mechanisms can be combined for higher levels of protection.

15.1 Storage-based Fencing

You can reliably avoid split-brain scenarios by using Split Brain Detector (SBD), `watchdog` support and the `external/sbd` STONITH agent.

15.1.1 Overview

In an environment where all nodes have access to shared storage, a small partition (1MB) is formated for the use with SBD. After the respective daemon is configured, it is brought online on each node before the rest of the cluster stack is started. It is terminated after all other cluster components have been shut down, thus ensuring that cluster resources are never activated without SBD supervision.

The daemon automatically allocates one of the message slots on the partition to itself, and constantly monitors it for messages addressed to itself. Upon receipt of a message, the daemon immediately complies with the request, such as initiating a power-off or reboot cycle for fencing.

The daemon constantly monitors connectivity to the storage device, and terminates itself in case the partition becomes unreachable. This guarantees that it is not disconnected from fencing messages. If the cluster data resides on the same logical unit in a different partition, this is not an additional point of failure: The work-load will terminate anyway if the storage connectivity has been lost.

Increased protection is offered through `watchdog` support. Modern systems support a `hardware watchdog` that has to be “tickled” or “fed” by a software component. The software component (usually a daemon) regularly writes a service pulse to the watchdog—if the daemon stops feeding the watchdog, the hardware will enforce a system restart. This protects against failures of the SBD process itself, such as dying, or becoming stuck on an IO error.

15.1.2 Setting Up Storage-based Protection

The following steps are necessary to set up storage-based protection:

1 Creating the SBD Partition (page 191)

2 Setting Up the Software Watchdog (page 192)

3 Starting the SBD Daemon (page 193)

4 Testing SBD (page 194)

5 Configuring the Fencing Resource (page 195)

All of the following procedures must be executed as `root`. Before you start, make sure the following requirements are met:

IMPORTANT: Requirements

- The environment must have shared storage reachable by all nodes.
- The shared storage segment must not make use of host-based RAID, cLVM2, nor DRBD.
- However, using storage-based RAID and multipathing is recommended for increased reliability.

Creating the SBD Partition

It is recommended to create a 1MB partition at the start of the device. If your SBD device resides on a multipath group, you need to adjust the timeouts SBD uses, as MPIO's path down detection can cause some latency. After the `msgwait` timeout, the message is assumed to have been delivered to the node. For multipath, this should be the time required for MPIO to detect a path failure and switch to the next path. You may have to test this in your environment. The node will terminate itself if the SBD daemon running on it has not updated the watchdog timer fast enough. The `watchdog` timeout must be shorter than the `msgwait` timeout—half the value is a good estimate.

NOTE: Device Name for SBD Partition

In the following, this SBD partition is referred to by `/dev/`*`SBD`*. Replace it with your actual pathname, for example: `/dev/sdc1`.

IMPORTANT: Overwriting Existing Data

Make sure the device you want to use for SBD does not hold any data. The `sdb` command will overwrite the device without further requests for confirmation.

1 Initialize the SBD device with the following command:

```
sbd -d /dev/SBD create
```

This will write a header to the device, and create slots for up to 255 nodes sharing this device with default timings.

2 If your SBD device resides on a multipath group, adjust the timeouts SBD uses. This can be specified when the SBD device is initialized (all timeouts are given in seconds):

```
/usr/sbin/sbd -d /dev/SDB -4 180❶ -1 90❷ create
```

❶ The `-4` option is used to specify the `msgwait` timeout. In the example above, it is set to `180` seconds.

❷ The `-1` option is used to specify the `watchdog` timeout. In the example above, it is set to `90` seconds.

3 With the following command, check what has been written to the device:

```
sbd -d /dev/SBD dump
Header version     : 2
Number of slots    : 255
Sector size        : 512
Timeout (watchdog) : 5
Timeout (allocate) : 2
Timeout (loop)     : 1
Timeout (msgwait)  : 10
```

As you can see, the timeouts are also stored in the header, to ensure that all participating nodes agree on them.

Setting Up the Software Watchdog

In SUSE Linux Enterprise High Availability Extension, watchdog support in the Kernel is enabled by default: It ships with a number of different Kernel modules that provide hardware-specific watchdog drivers. The High Availability Extension uses the SBD daemon as software component that “feeds” the watchdog. If configured as described in Section “Starting the SBD Daemon” (page 193), the SBD daemon will start automatically when the respective node is brought online with `rcopenais start`.

Usually, the appropriate watchdog driver for your hardware is automatically loaded during system boot. `softdog` is the most generic driver, but it is recommended to use a driver with actual hardware integration. For example:

- On HP hardware, this is the `hpwdt` driver.

- For systems with an Intel TCO, the `iTCO_wdt` driver can be used.

For a list of choices, refer to `/usr/src/`*`your_kernel_version`*`/drivers/watchdog`. Alternatively, list the drivers that have been installed with your Kernel version with the following command:

```
rpm -ql your_kernel_version | grep watchdog
```

As most watchdog driver names contain strings like `wd`, `wdt`, or `dog`, use one of the following commands to check which driver is currently loaded:

```
lsmod | grep wd
```

or

```
lsmod | grep dog
```

Starting the SBD Daemon

The SBD daemon is a critical piece of the cluster stack. It has to be running when the cluster stack is running, or even when part of it has crashed, so that it can be fenced.

1 Stop OpenAIS:

```
rcopenais stop
```

2 To make the OpenAIS init script start and stop SBD, create the file `/etc/sysconfig/sbd` and add the following lines:

```
SBD_DEVICE="/dev/SBD"
# The next line enables the watchdog support:
SBD_OPTS="-W"
```

If the SBD device is not accessible, the daemon will fail to start and inhibit OpenAIS startup.

NOTE

If the SBD device becomes inaccessible from a node, this could cause the node to enter an infinite reboot cycle. That is technically correct, but depending on your administrative policies, might be considered a nuisance. You may wish to not automatically start up OpenAIS on boot in such cases.

---

3. Copy `/etc/sysconfig/sbd` to all nodes (either manually or with Csync2, see also Section 3.2.3, "Transferring the Configuration to All Nodes" (page 26)).

4. Allocate the nodes with the following command:

   ```
   sbd -d /dev/SBD allocate nodename
   ```

5. Before proceeding, ensure that SBD has started on all nodes by executing `rcopenais restart`.

Testing SBD

1. The following command will dump the node slots and their current messages from the SBD device:

   ```
   sbd -d /dev/SBD list
   ```

   Now you should see all cluster nodes that have ever been started with SBD listed here, the message slot should show `clear`.

2. Try sending a test message to one of the nodes:

   ```
   sbd -d /dev/SBD message nodea test
   ```

3. The node will acknowledge the receipt of the message in the system logs:

   ```
   Aug 29 14:10:00 nodea sbd: [13412]: info: Received command test from nodeb
   ```

   This confirms that SBD is indeed up and running on the node, and that it is ready to receive messages.

Configuring the Fencing Resource

1 To complete the SBD setup, it is necessary to activate SBD as a STONITH/fencing mechanism in the CIB as follows:

```
crm configure
crm(live)configure# property stonith-enabled="true"
crm(live)configure# property stonith-timeout="30s"
crm(live)configure# primitive stonith_sbd stonith:external/sbd params
sbd_device="/dev/SBD"
crm(live)configure# commit
crm(live)configure# quit
```

Which value to set for `stonith-timeout` depends on the `msgwait` timeout. Provided you kept the default `msgwait` timeout value (`10` seconds), setting `stonith-timeout` to `30` seconds is appropriate.

Since node slots are allocated automatically, no manual hostlist needs to be defined.

2 Disable any other fencing devices you might have configured before, since the SBD mechanism is used for this function now.

Once the resource has started, your cluster is now successfully configured for shared-storage fencing, and will utilize this method in case a node needs to be fenced.

15.2 Ensuring Exclusive Storage Activation

This section introduces `sfex`, an additional low-level mechanism to lock access to shared storage exclusively to one node. Note that sfex does not replace STONITH. Since sfex requires shared storage, it is recommended that the `external/sbd` fencing mechanism described above is used on another partition of the storage.

By design, sfex cannot be used in conjunction with workloads that require concurrency (such as OCFS2), but serves as a layer of protection for classic fail-over style workloads. This is similar to a SCSI-2 reservation in effect, but more general.

15.2.1 Overview

In a shared storage environment, a small partition of the storage is set aside for storing one or more locks.

Before acquiring protected resources, the node must first acquire the protecting lock. The ordering is enforced by Pacemaker, and the sfex component ensures that even if Pacemaker were subject to a split-brain situation, the lock will never be granted more than once.

These locks must also be refreshed periodically, so that a node's death does not permanently block the lock and other nodes can proceed.

15.2.2 Setup

In the following, learn how to create a shared partition for use with sfex and how to configure a resource for the sfex lock in the CIB. A single sfex partition can hold any number of locks, it defaults to one, and needs 1 KB of storage space allocated per lock.

IMPORTANT: Requirements

- The shared partition for sfex should be on the same logical unit as the data you wish to protect.
- The shared sfex partition must not make use of host-based RAID, nor DRBD.
- Using a cLVM2 logical volume is possible.

Procedure 15.1 *Creating an sfex Partition*

1 Create a shared partition for use with sfex. Note the name of this partition and use it as a substitute for `/dev/sfex` below.

2 Create the sfex meta data with the following command:

```
sfex_init -n 1 /dev/sfex
```

3 Verify that the meta data has been created correctly:

```
sfex_stats -i 1 /dev/sfex ; echo $?
```

This should return `2`, since the lock is not currently held.

Procedure 15.2 *Configuring a Resource for the sfex Lock*

1 The sfex lock is represented via a resource in the CIB, configured as follows:

```
primitive sfex_1 ocf:heartbeat:sfex \
# params device="/dev/sfex" index="1" collision_timeout="1" \
      lock_timeout="70" monitor_interval="10" \
# op monitor interval="10s" timeout="30s" on_fail="fence"
```

2 To protect resources via a sfex lock, create mandatory ordering and placement constraints between the protectees and the sfex resource. If the resource to be protected has the id `filesystem1`:

```
# order order-sfex-1 inf: sfex_1 filesystem1
# colocation colo-sfex-1 inf: filesystem1 sfex_1
```

3 If using group syntax, add the sfex resource as the first resource to the group:

```
# group LAMP sfex_1 filesystem1 apache ipaddr
```

16 Samba Clustering

A clustered Samba server provides a High Availability solution in your heterogeneous networks. This chapter explains some background information and how to set up a clustered Samba server.

16.1 Conceptual Overview

Trivial Database (TDB), has been used by Samba for many years. It allows multiple applications to write simultaneously. To make sure all write operations are successfully performed and do not collide with each other, TDB uses an internal locking mechanism.

Cluster Trivial Database (CTDB) is a small extension of the existing TDB. CTDB is described by the project itself as a “cluster implementation of the TDB database used by Samba and other projects to store temporary data”.

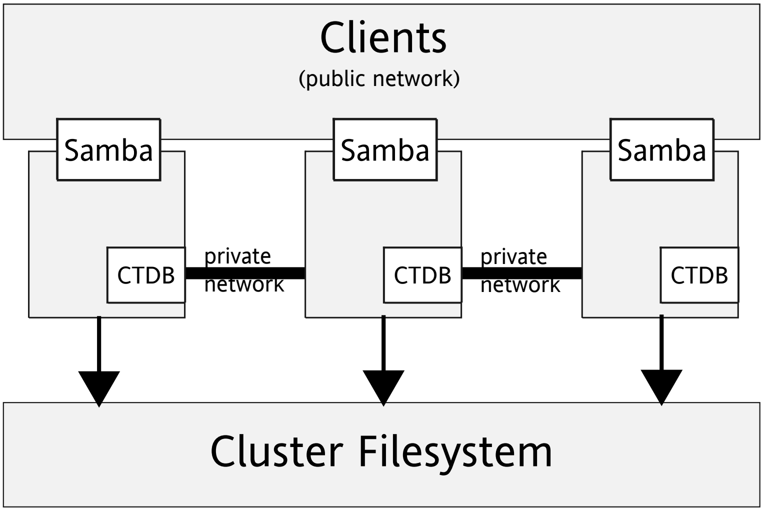
Each cluster node runs a local CTDB daemon. Samba communicates with its local CTDB daemon, instead of writing directly to its TDB. The daemons exchange metadata over the network, but actual write and read operations are done on a local copy with fast storage. The concept of CTDB is displayed in Figure 16.1, “Structure of a CTDB Cluster” (page 200).

NOTE: CTDB For Samba Only

The current implementation of the CTDB Resource Agent configures CTDB to only manage Samba. Everything else, including IP failover should be configured with Pacemaker.

Furthermore, CTDB is only supported for completely homogeneous clusters. For example, all nodes in the cluster need to have the same architecture, you cannot mix i586 with x86_64.

***Figure 16.1** Structure of a CTDB Cluster*



A clustered Samba server must share certain data:

- Mapping table that associates Unix user and group IDs to Windows users and groups.
- User database must be synchronized between all nodes.
- Join information for a member server in a Windows domain must be available on all nodes.
- Metadata has to be available on all nodes, like active SMB sessions, share connections, and various locks.

The goal is that a clustered Samba server with N+1 nodes is faster than with only N nodes. One node is not slower than an unclustered Samba server.

16.2 Basic Configuration

NOTE: Changed Configuration Files

The CTDB Resource Agent automatically changes `/etc/sysconfig/ctdb` and `/etc/samba/smb.conf`. Use `crm ra info CTDB` to list all parameters which can be specified for the CTDB resource.

To set up a clustered Samba server, proceed as follows:

1 Prepare your cluster:

1a Configure your cluster (OpenAIS, Pacemaker, OCFS2) as described in this guide in Part II, “Configuration and Administration” (page 33).

1b Configure a shared file system, like OCFS2 and mount it, for example, on `/shared`.

1c If you want to turn on POSIX ACLs, enable it:

- For a new OCFS2 file system use:

  ```
  mkfs.ocfs2 --fs-features=xattr ...
  ```

- For an existing OCFS2 file system use:

  ```
  tunefs.ocfs2 --fs-feature=xattr DEVICE
  ```

  Make sure the `acl` option is specified in the file system resource. Use the `crm` shell as follows:

  ```
  crm(live)configure# primary ocfs2-3 ocf:heartbeat:Filesystem
  options="acl" ...
  ```

1d Make sure the services `ctdb`, `smb`, `nmb`, and `winbind` are disabled:

```
chkconfig ctdb off
chkconfig smb off
chkconfig nmb off
chkconfig winbind off
```

2 Create a directory for the CTDB lock on the shared file system:

```
mkdir -p /shared/samba/
```

3 In `/etc/ctdb/nodes` insert all nodes which contain all private IP addresses of each node in the cluster:

```
192.168.1.10
192.168.1.11
```

4 Add a CTDB resource to the cluster:

```
crm configure
crm(live)configure# primitive ctdb ocf:heartbeat:CTDB params \
    ctdb_recovery_lock="/shared/samba/ctdb.lock" \
    op monitor timeout=20 interval=10
crm(live)configure# clone ctdb-clone ctdb \
    meta globally-unique="false" interleave="true"
crm(live)configure# colocation ctdb-with-fs inf: ctdb-clone fs-clone
crm(live)configure# order start-ctdb-after-fs inf: fs-clone ctdb-clone
crm(live)configure# commit
```

5 Add a clustered IP address:

```
crm(live)configure# primitive ip ocf:heartbeat:IPaddr2 params
ip=192.168.2.222 \
  clusterip_hash="sourceip-sourceport" op monitor interval=60s
crm(live)configure# clone ip-clone ip meta globally-unique="true"
crm(live)configure# colocation ip-with-ctdb inf: ip-clone ctdb-clone
crm(live)configure# order start-ip-after-ctdb inf: ctdb-clone ip-clone
crm(live)configure# commit
```

6 Check the result:

```
crm status
Clone Set: dlm-clone
     Started: [ hex-14 hex-13 ]
 Clone Set: o2cb-clone
     Started: [ hex-14 hex-13 ]
 Clone Set: c-ocfs2-3
     Started: [ hex-14 hex-13 ]
 Clone Set: ctdb-clone
     Started: [ hex-14 hex-13 ]
 Clone Set: ip-clone (unique)
     ip:0       (ocf::heartbeat:IPaddr2):       Started hex-13
     ip:1       (ocf::heartbeat:IPaddr2):       Started hex-14
```

7 Test from a client machine. On a Linux client, run the following command to see, if you can copy files from and to the system:

```
smbclient //192.168.2.222/myshare
```

16.3 Debugging and Testing Clustered Samba

To debug your clustered Samba server, the following tools which operate on different levels are available:

`ctdb_diagnostics`

Run this tool to diagnose your clustered Samba server. This gives you lots of debug messages which should help you track down any problems you might have.

The `ctdb_diagnostics` command searches for the following files which must be available on all nodes:

```
/etc/krb5.conf
/etc/hosts
/etc/ctdb/nodes
/etc/sysconfig/ctdb
/etc/resolv.conf
/etc/nsswitch.conf
/etc/sysctl.conf
/etc/samba/smb.conf
/etc/fstab
/etc/multipath.conf
/etc/pam.d/system-auth
/etc/sysconfig/nfs
/etc/exports
/etc/vsftpd/vsftpd.conf
```

If the files `/etc/ctdb/public_addresses` and `/etc/ctdb/static-routes` exist, they will be checked as well.

`ping_pong`

Check whether your file system is suitable for CTDB with `ping_pong`. It performs certain tests of your cluster file system like coherence and performance (see `http://wiki.samba.org/index.php/Ping_pong`) and as such gives some indications on how your cluster may behave under high load.

send_arp Tool and SendArp Resource Agent

> The SendArp resource agent is located in /usr/lib/heartbeat/send_arp (or /usr/lib64/heartbeat/send_arp). The send_arp tool sends out a gratuitous ARP (Address Resolution Protocol) packet and can be used for updating other machines' ARP tables. It can help to identify communication problems after a failover process. If you cannot connect to a node or ping it although it shows the clustered IP address for samba, use the send_arp command to test if the nodes only need an ARP table update.
>
> For more information, refer to http://wiki.wireshark.org/Gratuitous_ARP.

To test certain aspects of your cluster file system proceed as follows:

***Procedure 16.1** Test Coherence and Performance of Your Cluster File System*

1. Start the command ping_pong on one node and replace the placeholder *N* with the amount of nodes plus one. The filename is available in your shared storage and is therefore accessible on all nodes:

   ```
   ping_pong data.txt N
   ```

   Expect a very high locking rate as you are running only one node. If the program does not print a locking rate, replace your cluster file system.

2. Start a second copy of ping_pong on another node with the same parameters.

   Expect to see a dramatical drop concerning the locking rate. If any of the following applies to your cluster file system, replace it:

   - ping_pong does not print a locking rate per second
   - the locking rates in the two instances are not almost equal
   - the locking rate did not drop after you have started the second instance

3. Start a third copy of ping_pong. Add another node and note how the locking rates change.

4. Kill the `ping_pong` commands step-by-step. You should observe an increase of the locking rate until you get back to the single node case. If you did not get the expected behavior, replace your cluster file system.

NOTE: Joining a Domain Fails

If the CTDB resource agent repeatedly fails upon joining a domain, although the CTDB cluster setup was successful and the cluster is running fine as standalone samba server, refer to the technical information document *Joining a CTDB Cluster Into a Domain*. It is available at `http://www.novell.com/support/search.do?cmd=displayKC&docType=kc&externalId=7006496`.

16.4 For More Information

- `http://linux-ha.org/wiki/CTDB_(resource_agent)`
- `http://wiki.samba.org/index.php/CTDB_Setup`
- `http://ctdb.samba.org`
- `http://wiki.samba.org/index.php/Samba_%26_Clustering`

Part IV. Troubleshooting and Reference

17 Troubleshooting

Often, strange problems may occur that are not easy to understand (especially when starting to experiment with High Availability). However, there are several utilities that may be used to take a closer look at the High Availability internal processes. This chapter recommends various solutions.

17.1 Installation Problems

Troubleshooting difficulties installing the packages or in bringing the cluster online.

Are the HA packages installed?

> The packages needed for configuring and managing a cluster are included in the `High Availability` installation pattern, available with the High Availability Extension.
>
> Check if High Availability Extension is installed as an add-on to SUSE Linux Enterprise Server 11 SP1 on each of the cluster nodes and if the *High Availability* pattern is installed on each of the machines as described in Section 3.1, "Installing the High Availability Extension" (page 21).

Is the initial configuration the same for all cluster nodes?

> In order to communicate with each other, all nodes belonging to the same cluster need to use the same `bindnetaddr`, `mcastaddr` and `mcastport` as described in Section 3.2, "Initial Cluster Setup" (page 22).

Check if the communication channels and options configured in /etc/corosync/corosync.conf are the same for all cluster nodes.

In case you use encrypted communication, check if the /etc/corosync/authkey file is available on all cluster nodes.

All corosync.conf settings with the exception of nodeid must be the same; authkey files on all nodes must be identical.

Does the firewall allow communication via the mcastport?

If the mcastport used for communication between the cluster nodes is blocked by the firewall, the nodes cannot see each other. When configuring the initial setup with YaST as described in Section 3.1, "Installing the High Availability Extension" (page 21), the firewall settings are usually automatically adjusted.

To make sure the mcastport is not blocked by the firewall, check the settings in /etc/sysconfig/SuSEfirewall2 on each node. Alternatively, start the YaST firewall module on each cluster node. After clicking *Allowed Service* > *Advanced*, add the mcastport to the list of allowed *UDP Ports* and confirm your changes.

Is OpenAIS started on each cluster node?

Check the OpenAIS status on each cluster node with /etc/init.d/openais status. In case OpenAIS is not running, start it by executing /etc/init.d/openais start.

17.2 "Debugging" a HA Cluster

The following displays the resource operation history (option -o) and inactive resources (-r):

```
crm_mon -o -r
```

The display is refreshed when status changes (to cancel this press Ctrl + C.) An example could look like:

Example 17.1 *Stopped Resources*

```
Refresh in 10s...

============
Last updated: Mon Jan 19 08:56:14 2009
Current DC: d42 (d42)
3 Nodes configured.
3 Resources configured.
============

Online: [ d230 d42 ]
OFFLINE: [ clusternode-1 ]

Full list of resources:

Clone Set: o2cb-clone
         Stopped: [  o2cb:0 o2cb:1o2cb:2 ]
Clone Set: dlm-clone
         Stopped [ dlm:0 dlm:1 dlm:2 ]
mySecondIP      (ocf::heartbeat:IPaddr):        Stopped

Operations:
* Node d230:
   aa: migration-threshold=1000000
    + (5) probe: rc=0 (ok)
    + (37) stop: rc=0 (ok)
    + (38) start: rc=0 (ok)
    + (39) monitor: interval=15000ms rc=0 (ok)
* Node d42:
   aa: migration-threshold=1000000
    + (3) probe: rc=0 (ok)
    + (12) stop: rc=0 (ok)
```

First get your node online (see Section 17.3 (page 212)). After that, check your resources and operations.

The *Configuration Explained* PDF under http://clusterlabs.org/wiki/Documentation covers three different recovery types in the *How Does the Cluster Interpret the OCF Return Codes?* section.

17.3 FAQs

What is the state of my cluster?

To check the current state of your cluster, use one of the programs `crm_mon` or `crm status`. This displays the current DC as well as all the nodes and resources known by the current node.

Several nodes of my cluster do not see each other.

There could be several reasons:

- Look first in the configuration file `/etc/corosync/corosync.conf` and check if the multicast address is the same for every node in the cluster (look in the `interface` section with the key `mcastaddr`.)
- Check your firewall settings.
- Check if your switch supports multicast addresses
- Check if the connection between your nodes is broken. Most often, this is the result of a badly configured firewall. This also may be the reason for a *split brain* condition, where the cluster is partitioned.

I want to list my currently known resources.

Use the command `crm_resource -L` to learn about your current resources.

I configured a resource, but it always fails.

To check an OCF script use `ocf-tester`, for instance:

```
ocf-tester -n ip1 -o ip=YOUR_IP_ADDRESS \
  /usr/lib/ocf/resource.d/heartbeat/IPaddr
```

Use `-o` multiple times for more parameters. The list of required and optional parameters can be obtained by running `crm ra info` *`AGENT`*, for example:

```
crm ra info ocf:heartbeat:IPaddr
```

Before running ocf-tester, make sure the resource is not managed by the cluster.

I just get a failed message. Is it possible to get more information?

You may always add the `--verbose` parameter to your commands. If you do that multiple times, the debug output becomes very verbose. See `/var/log/messages` for useful hints.

How can I clean up my resources?

Use the following commands :

```
crm resource list
crm resource cleanup rscid [node]
```

If you leave out the node, the resource is cleaned on all nodes. More information can be found in Section 6.4.2, “Cleaning Up Resources” (page 110).

I can not mount an ocfs2 device.

Check `/var/log/messages` if there is the following line:

```
Jan 12 09:58:55 clusternode2 lrmd: [3487]: info: RA output:
(o2cb:1:start:stderr) 2009/01/12_09:58:55
  ERROR: Could not load ocfs2_stackglue
Jan 12 16:04:22 clusternode2 modprobe: FATAL: Module ocfs2_stackglue not
 found.
```

In this case the kernel module `ocfs2_stackglue.ko` is missing. Install the package `ocfs2-kmp-default`, `ocfs2-kmp-pae` or `ocfs2-kmp-xen` depending on the installed kernel.

17.4 Fore More Information

For additional information about high availability on Linux and Heartbeat including configuring cluster resources and managing and customizing a Heartbeat cluster, see http://clusterlabs.org/wiki/Documentation.

Cluster Management Tools 18

High Availability Extension ships with a comprehensive set of tools to assists you in managing your cluster from the command line. This chapter introduces the tools needed for managing the cluster configuration in the CIB and the cluster resources. Other command line tools for managing resource agents or tools used for debugging (and troubleshooting) your setup are covered in Chapter 17, *Troubleshooting* (page 209).

The following list presents several tasks related to cluster management and briefly introduces the tools to use to accomplish these tasks:

Monitoring the Cluster's Status
: The `crm_mon` command allows you to monitor your cluster's status and configuration. Its output includes the number of nodes, uname, uuid, status, the resources configured in your cluster, and the current status of each. The output of `crm_mon` can be displayed at the console or printed into an HTML file. When provided with a cluster configuration file without the status section, `crm_mon` creates an overview of nodes and resources as specified in the file. See crm_mon(8) (page 238) for a detailed introduction to this tool's usage and command syntax.

Managing the CIB
: The `cibadmin` command is the low-level administrative command for manipulating the Heartbeat CIB. It can be used to dump all or part of the CIB, update all or part of it, modify all or part of it, delete the entire CIB, or perform miscellaneous CIB administrative operations. See cibadmin(8) (page 218) for a detailed introduction to this tool's usage and command syntax.

Managing Configuration Changes

The `crm_diff` command assists you in creating and applying XML patches. This can be useful for visualizing the changes between two versions of the cluster configuration or saving changes so they can be applied at a later time using cibadmin(8) (page 218). See crm_diff(8) (page 230) for a detailed introduction to this tool's usage and command syntax.

Manipulating CIB Attributes

The `crm_attribute` command lets you query and manipulate node attributes and cluster configuration options that are used in the CIB. See crm_attribute(8) (page 227) for a detailed introduction to this tool's usage and command syntax.

Validating the Cluster Configuration

The `crm_verify` command checks the configuration database (CIB) for consistency and other problems. It can check a file containing the configuration or connect to a running cluster. It reports two classes of problems. Errors must be fixed before Heartbeat can work properly while warning resolution is up to the administrator. `crm_verify` assists in creating new or modified configurations. You can take a local copy of a CIB in the running cluster, edit it, validate it using `crm_verify`, then put the new configuration into effect using `cibadmin`. See crm_verify(8) (page 266) for a detailed introduction to this tool's usage and command syntax.

Managing Resource Configurations

The `crm_resource` command performs various resource-related actions on the cluster. It lets you modify the definition of configured resources, start and stop resources, or delete and migrate resources between nodes. See crm_resource(8) (page 242) for a detailed introduction to this tool's usage and command syntax.

Managing Resource Fail Counts

The `crm_failcount` command queries the number of failures per resource on a given node. This tool can also be used to reset the failcount, allowing the resource to again run on nodes where it had failed too often. See crm_failcount(8) (page 233) for a detailed introduction to this tool's usage and command syntax.

Managing a Node's Standby Status

The `crm_standby` command can manipulate a node's standby attribute. Any node in standby mode is no longer eligible to host resources and any resources that are there must be moved. Standby mode can be useful for performing maintenance tasks, such as kernel updates. Remove the standby attribute from the node for it to become a fully active member of the cluster again. See crm_standby(8) (page 263) for a detailed introduction to this tool's usage and command syntax.

cibadmin (8)

cibadmin — Provides direct access to the cluster configuration

Synopsis

Allows the configuration, or sections of it, to be queried, modified, replaced and/or deleted.

```
cibadmin (--query|-Q) -[Vrwlsmfbp] [-i xml-object-id|-o
  xml-object-type] [-t t-flag-whatever] [-h hostname]
```

```
cibadmin (--create|-C) -[Vrwlsmfbp] [-X xml-string]
  [-x xml- filename] [-t t-flag-whatever] [-h hostname]
```

```
cibadmin (--replace-R)  -[Vrwlsmfbp]  [-i xml-object-id|
  -o xml-object-type]  [-X  xml-string] [-x xml-filename] [-t
  t-flag- whatever] [-h hostname]
```

```
cibadmin (--update|-U) -[Vrwlsmfbp]  [-i xml-object-id|
  -o xml-object-type]  [-X  xml-string] [-x xml-filename] [-t
  t-flag- whatever] [-h hostname]
```

```
cibadmin (--modify|-M) -[Vrwlsmfbp]  [-i xml-object-id|
           -o xml-object-type]  [-X  xml-string] [-x xml-filename] [-t
           t-flag- whatever] [-h hostname]
```

```
cibadmin (--delete|-D) -[Vrwlsmfbp]  [-i xml-object-id|
           -o xml-object-type] [-t t-flag-whatever] [-h hostname]
```

```
cibadmin (--delete_alt|-d) -[Vrwlsmfbp] -o
           xml-object-type [-X xml-string|-x xml-filename]
           [-t t-flag-whatever]  [-h hostname]
```

```
cibadmin --erase (-E)
```

```
cibadmin --bump (-B)
```

```
cibadmin --ismaster (-m)
```

```
cibadmin --master (-w)
```

```
cibadmin --slave (-r)
```

```
cibadmin --sync (-S)
```

```
cibadmin --help (-?)
```

Description

The `cibadmin` command is the low-level administrative command for manipulating the Heartbeat CIB. Use it to dump all or part of the CIB, update all or part of it, modify all or part of it, delete the entire CIB, or perform miscellaneous CIB administrative operations.

`cibadmin` operates on the XML trees of the CIB, largely without knowledge of the purpose of the updates or queries performed. This means that shortcuts that seem natural to users who understand the meaning of the elements in the XML tree are impossible to use with `cibadmin`. It requires a complete lack of ambiguity and can only deal with valid XML subtrees (tags and elements) for both input and output.

NOTE

`cibadmin` should always be used in preference to editing the `cib.xml` file by hand—especially if the cluster is active. The cluster goes to great lengths to detect and discourage this practice so that your data is not lost or corrupted.

Options

`--obj_type` *`object-type`*, `-o` *`object-type`*

Specify the type of object on which to operate. Valid values are `nodes`, `resources`, `constraints`, `crm_status`, and `status`.

`--verbose`, `-V`

Turn on debug mode. Additional `-V` options increase the detail and frequency of the output.

`--help`, `-?`

Obtain a help message from `cibadmin`.

`--xpath` *`PATHSPEC`*, `-A` *`PATHSPEC`*

Supply a valid XPath to use instead of an obj_type.

Commands

`--bump`, `-B`
: Increase the `epoch` version counter in the CIB. Normally this value is increased automatically by the cluster when a new leader is elected. Manually increasing it can be useful if you want to make an older configuration obsolete (such as one stored on inactive cluster nodes).

`--create`, `-C`
: Create a new CIB from the XML content of the argument.

`--delete`, `-D`
: Delete the first object matching the supplied criteria, for example, `<op id="rsc1_op1" name="monitor"/>`. The tag name and all attributes must match in order for the element to be deleted

`--erase`, `-E`
: Erase the contents of the entire CIB.

`--ismaster`, `-m`
: Print a message indicating whether or not the local instance of the CIB software is the master instance or not. Exits with return code 0 if it is the master instance or 35 if not.

`--modify`, `-M`
: Find the object somewhere in the CIB's XML tree and update it.

`--query`, `-Q`
: Query a portion of the CIB.

`--replace`, `-R`
: Recursively replace an XML object in the CIB.

`--sync`, `-S`
: Force a resync of all nodes with the CIB on the specified host (if `-h` is used) or with the DC (if no `-h` option is used).

XML Data

`--xml-text` *`string`*, `-X` *`string`*
: Specify an XML tag or fragment on which crmadmin should operate. It must be a complete tag or XML fragment.

`--xml-file` *`filename`*, `-x` *`filename`*
: Specify the XML from a file on which cibadmin should operate. It must be a complete tag or an XML fragment.

`--xml_pipe`, `-p`
: Specify that the XML on which `cibadmin` should operate comes from standard input. It must be a complete tag or an XML fragment.

Advanced Options

`--host` *`hostname`*, `-h` *`hostname`*
: Send command to specified host. Applies to `query` and `sync` commands only.

`--local`, `-l`
: Let a command take effect locally (rarely used, advanced option).

`--no-bcast`, `-b`
: Command will not be broadcast even if it altered the CIB.

> **IMPORTANT**
>
> Use this option with care to avoid ending up with a divergent cluster.

`--sync-call`, `-s`
: Wait for call to complete before returning.

Examples

To get a copy of the entire active CIB (including status section, etc.) delivered to stdout, issue this command:

```
cibadmin -Q
```

To add an IPaddr2 resource to the *resources* section, first create a file `foo` with the following contents:

```
<primitive id="R_10.10.10.101" class="ocf" type="IPaddr2"
 provider="heartbeat">
 <instance_attributes id="RA_R_10.10.10.101">
  <attributes>
   <nvpair id="R_ip_P_ip" name="ip" value="10.10.10.101"/>
   <nvpair id="R_ip_P_nic" name="nic" value="eth0"/>
  </attributes>
 </instance_attributes>
</primitive>
```

Then issue the following command:

```
cibadmin --obj_type resources -U -x foo
```

To change the IP address of the IPaddr2 resource previously added, issue the command below:

```
cibadmin -M -X '<nvpair id="R_ip_P_ip" name="ip" value="10.10.10.102"/>'
```

NOTE

This does not change the resource name to match the new IP address. To do that, delete then re-add the resource with a new ID tag.

To stop (disable) the IP address resource added previously, and without removing it, create a file called `bar` with the following content in it:

```
<primitive id="R_10.10.10.101">
 <instance_attributes id="RA_R_10.10.10.101">
  <attributes>
   <nvpair id="stop_R_10.10.10.101" name="target-role" value="Stopped"/>
  </attributes>
 </instance_attributes>
</primitive>
```

Then issue the following command:

```
cibadmin --obj_type resources -U -x bar
```

To restart the IP address resource stopped by the previous step, issue:

```
cibadmin -D -X '<nvpair id="stop_R_10.10.10.101">'
```

To completely remove the IP address resource from the CIB, issue this command:

```
cibadmin -D -X '<primitive id="R_10.10.10.101"/>'
```

To replace the CIB with a new manually-edited version of the CIB, use the following command:

```
cibadmin -R -x $HOME/cib.xml
```

Files

`/var/lib/heartbeat/crm/cib.xml`—the CIB (minus status section) on disk.

See Also

crm_resource(8) (page 242), crmadmin(8) (page 224), lrmadmin(8), heartbeat(8)

Caveats

Avoid working on the automatically maintained copy of the CIB on the local disk. Whenever anything in the cluster changes, the CIB is updated. Therefore using an outdated backup copy of the CIB to propagate your configuration changes might result in an inconsistent cluster.

crmadmin (8)

crmadmin — controls the Cluster Resource Manager

Synopsis

```
crmadmin [-V|-q] [-i|-d|-K|-S|-E] node
```

```
crmadmin [-V|-q] -N -B
```

```
crmadmin [-V|-q] -D
```

```
crmadmin -v
```

```
crmadmin -?
```

Description

`crmadmin` was originally designed to control most of the actions of the CRM daemon. However, the largest part of its functionality has been made obsolete by other tools, such as `crm_attribute` and `crm_resource`. Its remaining functionality is mostly related to testing and the status of the crmd process.

WARNING

Some `crmadmin` options are geared towards testing and cause trouble if used incorrectly. In particular, do not use the `--kill` or `--election` options unless you know exactly what you are doing.

Options

`--help`, `-?`
Print the help text.

`--version`, `-v`
Print version details for HA, CRM, and CIB feature set.

`--verbose`, `-V`
Turn on command debug information.

NOTE

Increase the level of verbosity by providing additional instances.

`--quiet`, `-q`
Do not provide any debug information at all and reduce the output to a minimum.

`--bash-export`, `-B`
Create bash export entries of the form `export uname=uuid`. This applies only to the `crmadmin -N` *`node`* command.

NOTE

The `-B` functionality is rarely useful and may be removed in future versions.

Commands

`--debug_inc` *`node`*, `-i` *`node`*
Incrementally increase the CRM daemon's debug level on the specified node. This can also be achieved by sending the USR1 signal to the crmd process.

`--debug_dec` *`node`*, `-d` *`node`*
Incrementally decrease the CRM daemon's debug level on the specified node. This can also be achieved by sending the USR2 signal to the crmd process.

`--kill` *`node`*, `-K` *`node`*
Shut down the CRM daemon on the specified node.

WARNING

Use this with extreme caution. This action should normally only be issued by Heartbeat and may have unintended side effects.

`--status` *`node`*, `-S` *`node`*
Query the status of the CRM daemon on the specified node.

The output includes a general heath indicator and the internal FSM state of the crmd process. This can be helpful when determining what the cluster is doing.

`--election` *`node`*, `-E` *`node`*
: Initiate an election from the specified node.

> **WARNING**
>
> Use this with extreme caution. This action is normally initiated internally and may have unintended side effects.

`--dc_lookup`, `-D`
: Query the uname of the current DC.

  The location of the DC is only of significance to the crmd internally and is rarely useful to administrators except when deciding on which node to examine the logs.

`--nodes`, `-N`
: Query the uname of all member nodes. The results of this query may include nodes in `offline` mode.

> **NOTE**
>
> The `-i`, `-d`, `-K`, and `-E` options are rarely used and may be removed in future versions.

See Also

crm_attribute(8) (page 227), crm_resource(8) (page 242)

crm_attribute (8)

crm_attribute — Allows node attributes and cluster options to be queried, modified and deleted

Synopsis

```
crm_attribute [options]
```

Description

The `crm_attribute` command queries and manipulates node attributes and cluster configuration options that are used in the CIB.

Options

`--help`, `-?`
: Print a help message.

`--verbose`, `-V`
: Turn on debug information.

> **NOTE**
>
> Increase the level of verbosity by providing additional instances.

`--quiet`, `-Q`
: When doing an attribute query using `-G`, print just the value to stdout. Use this option with `-G`.

`--get-value`, `-G`
: Retrieve, rather than set, the preference.

`--delete-attr`, `-D`
: Delete, rather than set, the attribute.

`--attr-id` *`string`*, `-i` *`string`*
: For advanced users only. Identifies the id attribute.

`--attr-value` *`string`*, `-v` *`string`*
: Value to set. This is ignored when used with `-G`.

`--node` *`node_name`*, `-N` *`node_name`*
: The uname of the node to change

`--set-name` *`string`*, `-s` *`string`*
: Specify the set of attributes in which to read or write the attribute.

`--attr-name` *`string`*, `-n` *`string`*
: Specify the attribute to set or query.

`--type` *`string`*, `-t` *`type`*
: Determine to which section of the CIB the attribute should be set or to which section of the CIB the attribute that is queried belongs. Possible values are `nodes`, `status`, or `crm_config`.

Examples

Query the value of the `location` attribute in the `nodes` section for the host *`myhost`* in the CIB:

```
crm_attribute -G -t nodes -U myhost -n location
```

Query the value of the `cluster-delay` attribute in the `crm_config` section in the CIB:

```
crm_attribute -G -t crm_config -n cluster-delay
```

Query the value of the `cluster-delay` attribute in the `crm_config` section in the CIB. Print just the value:

```
crm_attribute -G -Q -t crm_config -n cluster-delay
```

Delete the `location` attribute for the host *`myhost`* from the `nodes` section of the CIB:

```
crm_attribute -D -t nodes -U myhost -n location
```

Add a new attribute called `location` with the value of `office` to the *`set`* subsection of the `nodes` section in the CIB (settings applied to the host *`myhost`*):

```
crm_attribute -t nodes -U myhost -s set -n location -v office
```

Change the value of the `location` attribute in the `nodes` section for the *`myhost`* host:

```
crm_attribute -t nodes -U myhost -n location -v backoffice
```

Files

`/var/lib/heartbeat/crm/cib.xml`—the CIB (minus status section) on disk. Editing this file directly is strongly discouraged.

See Also

cibadmin(8) (page 218)

crm_diff (8)

crm_diff — identify changes to the cluster configuration and apply patches to the configuration files

Synopsis

```
crm_diff [-?|-V] [-o filename] [-O string] [-p filename] [-n filename] [-N
string]
```

Description

The `crm_diff` command assists in creating and applying XML patches. This can be useful for visualizing the changes between two versions of the cluster configuration or saving changes so they can be applied at a later time using `cibadmin`.

Options

`--help`, `-?`
: Print a help message.

`--original` *`filename`*, `-o` *`filename`*
: Specify the original file against which to diff or apply patches.

`--new` *`filename`*, `-n` *`filename`*
: Specify the name of the new file.

`--original-string` *`string`*, `-O` *`string`*
: Specify the original string against which to diff or apply patches.

`--new-string` *`string`*, `-N` *`string`*
: Specify the new string.

`--patch` *`filename`*, `-p` *`filename`*
: Apply a patch to the original XML. Always use with `-o`.

`--cib`, `-c`
: Compare or patch the inputs as a CIB. Always specify the base version with `-o` and provide either the patch file or the second version with `-p` or `-n`, respectively.

`--stdin`, `-s`
: Read the inputs from stdin.

Examples

Use crm_diff to determine the differences between various CIB configuration files and to create patches. By means of patches, easily reuse configuration parts without having to use the cibadmin command on every single one of them.

1 Obtain the two different configuration files by running cibadmin on the two cluster setups to compare:

```
cibadmin -Q > cib1.xml
cibadmin -Q > cib2.xml
```

2 Determine whether to diff the entire files against each other or compare just a subset of the configurations.

3 To print the difference between the files to stdout, use the following command:

```
crm_diff -o cib1.xml -n cib2.xml
```

4 To print the difference between the files to a file and create a patch, use the following command:

```
crm_diff -o cib1.xml -n cib2.xml > patch.xml
```

5 Apply the patch to the original file:

```
crm_diff -o cib1.xml -p patch.xml
```

Files

`/var/lib/heartbeat/crm/cib.xml`—the CIB (minus status section) on disk. Editing this file directly is strongly discouraged.

See Also

cibadmin(8) (page 218)

crm_failcount (8)

crm_failcount — Manage the counter recording each resource's failures

Synopsis

```
crm_failcount [-?|-V] -D -u|-U node -r resource

crm_failcount [-?|-V] -G -u|-U node -r resource

crm_failcount [-?|-V] -v string -u|-U node -r resource
```

Description

Heartbeat implements a sophisticated method to compute and force failover of a resource to another node in case that resource tends to fail on the current node. A resource carries a `resource-stickiness` attribute to determine how much it prefers to run on a certain node. It also carries a `migration-threshold` that determines the threshold at which the resource should failover to another node.

The `failcount` attribute is added to the resource and increased on resource monitoring failure. The value of `failcount` multiplied by the value of `migration-threshold` determines the *failover score* of this resource. If this number exceeds the preference set for this resource, the resource is moved to another node and not run again on the original node until the failure count is reset.

The `crm_failcount` command queries the number of failures per resource on a given node. This tool can also be used to reset the failcount, allowing the resource to run again on nodes where it had previously failed too many times.

Options

`--help`, `-?`
  Print a help message.

`--verbose`, `-V`
  Turn on debug information.

NOTE

Increase the level of verbosity by providing additional instances.

`--quiet`, `-Q`
: When doing an attribute query using `-G`, print just the value to stdout. Use this option with `-G`.

`--get-value`, `-G`
: Retrieve rather than set the preference.

`--delete-attr`, `-D`
: Specify the attribute to delete.

`--attr-id` *`string`*, `-i` *`string`*
: For advanced users only. Identifies the id attribute.

`--attr-value` *`string`*, `-v` *`string`*
: Specify the value to use. This option is ignored when used with `-G`.

`--node` *`node_uname`*, `-U` *`node_uname`*
: Specify the uname of the node to change.

`--resource-id` *`resource name`*, `-r` *`resource name`*
: Specify the name of the resource on which to operate.

Examples

Reset the failcount for the resource `myrsc` on the node `node1`:

```
crm_failcount -D -U node1 -r my_rsc
```

Query the current failcount for the resource `myrsc` on the node `node1`:

```
crm_failcount -G -U node1 -r my_rsc
```

Files

`/var/lib/heartbeat/crm/cib.xml`—the CIB (minus status section) on disk. Editing this file directly is strongly discouraged.

See Also

crm_attribute(8) (page 227) and cibadmin(8) (page 218)

crm_master (8)

crm_master — Manage a master/slave resource's preference for being promoted on a given node

Synopsis

```
crm_master [-V|-Q] -D [-l lifetime]

crm_master [-V|-Q] -G [-l lifetime]

crm_master [-V|-Q] -v string [-l string]
```

Description

`crm_master` is called from inside the resource agent scripts to determine which resource instance should be promoted to master mode. It should never be used from the command line and is just a helper utility for the resource agents. RAs use `crm_master` to promote a particular instance to master mode or to remove this preference from it. By assigning a lifetime, determine whether this setting should survive a reboot of the node (set lifetime to `forever`) or whether it should not survive a reboot (set lifetime to `reboot`).

A resource agent needs to determine on which resource `crm_master` should operate. These queries must be handled inside the resource agent script. The actual calls of `crm_master` follow a syntax similar to those of the `crm_attribute` command.

Options

`--help`, `-?`
: Print a help message.

`--verbose`, `-V`
: Turn on debug information.

NOTE

Increase the level of verbosity by providing additional instances.

`--quiet`, `-Q`
: When doing an attribute query using `-G`, print just the value to stdout. Use this option with `-G`.

`--get-value`, `-G`
: Retrieve rather than set the preference to be promoted.

`--delete-attr`, `-D`
: Delete rather than set the attribute.

`--attr-id` *`string`*, `-i` *`string`*
: For advanced users only. Identifies the id attribute.

`--attr-value` *`string`*, `-v` *`string`*
: Value to set. This is ignored when used with `-G`.

`--lifetime` *`string`*, `-l` *`string`*
: Specify how long the preference lasts. Possible values are `reboot` or `forever`.

Environment Variables

`OCF_RESOURCE_INSTANCE`—the name of the resource instance

Files

`/var/lib/heartbeat/crm/cib.xml`—the CIB (minus status section) on disk.

See Also

cibadmin(8) (page 218), crm_attribute(8) (page 227)

crm_mon (8)

crm_mon — monitor the cluster's status

Synopsis

```
crm_mon [-V] -d -pfilename -h filename
```

```
crm_mon [-V] [-1|-n|-r] -h filename
```

```
crm_mon [-V] [-n|-r] -X filename
```

```
crm_mon [-V] [-n|-r] -c|-1
```

```
crm_mon [-V] -i interval
```

```
crm_mon -?
```

Description

The `crm_mon` command allows you to monitor your cluster's status and configuration. Its output includes the number of nodes, uname, uuid, status, the resources configured in your cluster, and the current status of each. The output of `crm_mon` can be displayed at the console or printed into an HTML file. When provided with a cluster configuration file without the status section, `crm_mon` creates an overview of nodes and resources as specified in the file.

Options

`--help`, `-?`
: Provide help.

`--verbose`, `-V`
: Increase the debug output.

`--interval` *`seconds`*, `-i` *`seconds`*
: Determine the update frequency. If `-i` is not specified, the default of 15 seconds is assumed.

`--group-by-node`, `-n`
: Group resources by node.

`--inactive`, `-r`
: Display inactive resources.

`--simple-status`, `-s`
: Display the cluster status once as a simple one line output (suitable for nagios).

`--one-shot`, `-1`
: Display the cluster status once on the console then exit (does not use ncurses).

`--as-html` *`filename`*, `-h` *`filename`*
: Write the cluster's status to the specified file.

`--web-cgi`, `-w`
: Web mode with output suitable for CGI.

`--daemonize`, `-d`
: Run in the background as a daemon.

`--pid-file` *`filename`*, `-p` *`filename`*
: Specify the daemon's pid file.

Examples

Display your cluster's status and get an updated listing every 15 seconds:

```
crm_mon
```

Display your cluster's status and get an updated listing after an interval specified by `-i`. If `-i` is not given, the default refresh interval of 15 seconds is assumed:

```
crm_mon -i interval[s]
```

Display your cluster's status on the console:

```
crm_mon -c
```

Display your cluster's status on the console just once then exit:

```
crm_mon -1
```

Display your cluster's status and group resources by node:

```
crm_mon -n
```

Display your cluster's status, group resources by node, and include inactive resources in the list:

```
crm_mon -n -r
```

Write your cluster's status to an HTML file:

```
crm_mon -h filename
```

Run `crm_mon` as a daemon in the background, specify the daemon's pid file for easier control of the daemon process, and create HTML output. This option allows you to constantly create HTML output that can be easily processed by other monitoring applications:

```
crm_mon -d -p filename -h filename
```

Display the cluster configuration laid out in an existing cluster configuration file (*`filename`*), group the resources by node, and include inactive resources. This command can be used for dry runs of a cluster configuration before rolling it out to a live cluster.

```
crm_mon -r -n -X filename
```

Files

`/var/lib/heartbeat/crm/cib.xml`—the CIB (minus status section) on disk. Editing this file directly is strongly discouraged.

crm_node (8)

crm_node — Lists the members of a cluster

Synopsis

```
crm_node [-V] [-p|-e|-q]
```

Description

Lists the members of a cluster.

Options

`-V`
: be verbose

`--partition`, `-p`
: print the members of this partition

`--epoch`, `-e`
: print the epoch this node joined the partition

`--quorum`, `-q`
: print a 1 if our partition has quorum

crm_resource (8)

crm_resource — Perform tasks related to cluster resources

Synopsis

```
crm_resource  [-?|-V|-S] -L|-Q|-W|-D|-C|-P|-p [options]
```

Description

The `crm_resource` command performs various resource-related actions on the cluster. It can modify the definition of configured resources, start and stop resources, and delete and migrate resources between nodes.

`--help`, `-?`
: Print the help message.

`--verbose`, `-V`
: Turn on debug information.

> **NOTE**
>
> Increase the level of verbosity by providing additional instances.

`--quiet`, `-Q`
: Print only the value on stdout (for use with `-W`).

Commands

`--list`, `-L`
: List all resources.

`--query-xml`, `-x`
: Query a resource.

  Requires: `-r`

`--locate`, `-W`
: Locate a resource.

  Requires: `-r`

`--migrate`, `-M`
: Migrate a resource from its current location. Use `-N` to specify a destination.

  If `-N` is not specified, the resource is forced to move by creating a rule for the current location and a score of `-INFINITY`.

  NOTE

  This prevents the resource from running on this node until the constraint is removed with `-U`.

  Requires: `-r`, Optional: `-N`, `-f`

`--un-migrate`, `-U`
: Remove all constraints created by `-M`

  Requires: `-r`

`--delete`, `-D`
: Delete a resource from the CIB.

  Requires: `-r`, `-t`

`--cleanup`, `-C`
: Delete a resource from the LRM.

  Requires: `-r`. Optional: `-H`

`--reprobe`, `-P`
: Recheck for resources started outside the CRM.

  Optional: `-H`

`--refresh`, `-R`
: Refresh the CIB from the LRM.

Optional: `-H`

`--set-parameter` *`string`*, `-p` *`string`*
: Set the named parameter for a resource.

Requires: `-r`, `-v`. Optional: `-i`, `-s`, and `--meta`

`--get-parameter` *`string`*, `-g` *`string`*
: Get the named parameter for a resource.

Requires: `-r`. Optional: `-i`, `-s`, and `--meta`

`--delete-parameter` *`string`*, `-d` *`string`*
: Delete the named parameter for a resource.

Requires: `-r`. Optional: `-i`, and `--meta`

`--list-operations` *`string`* , `-O` *`string`*
: List the active resource operations. Optionally filtered by resource, node, or both. Optional: `-N`, `-r`

`--list-all-operations` *`string`* , `-o` *`string`*
: List all resource operations. Optionally filtered by resource, node, or both. Optional: `-N`, `-r`

Options

`--resource` *`string`*, `-r` *`string`*
: Specify the resource ID.

`--resource-type` *`string`*, `-t` *`string`*
: Specify the resource type (`primitive`, `clone`, `group`, etc.).

`--property-value` *`string`*, `-v` *`string`*
: Specify the property value.

`--node` *`string`*, `-N` *`string`*
: Specify the hostname.

`--meta`

Modify a resource's configuration option rather than one which is passed to the resouce agent script. For use with `-p`, `-g` and `-d`.

`--lifetime` *`string`*, `-u` *`string`*

Lifespan of migration constraints.

`--force`, `-f`

Force the resource to move by creating a rule for the current location and a score of `-INFINITY`

This should be used if the resource's stickiness and constraint scores total more than `INFINITY` (currently 100,000).

NOTE

This prevents the resource from running on this node until the constraint is removed with `-U`.

`-s` *`string`*

(Advanced Use Only) Specify the ID of the `instance_attributes` object to change.

`-i` *`string`*

(Advanced Use Only) Specify the ID of the `nvpair` object to change or delete.

Examples

Listing all resources:

```
crm_resource -L
```

Checking where a resource is running (and if it is):

```
crm_resource -W -r my_first_ip
```

If the `my_first_ip` resource is running, the output of this command reveals the node on which it is running. If it is not running, the output shows this.

Start or stop a resource:

```
crm_resource -r my_first_ip -p target_role -v started
```

```
crm_resource -r my_first_ip -p target_role -v stopped
```

Query the definition of a resource:

```
crm_resource -Q -r my_first_ip
```

Migrate a resource away from its current location:

```
crm_resource -M -r my_first_ip
```

Migrate a resource to a specific location:

```
crm_resource -M -r my_first_ip -H c001n02
```

Allow a resource to return to its normal location:

```
crm_resource -U -r my_first_ip
```

NOTE

The values of `resource_stickiness` and `default_resource_stickiness` may mean that it does not move back. In such cases, you should use `-M` to move it back before running this command.

Delete a resource from the CRM:

```
crm_resource -D -r my_first_ip -t primitive
```

Delete a resource group from the CRM:

```
crm_resource -D -r my_first_group -t group
```

Disable resource management for a resource in the CRM:

```
crm_resource -p is-managed -r my_first_ip -t primitive -v off
```

Enable resource management for a resource in the CRM:

```
crm_resource -p is-managed -r my_first_ip -t primitive -v on
```

Reset a failed resource after having been manually cleaned up:

```
crm_resource -C -H c001n02 -r my_first_ip
```

Recheck all nodes for resources started outside the CRM:

```
crm_resource -P
```

Recheck one node for resources started outside the CRM:

```
crm_resource -P -H c001n02
```

Files

/var/lib/heartbeat/crm/cib.xml—the CIB (minus status section) on disk. Editing this file directly is strongly discouraged.

See Also

cibadmin(8) (page 218), crmadmin(8) (page 224), lrmadmin(8), heartbeat(8)

crm_shadow (8)

crm_shadow — Perform Configuration Changes in a Sandbox Before Updating The Live Cluster

Synopsis

```
crm_shadow [-V] [-p|-e|-q]
```

Description

Sets up an environment in which configuration tools (`cibadmin`, `crm_resource`, etc) work offline instead of against a live cluster, allowing changes to be previewed and tested for side-effects.

Options

`--verbose`, `-V`
: turn on debug info. additional instance increase verbosity

`--which`, `-w`
: indicate the active shadow copy

`--display`, `-p`
: display the contents of the shadow copy

`--diff`, `-d`
: display the changes in the shadow copy

`--create-empty`, `-e`*`NAME`*
: create the named shadow copy with an empty cluster configuration

`--create`, `-c`*`NAME`*
: create the named shadow copy of the active cluster configuration

`--reset`, `-r`*NAME*
: recreate the named shadow copy from the active cluster configuration

`--commit`, `-c`*NAME*
: upload the contents of the named shadow copy to the cluster

`--delete`, `-d`*NAME*
: delete the contents of the named shadow copy

`--edit`, `-e`*NAME*
: Edit the contents of the named shadow copy with your favorite editor

`--batch`, `-b`
: do not spawn a new shell

`--force`, `-f`
: do not spawn a new shell

`--switch`, `-s`
: switch to the named shadow copy

Internal Commands

To work with a shadow configuration, you need to create one first:

```
crm_shadow --create-empty YOUR_NAME
```

It gives you an internal shell like the one from the `crm` tool. Use `help` to get an overview of all internal commands, or `help` *`subcommand`* for a specific command.

Table 18.1 *Overview of Internal Commands*

Command	Syntax/Description
`alias`	`alias [-p] [name[=value] ... ]` `alias` with no arguments or with the -p option prints the list of aliases in the form alias `NAME=VALUE` on standard output. Otherwise, an alias is defined for each NAME whose VALUE is given. A trailing space in VALUE causes the next word to be checked for alias substi-

Command	Syntax/Description
	tution when the alias is expanded. Alias returns true unless a NAME is given for which no alias has been defined.
`bg`	`bg [JOB_SPEC ...]` Place each JOB_SPEC in the background, as if it had been started with &. If JOB_SPEC is not present, the shell's notion of the current job is used.

`bind`	`bind [-lpvsPVS] [-m keymap] [-f filename]`
	`[-q name] [-u name] [-r keyseq]`
	`[-x keyseq:shell-command]`
	`[keyseq:readline-function or readline-command]`
	Bind a key sequence to a Readline function or a macro, or set a Readline variable. The non-option argument syntax is equivalent to that found in `~/.inputrc`, but must be passed as a single argument: bind `"\C-x\C-r": re-read-init-file`.

`break`	`break [N]` Exit from within a `for`, `while` or `until` loop. If N is specified, break N levels.
`builtin`	`builtin [shell-builtin [arg ...]]` Run a shell builtin. This is useful when you wish to rename a shell builtin to be a function, but need the functionality of the builtin within the function itself.
`caller`	`caller [EXPR]` Returns the context of the current subroutine call. Without *`EXPR`*, returns `$line $filename`. With *`EXPR`*, returns `$line $subroutine $filename`; this extra information can be used to provide a stack trace.
`case`	`case WORD in [PATTERN [| PATTERN] [COMMANDS;;] ... esac`

Command	Syntax/Description
	Selectively execute *COMMANDS* based upon *WORD* matching *PATTERN*. The `|' is used to separate multiple patterns.
`cd`	`cd [-L|-P] [dir]` Change the current directory to DIR.

`command`	`command [-pVv]`
	`command [arg ...]`
	Runs *COMMAND* with *ARGS* ignoring shell functions. If you have a shell function called `ls', and you wish to call the command `ls', you can say "command ls". If the -p option is given, a default value is used for PATH that is guaranteed to find all of the standard utilities. If the -V or -v option is given, a string is printed describing COMMAND. The -V option produces a more verbose description.

`compgen`	`compgen [-abcdefgjksuv] [-o option] [-A action]`
	`  [-G globpat] [-W wordlist] [-P prefix]`
	`  [-S suffix] [-X filterpat] [-F function]`
	`  [-C command] [WORD]`
	Display the possible completions depending on the options. Intended to be used from within a shell function generating possible completions. If the optional *WORD* argument is supplied, matches against *WORD* are generated.

`complete`	`complete [-abcdefgjksuv] [-pr] [-o option]`
	`   [-A action] [-G globpat] [-W wordlist] [-P prefix]`
	`   [-S suffix] [-X filterpat] [-F function] [-C command]`
	`   [name ...]`
	For each *NAME*, specify how arguments are to be completed. If the `-p` option is supplied, or if no options are supplied, existing completion specifications are printed in a way that allows them to be reused as input. The `-r` option removes a completion specification for each *NAME*, or, if no *NAME*s are supplied, all completion specifications.

`continue`	`continue [N]`

Command	Syntax/Description
	Resume the next iteration of the enclosing FOR, WHILE or UNTIL loop. If *N* is specified, resume at the *N*-th enclosing loop.
`declare`	`declare [-afFirtx] [-p] [name[=value] ...]` Declare variables and/or give them attributes. If no *NAME*s are given, then display the values of variables instead. The `-p` option will display the attributes and values of each *NAME*.
`dirs`	`dirs [-clpv] [+N] [-N]` Display the list of currently remembered directories. Directories find their way onto the list with the `pushd` command; you can get back up through the list with the `popd` command.
`disown`	`disown [-h] [-ar] [JOBSPEC ...]` By default, removes each *JOBSPEC* argument from the table of active jobs. If the `-h` option is given, the job is not removed from the table, but is marked so that SIGHUP is not sent to the job if the shell receives a SIGHUP. The `-a` option, when *JOBSPEC* is not supplied, means to remove all jobs from the job table; the `-r` option means to remove only running jobs.
`echo`	`echo [-neE] [arg ...]` Output the ARGs. If -n is specified, the trailing newline is suppressed. If the -e option is given, interpretation of the following backslash-escaped characters is turned on: `\a` (alert, bell) `\b` (backspace) `\c` (suppress trailing newline) `\E` (escape character) `\f` (form feed) `\n` (new line)

Command	Syntax/Description
	`\r` (carriage return) `\t` (horizontal tab) `\v` (vertical tab) `\\` (backslash) `\0nnn` (the character whose ASCII code is NNN (octal). NNN can be 0 to 3 octal digits) You can turn off the interpretation of the above characters with the `-E` option.
`enable`	`enable [-pnds] [-a] [-f filename] [name...]` Enable and disable builtin shell commands. This allows you to use a disk command which has the same name as a shell builtin without specifying a full pathname. If `-n` is used, the *NAME*s become disabled; otherwise *NAME*s are enabled. For example, to use the `test` found in `$PATH` instead of the shell builtin version, type `enable -n test`. On systems supporting dynamic loading, the `-f` option may be used to load new builtins from the shared object *FILENAME*. The `-d` option will delete a builtin previously loaded with `-f`. If no non-option names are given, or the `-p` option is supplied, a list of builtins is printed. The `-a` option means to print every builtin with an indication of whether or not it is enabled. The `-s` option restricts the output to the POSIX.2 `special' builtins. The `-n` option displays a list of all disabled builtins.
`eval`	`eval [ARG ...]` Read *ARG*s as input to the shell and execute the resulting command(s).
`exec`	`exec [-cl] [-a name] file [redirection ...]` Exec *FILE*, replacing this shell with the specified program. If *FILE* is not specified, the redirections take effect in this shell. If the first argument is `-l`, then place a dash in the zeroth arg passed to *FILE*, as login does. If the `-c` option is supplied, FILE is executed with a null environment. The `-a` option means to make set argv[0] of the executed

Command	Syntax/Description
	process to *NAME*. If the file cannot be executed and the shell is not interactive, then the shell exits, unless the shell option `execfail` is set.
`exit`	`exit [N]` Exit the shell with a status of *N*. If *N* is omitted, the exit status is that of the last command executed.

`export`	`export [-nf] [NAME[=value] ...]`
	`export -p`
	*NAME*s are marked for automatic export to the environment of subsequently executed commands. If the `-f` option is given, the *NAME*s refer to functions. If no *NAME*s are given, or if `-p` is given, a list of all names that are exported in this shell is printed. An argument of `-n` says to remove the export property from subsequent *NAME*s. An argument of `--` disables further option processing.

`false`	`false` Return an unsuccessful result.

`fc`	`fc [-e ename] [-nlr] [FIRST] [LAST]`
	`fc -s [pat=rep] [cmd]`
	fc is used to list or edit and re-execute commands from the history list. *FIRST* and *LAST* can be numbers specifying the range, or *FIRST* can be a string, which means the most recent command beginning with that string.

`fg`	`fg [JOB_SPEC]` Place *JOB_SPEC* in the foreground, and make it the current job. If *JOB_SPEC* is not present, the shell's notion of the current job is used.
`for`	`for NAME [in WORDS ... ;] do COMMANDS; done`

Command	Syntax/Description
	The `for` loop executes a sequence of commands for each member in a list of items. If `in WORDS ...;` is not present, then `in "$@"` is assumed. For each element in *`WORDS`*, *`NAME`* is set to that element, and the *`COMMANDS`* are executed.

`function`	`function NAME { COMMANDS ; }`
	`function NAME () { COMMANDS ; }`
	Create a simple command invoked by *`NAME`* which runs *`COMMANDS`*. Arguments on the command line along with *`NAME`* are passed to the function as `$0` .. `$n`.

`getopts`	`getopts OPTSTRING NAME [arg]` Getopts is used by shell procedures to parse positional parameters.
`hash`	`hash [-lr] [-p PATHNAME] [-dt] [NAME...]` For each *`NAME`*, the full pathname of the command is determined and remembered. If the `-p` option is supplied, *`PATHNAME`* is used as the full pathname of *`NAME`*, and no path search is performed. The `-r` option causes the shell to forget all remembered locations. The `-d` option causes the shell to forget the remembered location of each *`NAME`*. If the `-t` option is supplied the full pathname to which each *`NAME`* corresponds is printed. If multiple *`NAME`* arguments are supplied with `-t`, the *`NAME`* is printed before the hashed full pathname. The `-l` option causes output to be displayed in a format that may be reused as input. If no arguments are given, information about remembered commands is displayed.

`history`	`history [-c] [-d OFFSET] [n]`
	`history -ps arg [arg...]`
	`history -awrm [filename]`
	Display the history list with line numbers. Lines listed with with a `*` have been modified. Argument of *`N`* says to list only the last *`N`* lines. The `-c` option causes the history list to be cleared by deleting all of

Command	Syntax/Description
	the entries. The `-d` option deletes the history entry at offset *`OFFSET`*. The `-w` option writes out the current history to the history file; `-r` means to read the file and append the contents to the history list instead. `-a` means to append history lines from this session to the history file. Argument `-n` means to read all history lines not already read from the history file and append them to the history list.

`jobs`	`jobs [-lnprs] [JOBSPEC ...]`
	`job -x COMMAND [ARGS]`
	Lists the active jobs. The `-l` option lists process id's in addition to the normal information; the `-p` option lists process id's only. If `-n` is given, only processes that have changed status since the last notification are printed. *`JOBSPEC`* restricts output to that job. The `-r` and `-s` options restrict output to running and stopped jobs only, respectively. Without options, the status of all active jobs is printed. If `-x` is given, *`COMMAND`* is run after all job specifications that appear in *`ARGS`* have been replaced with the process ID of that job's process group leader.

`kill`	`kill [-s sigspec | -n signum | -sigspec] pid | JOBSPEC ...`
	`kill -l [sigspec]`
	Send the processes named by PID (or *`JOBSPEC`*) the signal SIGSPEC. If SIGSPEC is not present, then SIGTERM is assumed. An argument of `-l` lists the signal names; if arguments follow `-l` they are assumed to be signal numbers for which names should be listed. Kill is a shell builtin for two reasons: it allows job IDs to be used instead of process IDs, and, if you have reached the limit on processes that you can create, you don't have to start a process to kill another one.

`let`	`let ARG [ARG ...]` Each *`ARG`* is a mathematical expression to be evaluated. Evaluation is done in fixed-width integers with no check for overflow, though division by 0 is trapped and flagged as an error. The following list of operators is grouped into levels of equal-precedence operators. The levels are listed in order of decreasing precedence.

Command	Syntax/Description
`local`	`local NAME[=VALUE] ...` Create a local variable called *NAME*, and give it *VALUE*. `local` can only be used within a function; it makes the variable *NAME* have a visible scope restricted to that function and its children.
`logout`	`logout` Logout of a login shell.
`popd`	`popd [+N | -N] [-n]` Removes entries from the directory stack. With no arguments, removes the top directory from the stack, and cd's to the new top directory.
`printf`	`printf [-v var] format [ARGUMENTS]` printf formats and prints *ARGUMENTS* under control of the *FORMAT*. *FORMAT* is a character string which contains three types of objects: plain characters, which are simply copied to standard output, character escape sequences which are converted and copied to the standard output, and format specifications, each of which causes printing of the next successive argument. In addition to the standard printf(1) formats, `%b` means to expand backslash escape sequences in the corresponding argument, and `%q` means to quote the argument in a way that can be reused as shell input. If the `-v` option is supplied, the output is placed into the value of the shell variable VAR rather than being sent to the standard output.
`pushd`	`pushd [dir | +N | -N] [-n]` Adds a directory to the top of the directory stack, or rotates the stack, making the new top of the stack the current working directory. With no arguments, exchanges the top two directories.
`pwd`	`pwd [-LP]`

Command	Syntax/Description
	Print the current working directory. With the `-P` option, `pwd` prints the physical directory, without any symbolic links; the `-L` option makes `pwd` follow symbolic links.
`read`	`read [-ers] [-u fd] [-t timeout] [-p prompt] [-a array] [-n nchars] [-d delim] [NAME ...]` The given *NAME*s are marked readonly and the values of these *NAME*s may not be changed by subsequent assignment. If the `-f` option is given, then functions corresponding to the NAMEs are so marked. If no arguments are given, or if `-p` is given, a list of all readonly names is printed. The `-a` option means to treat each NAME as an array variable. An argument of `--` disables further option processing.

`readonly`	`readonly [-af] [NAME[=VALUE] ...]`
	`readonly -p`
	The given *NAME*s are marked readonly and the values of these *NAME*s may not be changed by subsequent assignment. If the `-f` option is given, then functions corresponding to the *NAME*s are so marked. If no arguments are given, or if `-p` is given, a list of all readonly names is printed. The `-a` option means to treat each *NAME* as an array variable. An argument of `--` disables further option processing.

`return`	`return [N]` Causes a function to exit with the return value specified by *N*. If *N* is omitted, the return status is that of the last command.
`select`	`select NAME [in WORDS ... ;] do COMMANDS; done` The *WORDS* are expanded, generating a list of words. The set of expanded words is printed on the standard error, each preceded by a number. If `in WORDS` is not present, `in "$@"` is assumed. The `PS3` prompt is then displayed and a line read from the standard input. If the line consists of the number corresponding to one of the displayed words, then *NAME* is set to that word. If the line is empty, *WORDS* and

Command	Syntax/Description
	the prompt are redisplayed. If EOF is read, the command completes. Any other value read causes *NAME* to be set to null. The line read is saved in the variable *REPLY*. *COMMANDS* are executed after each selection until a break command is executed.
`set`	`set [--abefhkmnptuvxBCHP] [-o OPTION] [ARG...]` Sets internal shell options.
`shift`	`shift [n]` The positional parameters from `$N+1 ...` are renamed to `$1 ...` If *N* is not given, it is assumed to be 1.
`shopt`	`shopt [-pqsu] [-o long-option] OPTNAME [OPTNAME...]` Toggle the values of variables controlling optional behavior. The `-s` flag means to enable (set) each *OPTNAME*; the `-u` flag unsets each *OPTNAME*. The `-q` flag suppresses output; the exit status indicates whether each *OPTNAME* is set or unset. The `-o` option restricts the *OPTNAME*s to those defined for use with `set -o`. With no options, or with the `-p` option, a list of all settable options is displayed, with an indication of whether or not each is set.
`source`	`source FILENAME [ARGS]` Read and execute commands from *FILENAME* and return. The pathnames in `$PATH` are used to find the directory containing *FILENAME*. If any *ARGS* are supplied, they become the positional parameters when *FILENAME* is executed.
`suspend`	`suspend [-f]` Suspend the execution of this shell until it receives a SIGCONT signal. The `-f` if specified says not to complain about this being a login shell if it is; just suspend anyway.

Command	Syntax/Description
`test`	`test [expr]` Exits with a status of 0 (true) or 1 (false) depending on the evaluation of *`EXPR`*. Expressions may be unary or binary. Unary expressions are often used to examine the status of a file. There are string operators as well, and numeric comparison operators.
`time`	`time [-p] PIPELINE` Execute *`PIPELINE`* and print a summary of the real time, user CPU time, and system CPU time spent executing *`PIPELINE`* when it terminates. The return status is the return status of PIPELINE. The `-p` option prints the timing summary in a slightly different format. This uses the value of the TIMEFORMAT variable as the output format.
`times`	`times` Print the accumulated user and system times for processes run from the shell.
`trap`	`trap [-lp] [ARG SIGNAL_SPEC ...]` The command *`ARG`* is to be read and executed when the shell receives signal(s) *`SIGNAL_SPEC`*. If *`ARG`* is absent (and a single *`SIGNAL_SPEC`* is supplied) or `-`, each specified signal is reset to its original value. If *`ARG`* is the null string each *`SIGNAL_SPEC`* is ignored by the shell and by the commands it invokes. If a *`SIGNAL_SPEC`* is EXIT (0) the command ARG is executed on exit from the shell. If a *`SIGNAL_SPEC`* is DEBUG, *`ARG`* is executed after every simple command. If the `-p` option is supplied then the trap commands associated with each *`SIGNAL_SPEC`* are displayed. If no arguments are supplied or if only `-p` is given, trap prints the list of commands associated with each signal. Each *`SIGNAL_SPEC`* is either a signal name in `signal.h` or a signal number. Signal names are case insensitive and the SIG prefix is optional. `trap -l` prints a list of signal names and their corresponding numbers. Note that a signal can be sent to the shell with `kill -signal $$`.

Command	Syntax/Description
`true`	`true` Return a successful result.
`type`	`type [-afptP] NAME [NAME ...]` Obsolete, see `declare`.
`typeset`	`typeset [-afFirtx] [-p] name[=value]` Obsolete, see `declare`.
`ulimit`	`ulimit [-SHacdfilmnpqstuvx] [limit` Ulimit provides control over the resources available to processes started by the shell, on systems that allow such control.
`umask`	`umask [-p] [-S] [MODE]` The user file-creation mask is set to *MODE*. If *MODE* is omitted, or if `-S` is supplied, the current value of the mask is printed. The `-S` option makes the output symbolic; otherwise an octal number is output. If `-p` is supplied, and *MODE* is omitted, the output is in a form that may be used as input. If *MODE* begins with a digit, it is interpreted as an octal number, otherwise it is a symbolic mode string like that accepted by chmod(1).
`unalias`	`unalias [-a] NAME [NAME ...]` Remove *NAME*s from the list of defined aliases. If the `-a` option is given, then remove all alias definitions.
`unset`	`unset [-f] [-v] [NAME ...]` For each *NAME*, remove the corresponding variable or function. Given the `-v`, unset will only act on variables. Given the `-f` flag, unset will only act on functions. With neither flag, unset first tries to unset a variable. If that fails, it then tries to unset a function. Some variables cannot be unset; also see readonly.

Command	Syntax/Description
`until`	`until COMMANDS; do COMMANDS; done` Expand and execute *`COMMANDS`* as long as the final command in the `until` *`COMMANDS`* has an exit status which is not zero.
`wait`	`wait [N]` Wait for the specified process and report its termination status. If *`N`* is not given, all currently active child processes are waited for, and the return code is zero. *`N`* may be a process ID or a job specification; if a job spec is given, all processes in the job's pipeline are waited for.
`while`	`while COMMANDS; do COMMANDS; done` Expand and execute *`COMMANDS`* as long as the final command in the `while` *`COMMANDS`* has an exit status of zero.

crm_standby (8)

crm_standby — manipulate a node's standby attribute to determine whether resources can be run on this node

Synopsis

```
crm_standby [-?|-V] -D -u|-U node -r resource

crm_standby  [-?|-V] -G -u|-U node -r resource

crm_standby [-?|-V] -v string -u|-U node -r resource [-l string]
```

Description

The `crm_standby` command manipulates a node's standby attribute. Any node in standby mode is no longer eligible to host resources and any resources that are there must be moved. Standby mode can be useful for performing maintenance tasks, such as kernel updates. Remove the standby attribute from the node when it needs to become a fully active member of the cluster again.

By assigning a lifetime to the `standby` attribute, determine whether the standby setting should survive a reboot of the node (set lifetime to `forever`) or should be reset with reboot (set lifetime to `reboot`). Alternatively, remove the `standby` attribute and bring the node back from standby manually.

Options

`--help`, `-?`
Print a help message.

`--verbose`, `-V`
Turn on debug information.

NOTE

Increase the level of verbosity by providing additional instances.

`--quiet`, `-Q`
: When doing an attribute query using `-G`, print just the value to stdout. Use this option with `-G`.

`--get-value`, `-G`
: Retrieve rather than set the preference.

`--delete-attr`, `-D`
: Specify the attribute to delete.

`--attr-value` *`string`*, `-v` *`string`*
: Specify the value to use. This option is ignored when used with `-G`.

`--attr-id` *`string`*, `-i` *`string`*
: For advanced users only. Identifies the id attribute..

`--node` *`node_uname`*, `-u` *`node_uname`*
: Specify the uname of the node to change.

`--lifetime` *`string`*, `-l` *`string`*
: Determine how long this preference lasts. Possible values are `reboot` or `forever`.

> **NOTE**
>
> If a `forever` value exists, it is always used by the CRM instead of any `reboot` value.

Examples

Have a local node go to standby:

```
crm_standby -v true
```

Have a node (`node1`) go to standby:

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

Query the standby status of a node:

```
crm_standby -G -U node1
```

Remove the standby property from a node:

```
crm_standby -D -U node1
```

Have a node go to standby for an indefinite period of time:

```
crm_standby -v true -l forever -U node1
```

Have a node go to standby until the next reboot of this node:

```
crm_standby -v true -l reboot -U node1
```

Files

`/var/lib/heartbeat/crm/cib.xml`—the CIB (minus status section) on disk. Editing this file directly is strongly discouraged.

See Also

cibadmin(8) (page 218), crm_attribute(8) (page 227)

crm_verify (8)

crm_verify — check the CIB for consistency

Synopsis

```
crm_verify [-V] -x file
```

```
crm_verify [-V] -X string
```

```
crm_verify [-V] -L|-p
```

```
crm_verify [-?]
```

Description

`crm_verify` checks the configuration database (CIB) for consistency and other problems. It can be used to check a file containing the configuration or can it can connect to a running cluster. It reports two classes of problems, errors and warnings. Errors must be fixed before High Availability can work properly. However, it is left up to the administrator to decide if the warnings should also be fixed.

`crm_verify` assists in creating new or modified configurations. You can take a local copy of a CIB in the running cluster, edit it, validate it using `crm_verify`, then put the new configuration into effect using `cibadmin`.

Options

`--help`, `-h`
: Print a help message.

`--verbose`, `-V`
: Turn on debug information.

> **NOTE**
>
> Increase the level of verbosity by providing additional instances.

`--live-check`, `-L`
  Connect to the running cluster and check the CIB.

`--crm_xml` *`string`*, `-X` *`string`*
  Check the configuration in the supplied string. Pass complete CIBs only.

`--xml-file` *`file`*, `-x` *`file`*
  Check the configuration in the named file.

`--xml-pipe`, `-p`
  Use the configuration piped in via stdin. Pass complete CIBs only.

Examples

Check the consistency of the configuration in the running cluster and produce verbose output:

```
crm_verify -VL
```

Check the consistency of the configuration in a given file and produce verbose output:

```
crm_verify -Vx file1
```

Pipe a configuration into `crm_verify` and produce verbose output:

```
cat file1.xml | crm_verify -Vp
```

Files

`/var/lib/heartbeat/crm/cib.xml`—the CIB (minus status section) on disk. Editing this file directly is strongly discouraged.

See Also

cibadmin(8) (page 218)

19 HA OCF Agents

All OCF agents require several parameters to be set when they are started. The following overview shows how to manually operate these agents. The data that is available in this appendix is directly taken from the `meta-data` invocation of the respective RA. Find all these agents in `/usr/lib/ocf/resource.d/heartbeat/`.

When configuring an RA, omit the `OCF_RESKEY_` prefix to the parameter name. Parameters that are in square brackets may be omitted in the configuration.

ocf:anything (7)

ocf:anything — Manages an arbitrary service

Synopsis

`OCF_RESKEY_binfile=`string [`OCF_RESKEY_cmdline_options=`string] [`OCF_RESKEY_pidfile=`string] [`OCF_RESKEY_logfile=`string] [`OCF_RESKEY_errlogfile=`string] [`OCF_RESKEY_user=`string] [`OCF_RESKEY_monitor_hook=`string] [`OCF_RESKEY_stop_timeout=`string] `anything` [start | stop | monitor | meta-data | validate-all]

Description

This is a generic OCF RA to manage almost anything.

Supported Parameters

`OCF_RESKEY_binfile=`Full path name of the binary to be executed
: The full name of the binary to be executed. This is expected to keep running with the same pid and not just do something and exit.

`OCF_RESKEY_cmdline_options=`Command line options
: Command line options to pass to the binary

`OCF_RESKEY_pidfile=`File to write STDOUT to
: File to read/write the PID from/to.

`OCF_RESKEY_logfile=`File to write STDOUT to
: File to write STDOUT to

`OCF_RESKEY_errlogfile=`File to write STDERR to
: File to write STDERR to

`OCF_RESKEY_user`=User to run the command as
User to run the command as

`OCF_RESKEY_monitor_hook`=Command to run in monitor operation
Command to run in monitor operation

`OCF_RESKEY_stop_timeout`=Seconds to wait after having sent SIGTERM before sending SIGKILL in stop operation
In the stop operation: Seconds to wait for kill -TERM to succeed before sending kill -SIGKILL. Defaults to 2/3 of the stop operation timeout.

ocf:AoEtarget (7)

ocf:AoEtarget — Manages ATA-over-Ethernet (AoE) target exports

Synopsis

[OCF_RESKEY_device=string] [OCF_RESKEY_nic=string] [OCF_RESKEY_shelf=integer] [OCF_RESKEY_slot=integer] OCF_RESKEY_pid=string [OCF_RESKEY_binary=string] AoEtarget [start | stop | monitor | reload | meta-data | validate-all]

Description

This resource agent manages an ATA-over-Ethernet (AoE) target using vblade. It exports any block device, or file, as an AoE target using the specified Ethernet device, shelf, and slot number.

Supported Parameters

OCF_RESKEY_device=Device to export
The local block device (or file) to export as an AoE target.

OCF_RESKEY_nic=Ethernet interface
The local Ethernet interface to use for exporting this AoE target.

OCF_RESKEY_shelf=AoE shelf number
The AoE shelf number to use when exporting this target.

OCF_RESKEY_slot=AoE slot number
The AoE slot number to use when exporting this target.

OCF_RESKEY_pid=Daemon pid file
The file to record the daemon pid to.

OCF_RESKEY_binary=vblade binary
Location of the vblade binary.

ocf:apache (7)

ocf:apache — Manages an Apache web server instance

Synopsis

`OCF_RESKEY_configfile`=string [`OCF_RESKEY_httpd`=string] [`OCF_RESKEY_port`=integer] [`OCF_RESKEY_statusurl`=string] [`OCF_RESKEY_testregex`=string] [`OCF_RESKEY_client`=string] [`OCF_RESKEY_testurl`=string] [`OCF_RESKEY_testregex10`=string] [`OCF_RESKEY_testconffile`=string] [`OCF_RESKEY_testname`=string] [`OCF_RESKEY_options`=string] [`OCF_RESKEY_envfiles`=string] `apache` [start | stop | status | monitor | meta-data | validate-all]

Description

This is the resource agent for the Apache web server. Thie resource agent operates both version 1.x and version 2.x Apache servers. The start operation ends with a loop in which monitor is repeatedly called to make sure that the server started and that it is operational. Hence, if the monitor operation does not succeed within the start operation timeout, the apache resource will end with an error status. The monitor operation by default loads the server status page which depends on the mod_status module and the corresponding configuration file (usually /etc/apache2/mod_status.conf). Make sure that the server status page works and that the access is allowed *only* from localhost (address 127.0.0.1). See the statusurl and testregex attributes for more details. See also http://httpd.apache.org/

Supported Parameters

`OCF_RESKEY_configfile`=configuration file path
: The full pathname of the Apache configuration file. This file is parsed to provide defaults for various other resource agent parameters.

`OCF_RESKEY_httpd`=httpd binary path
: The full pathname of the httpd binary (optional).

`OCF_RESKEY_port`=httpd port
: A port number that we can probe for status information using the statusurl. This will default to the port number found in the configuration file, or 80, if none can be found in the configuration file.

`OCF_RESKEY_statusurl`=url name
: The URL to monitor (the apache server status page by default). If left unspecified, it will be inferred from the apache configuration file. If you set this, make sure that it succeeds *only* from the localhost (127.0.0.1). Otherwise, it may happen that the cluster complains about the resource being active on multiple nodes.

`OCF_RESKEY_testregex`=monitor regular expression
: Regular expression to match in the output of statusurl. Case insensitive.

`OCF_RESKEY_client`=http client
: Client to use to query to Apache. If not specified, the RA will try to find one on the system. Currently, wget and curl are supported. For example, you can set this paramter to "curl" if you prefer that to wget.

`OCF_RESKEY_testurl`=test url
: URL to test. If it does not start with "http", then it's considered to be relative to the Listen address.

`OCF_RESKEY_testregex10`=extended monitor regular expression
: Regular expression to match in the output of testurl. Case insensitive.

`OCF_RESKEY_testconffile`=test configuration file
: A file which contains test configuration. Could be useful if you have to check more than one web application or in case sensitive info should be passed as arguments (passwords). Furthermore, using a config file is the only way to specify certain parameters. Please see README.webapps for examples and file description.

`OCF_RESKEY_testname`=test name
: Name of the test within the test configuration file.

`OCF_RESKEY_options`=command line options
: Extra options to apply when starting apache. See man httpd(8).

`OCF_RESKEY_envfiles=`environment settings files

Files (one or more) which contain extra environment variables. If you want to prevent script from reading the default file, set this parameter to empty string.

ocf:AudibleAlarm (7)

ocf:AudibleAlarm — Emits audible beeps at a configurable interval

Synopsis

[`OCF_RESKEY_nodelist=`string] `AudibleAlarm` [start | stop | restart | status | monitor | meta-data | validate-all]

Description

Resource script for AudibleAlarm. It sets an audible alarm running by beeping at a set interval.

Supported Parameters

`OCF_RESKEY_nodelist=`Node list
    The node list that should never sound the alarm.

ocf:ClusterMon (7)

ocf:ClusterMon — Runs crm_mon in the background, recording the cluster status to an HTML file

Synopsis

[OCF_RESKEY_user=string] [OCF_RESKEY_update=integer] [OCF_RESKEY_extra_options=string] OCF_RESKEY_pidfile=string OCF_RESKEY_htmlfile=string ClusterMon [start | stop | monitor | meta-data | validate-all]

Description

This is a ClusterMon Resource Agent. It outputs current cluster status to the html.

Supported Parameters

OCF_RESKEY_user=The user we want to run crm_mon as
: The user we want to run crm_mon as

OCF_RESKEY_update=Update interval
: How frequently should we update the cluster status

OCF_RESKEY_extra_options=Extra options
: Additional options to pass to crm_mon. Eg. -n -r

OCF_RESKEY_pidfile=PID file
: PID file location to ensure only one instance is running

OCF_RESKEY_htmlfile=HTML output
: Location to write HTML output to.

ocf:CTDB (7)

ocf:CTDB — CTDB Resource Agent

Synopsis

OCF_RESKEY_ctdb_recovery_lock=string
OCF_RESKEY_smb_private_dir=string
[OCF_RESKEY_ctdb_config_dir=string]
[OCF_RESKEY_ctdb_binary=string] [OCF_RESKEY_ctdbd_binary=string]
[OCF_RESKEY_ctdb_socket=string] [OCF_RESKEY_ctdb_dbdir=string]
[OCF_RESKEY_ctdb_logfile=string]
[OCF_RESKEY_ctdb_debuglevel=integer] [OCF_RESKEY_smb_conf=string]
CTDB [start | stop | monitor | meta-data | validate-all]

Description

This resource agent manages CTDB, allowing one to use Clustered Samba in a Linux-HA/Pacemaker cluster. You need a shared filesystem (e.g. OCFS2) on which the CTDB lock will be stored. Configure shares in smb.conf on all nodes, and create /etc/ctdb/nodes containing a list of private IP addresses of each node in the cluster. Configure this RA as a clone, and it will take care of the rest. For more information see http://linux-ha.org/wiki/CTDB_(resource_agent)

Supported Parameters

OCF_RESKEY_ctdb_recovery_lock=CTDB shared lock file
: The location of a shared lock file, common across all nodes. This must be on shared storage, e.g.: /shared-fs/samba/ctdb.lock

OCF_RESKEY_smb_private_dir=Samba private dir (deprecated)
: The directory for smbd to use for storing such files as smbpasswd and secrets.tdb. Old versions of CTBD (prior to 1.0.50) required this to be on shared storage. This parameter should not be set for current versions of CTDB, and only remains in the RA for backwards compatibility.

`OCF_RESKEY_ctdb_config_dir`=CTDB config file directory
: The directory containing various CTDB configuration files. The "nodes" and "notify.sh" scripts are expected to be in this directory, as is the "events.d" subdirectory.

`OCF_RESKEY_ctdb_binary`=CTDB binary path
: Full path to the CTDB binary.

`OCF_RESKEY_ctdbd_binary`=CTDB Daemon binary path
: Full path to the CTDB cluster daemon binary.

`OCF_RESKEY_ctdb_socket`=CTDB socket location
: Full path to the domain socket that ctdbd will create, used for local clients to attach and communicate with the ctdb daemon.

`OCF_RESKEY_ctdb_dbdir`=CTDB database directory
: The directory to put the local CTDB database files in. Persistent database files will be put in ctdb_dbdir/persistent.

`OCF_RESKEY_ctdb_logfile`=CTDB log file location
: Full path to log file. To log to syslog instead, use the value "syslog".

`OCF_RESKEY_ctdb_debuglevel`=CTDB debug level
: What debug level to run at (0-10). Higher means more verbose.

`OCF_RESKEY_smb_conf`=Path to smb.conf
: Path to default samba config file.

ocf:db2 (7)

ocf:db2 — Manages an IBM DB2 Universal Database instance

Synopsis

[`OCF_RESKEY_instance`=string] [`OCF_RESKEY_admin`=string] `db2` [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for db2. It manages a DB2 Universal Database instance as an HA resource.

Supported Parameters

`OCF_RESKEY_instance`=instance
  The instance of database.

`OCF_RESKEY_admin`=admin
  The admin user of the instance.

ocf:Delay (7)

ocf:Delay — Waits for a defined timespan

Synopsis

[OCF_RESKEY_startdelay=integer] [OCF_RESKEY_stopdelay=integer] [OCF_RESKEY_mondelay=integer] Delay [start | stop | status | monitor | meta-data | validate-all]

Description

This script is a test resource for introducing delay.

Supported Parameters

OCF_RESKEY_startdelay=Start delay
: How long in seconds to delay on start operation.

OCF_RESKEY_stopdelay=Stop delay
: How long in seconds to delay on stop operation. Defaults to "startdelay" if unspecified.

OCF_RESKEY_mondelay=Monitor delay
: How long in seconds to delay on monitor operation. Defaults to "startdelay" if unspecified.

ocf:drbd (7)

ocf:drbd — Manages a DRBD resource (deprecated)

Synopsis

`OCF_RESKEY_drbd_resource=`string [`OCF_RESKEY_drbdconf=`string] [`OCF_RESKEY_clone_overrides_hostname=`boolean] [`OCF_RESKEY_ignore_deprecation=`boolean] `drbd` [start | promote | demote | notify | stop | monitor | monitor | meta-data | validate-all]

Description

Deprecation warning: This agent is deprecated and may be removed from a future release. See the ocf:linbit:drbd resource agent for a supported alternative. -- This resource agent manages a Distributed Replicated Block Device (DRBD) object as a master/slave resource. DRBD is a mechanism for replicating storage; please see the documentation for setup details.

Supported Parameters

`OCF_RESKEY_drbd_resource=`drbd resource name
The name of the drbd resource from the drbd.conf file.

`OCF_RESKEY_drbdconf=`Path to drbd.conf
Full path to the drbd.conf file.

`OCF_RESKEY_clone_overrides_hostname=`Override drbd hostname
Whether or not to override the hostname with the clone number. This can be used to create floating peer configurations; drbd will be told to use node_<cloneno> as the hostname instead of the real uname, which can then be used in drbd.conf.

`OCF_RESKEY_ignore_deprecation=`Suppress deprecation warning
If set to true, suppresses the deprecation warning for this agent.

ocf:Dummy (7)

ocf:Dummy — Example stateless resource agent

Synopsis

`OCF_RESKEY_state`=string `Dummy` [start | stop | monitor | reload | migrate_to | migrate_from | meta-data | validate-all]

Description

This is a Dummy Resource Agent. It does absolutely nothing except keep track of whether its running or not. Its purpose in life is for testing and to serve as a template for RA writers. NB: Please pay attention to the timeouts specified in the actions section below. They should be meaningful for the kind of resource the agent manages. They should be the minimum advised timeouts, but they shouldn't/cannot cover _all_ possible resource instances. So, try to be neither overly generous nor too stingy, but moderate. The minimum timeouts should never be below 10 seconds.

Supported Parameters

`OCF_RESKEY_state`=State file
: Location to store the resource state in.

ocf:eDir88 (7)

ocf:eDir88 — Manages a Novell eDirectory directory server

Synopsis

`OCF_RESKEY_eDir_config_file`=string [`OCF_RESKEY_eDir_monitor_ldap`=boolean] [`OCF_RESKEY_eDir_monitor_idm`=boolean] [`OCF_RESKEY_eDir_jvm_initial_heap`=integer] [`OCF_RESKEY_eDir_jvm_max_heap`=integer] [`OCF_RESKEY_eDir_jvm_options`=string] `eDir88` [start | stop | monitor | meta-data | validate-all]

Description

Resource script for managing an eDirectory instance. Manages a single instance of eDirectory as an HA resource. The "multiple instances" feature or eDirectory has been added in version 8.8. This script will not work for any version of eDirectory prior to 8.8. This RA can be used to load multiple eDirectory instances on the same host. It is very strongly recommended to put eDir configuration files (as per the eDir_config_file parameter) on local storage on each node. This is necessary for this RA to be able to handle situations where the shared storage has become unavailable. If the eDir configuration file is not available, this RA will fail, and heartbeat will be unable to manage the resource. Side effects include STONITH actions, unmanageable resources, etc... Setting a high action timeout value is _very_ _strongly_ recommended. eDir with IDM can take in excess of 10 minutes to start. If heartbeat times out before eDir has had a chance to start properly, mayhem _WILL ENSUE_. The LDAP module seems to be one of the very last to start. So this script will take even longer to start on installations with IDM and LDAP if the monitoring of IDM and/or LDAP is enabled, as the start command will wait for IDM and LDAP to be available.

Supported Parameters

`OCF_RESKEY_eDir_config_file`=eDir config file
: Path to configuration file for eDirectory instance.

`OCF_RESKEY_eDir_monitor_ldap`=eDir monitor ldap
: Should we monitor if LDAP is running for the eDirectory instance?

`OCF_RESKEY_eDir_monitor_idm`=eDir monitor IDM
: Should we monitor if IDM is running for the eDirectory instance?

`OCF_RESKEY_eDir_jvm_initial_heap`=DHOST_INITIAL_HEAP value
: Value for the DHOST_INITIAL_HEAP java environment variable. If unset, java defaults will be used.

`OCF_RESKEY_eDir_jvm_max_heap`=DHOST_MAX_HEAP value
: Value for the DHOST_MAX_HEAP java environment variable. If unset, java defaults will be used.

`OCF_RESKEY_eDir_jvm_options`=DHOST_OPTIONS value
: Value for the DHOST_OPTIONS java environment variable. If unset, original values will be used.

ocf:Evmsd (7)

ocf:Evmsd — Controls clustered EVMS volume management (deprecated)

Synopsis

[`OCF_RESKEY_ignore_deprecation`=boolean] `Evmsd` [start | stop | monitor | meta-data]

Description

Deprecation warning: EVMS is no longer actively maintained and should not be used. This agent is deprecated and may be removed from a future release. -- This is a Evmsd Resource Agent.

Supported Parameters

`OCF_RESKEY_ignore_deprecation`=Suppress deprecation warning
: If set to true, suppresses the deprecation warning for this agent.

ocf:EvmsSCC (7)

ocf:EvmsSCC — Manages EVMS Shared Cluster Containers (SCCs) (deprecated)

Synopsis

[`OCF_RESKEY_ignore_deprecation=`boolean] `EvmsSCC` [start | stop | notify | status | monitor | meta-data]

Description

Deprecation warning: EVMS is no longer actively maintained and should not be used. This agent is deprecated and may be removed from a future release. -- Resource script for EVMS shared cluster container. It runs evms_activate on one node in the cluster.

Supported Parameters

`OCF_RESKEY_ignore_deprecation=`Suppress deprecation warning
: If set to true, suppresses the deprecation warning for this agent.

ocf:exportfs (7)

ocf:exportfs — Manages NFS exports

Synopsis

[`OCF_RESKEY_clientspec`=string] [`OCF_RESKEY_options`=string] [`OCF_RESKEY_directory`=string] `OCF_RESKEY_fsid`=integer `exportfs` [start | stop | monitor | meta-data | validate-all]

Description

Exportfs uses the exportfs command to add/remove nfs exports. It does NOT manage the nfs server daemon. It depends on Linux specific NFS implementation details, so is considered not portable to other platforms yet.

Supported Parameters

`OCF_RESKEY_clientspec`= Client ACL.
The client specification allowing remote machines to mount the directory over NFS.

`OCF_RESKEY_options`= Export options.
The options to pass to exportfs for the exported directory.

`OCF_RESKEY_directory`= The directory to export.
The directory which you wish to export using NFS.

`OCF_RESKEY_fsid`= Unique fsid within cluster.
The fsid option to pass to exportfs. This should be a unique positive integer, avoid 0 unless you understand its special status. This value will override any fsid provided via the options parameter.

ocf:Filesystem (7)

ocf:Filesystem — Manages filesystem mounts

Synopsis

[OCF_RESKEY_device=string] [OCF_RESKEY_directory=string] [OCF_RESKEY_fstype=string] [OCF_RESKEY_options=string] [OCF_RESKEY_statusfile_prefix=string] [OCF_RESKEY_run_fsck=string] Filesystem [start | stop | notify | monitor | validate-all | meta-data]

Description

Resource script for Filesystem. It manages a Filesystem on a shared storage medium. The standard monitor operation of depth 0 (also known as probe) checks if the filesystem is mounted. If you want deeper tests, set OCF_CHECK_LEVEL to one of the following values: 10: read first 16 blocks of the device (raw read) This doesn't exercise the filesystem at all, but the device on which the filesystem lives. This is noop for non-block devices such as NFS, SMBFS, or bind mounts. 20: test if a status file can be written and read The status file must be writable by root. This is not always the case with an NFS mount, as NFS exports usually have the "root_squash" option set. In such a setup, you must either use read-only monitoring (depth=10), export with "no_root_squash" on your NFS server, or grant world write permissions on the directory where the status file is to be placed.

Supported Parameters

OCF_RESKEY_device=block device
: The name of block device for the filesystem, or -U, -L options for mount, or NFS mount specification.

OCF_RESKEY_directory=mount point
: The mount point for the filesystem.

`OCF_RESKEY_fstype`=filesystem type
: The optional type of filesystem to be mounted.

`OCF_RESKEY_options`=options
: Any extra options to be given as -o options to mount. For bind mounts, add "bind" here and set fstype to "none". We will do the right thing for options such as "bind,ro".

`OCF_RESKEY_statusfile_prefix`=status file prefix
: The prefix to be used for a status file for resource monitoring with depth 20. If you don't specify this parameter, all status files will be created in a separate directory.

`OCF_RESKEY_run_fsck`=run_fsck
: Specify how to decide whether to run fsck or not. "auto" : decide to run fsck depending on the fstype(default) "force" : always run fsck regardless of the fstype "no" : do not run fsck ever.

ocf:fio (7)

ocf:fio — fio IO load generator

Synopsis

[OCF_RESKEY_args=string] fio [start | stop | monitor | meta-data | validate-all]

Description

fio is a generic I/O load generator. This RA allows start/stop of fio instances to simulate load on a cluster without configuring complex services.

Supported Parameters

OCF_RESKEY_args=fio arguments
: Arguments to the fio client. Minimally, this should be a (list of) job descriptions to run.

ocf:ICP (7)

ocf:ICP — Manages an ICP Vortex clustered host drive

Synopsis

[OCF_RESKEY_driveid=string] [OCF_RESKEY_device=string] ICP [start | stop | status | monitor | validate-all | meta-data]

Description

Resource script for ICP. It Manages an ICP Vortex clustered host drive as an HA resource.

Supported Parameters

OCF_RESKEY_driveid=ICP cluster drive ID
: The ICP cluster drive ID.

OCF_RESKEY_device=device
: The device name.

ocf:ids (7)

ocf:ids — Manages an Informix Dynamic Server (IDS) instance

Synopsis

[OCF_RESKEY_informixdir=string] [OCF_RESKEY_informixserver=string] [OCF_RESKEY_onconfig=string] [OCF_RESKEY_dbname=string] [OCF_RESKEY_sqltestquery=string] ids [start | stop | status | monitor | validate-all | meta-data | methods | usage]

Description

OCF resource agent to manage an IBM Informix Dynamic Server (IDS) instance as an High-Availability resource.

Supported Parameters

OCF_RESKEY_informixdir= INFORMIXDIR environment variable

The value the environment variable INFORMIXDIR has after a typical installation of IDS. Or in other words: the path (without trailing '/') where IDS was installed to. If this parameter is unspecified the script will try to get the value from the shell environment.

OCF_RESKEY_informixserver= INFORMIXSERVER environment variable

The value the environment variable INFORMIXSERVER has after a typical installation of IDS. Or in other words: the name of the IDS server instance to manage. If this parameter is unspecified the script will try to get the value from the shell environment.

OCF_RESKEY_onconfig= ONCONFIG environment variable

The value the environment variable ONCONFIG has after a typical installation of IDS. Or in other words: the name of the configuration file for the IDS instance specified in INFORMIXSERVER. The specified configuration file will be searched at '/etc/'. If this parameter is unspecified the script will try to get the value from the shell environment.

`OCF_RESKEY_dbname=` database to use for monitoring, defaults to 'sysmaster'
This parameter defines which database to use in order to monitor the IDS instance. If this parameter is unspecified the script will use the 'sysmaster' database as a default.

`OCF_RESKEY_sqltestquery=` SQL test query to use for monitoring, defaults to 'SELECT COUNT(*) FROM systables;'
SQL test query to run on the database specified by the parameter 'dbname' in order to monitor the IDS instance and determine if it's functional or not. If this parameter is unspecified the script will use 'SELECT COUNT(*) FROM systables;' as a default.

ocf:IPaddr2 (7)

ocf:IPaddr2 — Manages virtual IPv4 addresses (Linux specific version)

Synopsis

OCF_RESKEY_ip=string [OCF_RESKEY_nic=string] [OCF_RESKEY_cidr_netmask=string] [OCF_RESKEY_broadcast=string] [OCF_RESKEY_iflabel=string] [OCF_RESKEY_lvs_support=boolean] [OCF_RESKEY_mac=string] [OCF_RESKEY_clusterip_hash=string] [OCF_RESKEY_unique_clone_address=boolean] [OCF_RESKEY_arp_interval=integer] [OCF_RESKEY_arp_count=integer] [OCF_RESKEY_arp_bg=string] [OCF_RESKEY_arp_mac=string] IPaddr2 [start | stop | status | monitor | meta-data | validate-all]

Description

This Linux-specific resource manages IP alias IP addresses. It can add an IP alias, or remove one. In addition, it can implement Cluster Alias IP functionality if invoked as a clone resource.

Supported Parameters

OCF_RESKEY_ip=IPv4 address
: The IPv4 address to be configured in dotted quad notation, for example "192.168.1.1".

OCF_RESKEY_nic=Network interface
: The base network interface on which the IP address will be brought online. If left empty, the script will try and determine this from the routing table. Do NOT specify an alias interface in the form eth0:1 or anything here; rather, specify the base interface only. Prerequisite: There must be at least one static IP address, which is not managed by the cluster, assigned to the network interface. If you can not assign any static IP address on the interface, modify this kernel parameter: sysctl -w net.ipv4.conf.all.promote_secondaries=1 (or per device)

`OCF_RESKEY_cidr_netmask`=CIDR netmask
: The netmask for the interface in CIDR format (e.g., 24 and not 255.255.255.0) If unspecified, the script will also try to determine this from the routing table.

`OCF_RESKEY_broadcast`=Broadcast address
: Broadcast address associated with the IP. If left empty, the script will determine this from the netmask.

`OCF_RESKEY_iflabel`=Interface label
: You can specify an additional label for your IP address here. This label is appended to your interface name. If a label is specified in nic name, this parameter has no effect.

`OCF_RESKEY_lvs_support`=Enable support for LVS DR
: Enable support for LVS Direct Routing configurations. In case a IP address is stopped, only move it to the loopback device to allow the local node to continue to service requests, but no longer advertise it on the network.

`OCF_RESKEY_mac`=Cluster IP MAC address
: Set the interface MAC address explicitly. Currently only used in case of the Cluster IP Alias. Leave empty to chose automatically.

`OCF_RESKEY_clusterip_hash`=Cluster IP hashing function
: Specify the hashing algorithm used for the Cluster IP functionality.

`OCF_RESKEY_unique_clone_address`=Create a unique address for cloned instances
: If true, add the clone ID to the supplied value of ip to create a unique address to manage

`OCF_RESKEY_arp_interval`=ARP packet interval in ms
: Specify the interval between unsolicited ARP packets in milliseconds.

`OCF_RESKEY_arp_count`=ARP packet count
: Number of unsolicited ARP packets to send.

`OCF_RESKEY_arp_bg`=ARP from background
: Whether or not to send the arp packets in the background.

`OCF_RESKEY_arp_mac`=ARP MAC
MAC address to send the ARP packets too. You really shouldn't be touching this.

ocf:IPaddr (7)

ocf:IPaddr — Manages virtual IPv4 addresses (portable version)

Synopsis

`OCF_RESKEY_ip`=string [`OCF_RESKEY_nic`=string] [`OCF_RESKEY_cidr_netmask`=string] [`OCF_RESKEY_broadcast`=string] [`OCF_RESKEY_iflabel`=string] [`OCF_RESKEY_lvs_support`=boolean] [`OCF_RESKEY_local_stop_script`=string] [`OCF_RESKEY_local_start_script`=string] [`OCF_RESKEY_ARP_INTERVAL_MS`=integer] [`OCF_RESKEY_ARP_REPEAT`=integer] [`OCF_RESKEY_ARP_BACKGROUND`=boolean] [`OCF_RESKEY_ARP_NETMASK`=string] `IPaddr` [start | stop | monitor | validate-all | meta-data]

Description

This script manages IP alias IP addresses It can add an IP alias, or remove one.

Supported Parameters

`OCF_RESKEY_ip`=IPv4 address

The IPv4 address to be configured in dotted quad notation, for example "192.168.1.1".

`OCF_RESKEY_nic`=Network interface

The base network interface on which the IP address will be brought online. If left empty, the script will try and determine this from the routing table. Do NOT specify an alias interface in the form eth0:1 or anything here; rather, specify the base interface only. Prerequisite: There must be at least one static IP address, which is not managed by the cluster, assigned to the network interface. If you can not assign any static IP address on the interface, modify this kernel parameter: sysctl -w net.ipv4.conf.all.promote_secondaries=1 (or per device)

`OCF_RESKEY_cidr_netmask=`Netmask
: The netmask for the interface in CIDR format. (ie, 24), or in dotted quad notation 255.255.255.0). If unspecified, the script will also try to determine this from the routing table.

`OCF_RESKEY_broadcast=`Broadcast address
: Broadcast address associated with the IP. If left empty, the script will determine this from the netmask.

`OCF_RESKEY_iflabel=`Interface label
: You can specify an additional label for your IP address here.

`OCF_RESKEY_lvs_support=`Enable support for LVS DR
: Enable support for LVS Direct Routing configurations. In case a IP address is stopped, only move it to the loopback device to allow the local node to continue to service requests, but no longer advertise it on the network.

`OCF_RESKEY_local_stop_script=`Script called when the IP is released
: Script called when the IP is released

`OCF_RESKEY_local_start_script=`Script called when the IP is added
: Script called when the IP is added

`OCF_RESKEY_ARP_INTERVAL_MS=`milliseconds between gratuitous ARPs
: milliseconds between ARPs

`OCF_RESKEY_ARP_REPEAT=`repeat count
: How many gratuitous ARPs to send out when bringing up a new address

`OCF_RESKEY_ARP_BACKGROUND=`run in background
: run in background (no longer any reason to do this)

`OCF_RESKEY_ARP_NETMASK=`netmask for ARP
: netmask for ARP - in nonstandard hexadecimal format.

ocf:IPsrcaddr (7)

ocf:IPsrcaddr — Manages the preferred source address for outgoing IP packets

Synopsis

[`OCF_RESKEY_ipaddress`=string] [`OCF_RESKEY_cidr_netmask`=string]
`IPsrcaddr` [start | stop | monitor | validate-all | meta-data]

Description

Resource script for IPsrcaddr. It manages the preferred source address modification.

Supported Parameters

`OCF_RESKEY_ipaddress`=IP address
: The IP address.

`OCF_RESKEY_cidr_netmask`=Netmask
: The netmask for the interface in CIDR format. (ie, 24), or in dotted quad notation 255.255.255.0).

ocf:IPv6addr (7)

ocf:IPv6addr — Manages IPv6 aliases

Synopsis

[OCF_RESKEY_ipv6addr=string] [OCF_RESKEY_cidr_netmask=string] [OCF_RESKEY_nic=string] IPv6addr [start | stop | status | monitor | validate-all | meta-data]

Description

This script manages IPv6 alias IPv6 addresses,It can add an IP6 alias, or remove one.

Supported Parameters

OCF_RESKEY_ipv6addr=IPv6 address
The IPv6 address this RA will manage

OCF_RESKEY_cidr_netmask=Netmask
The netmask for the interface in CIDR format. (ie, 24). The value of this parameter overwrites the value of _prefix_ of ipv6addr parameter.

OCF_RESKEY_nic=Network interface
The base network interface on which the IPv6 address will be brought online.

ocf:iSCSILogicalUnit (7)

ocf:iSCSILogicalUnit — Manages iSCSI Logical Units (LUs)

Synopsis

[OCF_RESKEY_implementation=string] [OCF_RESKEY_target_iqn=string] [OCF_RESKEY_lun=integer] [OCF_RESKEY_path=string] OCF_RESKEY_scsi_id=string OCF_RESKEY_scsi_sn=string [OCF_RESKEY_vendor_id=string] [OCF_RESKEY_product_id=string] [OCF_RESKEY_additional_parameters=string] iSCSILogicalUnit [start | stop | status | monitor | meta-data | validate-all]

Description

Manages iSCSI Logical Unit. An iSCSI Logical unit is a subdivision of an SCSI Target, exported via a daemon that speaks the iSCSI protocol.

Supported Parameters

OCF_RESKEY_implementation=iSCSI target daemon implementation
: The iSCSI target daemon implementation. Must be one of "iet", "tgt", or "lio". If unspecified, an implementation is selected based on the availability of management utilities, with "iet" being tried first, then "tgt", then "lio".

OCF_RESKEY_target_iqn=iSCSI target IQN
: The iSCSI Qualified Name (IQN) that this Logical Unit belongs to.

OCF_RESKEY_lun=Logical Unit number (LUN)
: The Logical Unit number (LUN) exposed to initiators.

OCF_RESKEY_path=Block device (or file) path
: The path to the block device exposed. Some implementations allow this to be a regular file, too.

`OCF_RESKEY_scsi_id`=SCSI ID
: The SCSI ID to be configured for this Logical Unit. The default is the resource name, truncated to 24 bytes.

`OCF_RESKEY_scsi_sn`=SCSI serial number
: The SCSI serial number to be configured for this Logical Unit. The default is a hash of the resource name, truncated to 8 bytes.

`OCF_RESKEY_vendor_id`=SCSI vendor ID
: The SCSI vendor ID to be configured for this Logical Unit.

`OCF_RESKEY_product_id`=SCSI product ID
: The SCSI product ID to be configured for this Logical Unit.

`OCF_RESKEY_additional_parameters`=List of iSCSI LU parameters
: Additional LU parameters. A space-separated list of "name=value" pairs which will be passed through to the iSCSI daemon's management interface. The supported parameters are implementation dependent. Neither the name nor the value may contain whitespace.

ocf:iSCSITarget (7)

ocf:iSCSITarget — iSCSI target export agent

Synopsis

[OCF_RESKEY_implementation=string] OCF_RESKEY_iqn=string OCF_RESKEY_tid=integer [OCF_RESKEY_portals=string] [OCF_RESKEY_allowed_initiators=string] OCF_RESKEY_incoming_username=string [OCF_RESKEY_incoming_password=string] [OCF_RESKEY_additional_parameters=string] iSCSITarget [start | stop | status | monitor | meta-data | validate-all]

Description

Manages iSCSI targets. An iSCSI target is a collection of SCSI Logical Units (LUs) exported via a daemon that speaks the iSCSI protocol.

Supported Parameters

OCF_RESKEY_implementation=Manages an iSCSI target export
: The iSCSI target daemon implementation. Must be one of "iet", "tgt", or "lio". If unspecified, an implementation is selected based on the availability of management utilities, with "iet" being tried first, then "tgt", then "lio".

OCF_RESKEY_iqn=iSCSI target IQN
: The target iSCSI Qualified Name (IQN). Should follow the conventional "iqn.yyyy-mm.<reversed domain name>[:identifier]" syntax.

OCF_RESKEY_tid=iSCSI target ID
: The iSCSI target ID. Required for tgt.

`OCF_RESKEY_portals`=iSCSI portal addresses

iSCSI network portal addresses. Not supported by all implementations. If unset, the default is to create one portal that listens on .

`OCF_RESKEY_allowed_initiators`=List of iSCSI initiators allowed to connect to this target

Allowed initiators. A space-separated list of initiators allowed to connect to this target. Initiators may be listed in any syntax the target implementation allows. If this parameter is empty or not set, access to this target will be allowed from any initiator.

`OCF_RESKEY_incoming_username`=Incoming account username

A username used for incoming initiator authentication. If unspecified, allowed initiators will be able to log in without authentication.

`OCF_RESKEY_incoming_password`=Incoming account password

A password used for incoming initiator authentication.

`OCF_RESKEY_additional_parameters`=List of iSCSI target parameters

Additional target parameters. A space-separated list of "name=value" pairs which will be passed through to the iSCSI daemon's management interface. The supported parameters are implementation dependent. Neither the name nor the value may contain whitespace.

ocf:iscsi (7)

ocf:iscsi — Manages a local iSCSI initiator and its connections to iSCSI targets

Synopsis

[`OCF_RESKEY_portal=`string] `OCF_RESKEY_target=`string [`OCF_RESKEY_discovery_type=`string] [`OCF_RESKEY_iscsiadm=`string] [`OCF_RESKEY_udev=`string] `iscsi` [start | stop | status | monitor | validate-all | methods | meta-data]

Description

OCF Resource Agent for iSCSI. Add (start) or remove (stop) iSCSI targets.

Supported Parameters

`OCF_RESKEY_portal=`Portal address
: The iSCSI portal address in the form: {ip_address|hostname}[":"port]

`OCF_RESKEY_target=`Target IQN
: The iSCSI target IQN.

`OCF_RESKEY_discovery_type=`Target discovery type
: Target discovery type. Check the open-iscsi documentation for supported discovery types.

`OCF_RESKEY_iscsiadm=`iscsiadm binary
: open-iscsi administration utility binary.

`OCF_RESKEY_udev=`udev
: If the next resource depends on the udev creating a device then we wait until it is finished. On a normally loaded host this should be done quickly, but you may be unlucky. If you are not using udev set this to "no", otherwise we will spin in a loop until a timeout occurs.

ocf:ldirectord (7)

ocf:ldirectord — Wrapper OCF Resource Agent for ldirectord

Synopsis

`OCF_RESKEY_configfile=`string [`OCF_RESKEY_ldirectord=`string]
`ldirectord` [start | stop | monitor | meta-data | validate-all]

Description

It's a simple OCF RA wrapper for ldirectord and uses the ldirectord interface to create the OCF compliant interface. You win monitoring of ldirectord. Be warned: Asking ldirectord status is an expensive action.

Supported Parameters

`OCF_RESKEY_configfile=`configuration file path
  The full pathname of the ldirectord configuration file.

`OCF_RESKEY_ldirectord=`ldirectord binary path
  The full pathname of the ldirectord.

ocf:LinuxSCSI (7)

ocf:LinuxSCSI — Enables and disables SCSI devices through the kernel SCSI hot-plug subsystem (deprecated)

Synopsis

[`OCF_RESKEY_scsi=`string] [`OCF_RESKEY_ignore_deprecation=`boolean] `LinuxSCSI` [start | stop | methods | status | monitor | meta-data | validate-all]

Description

Deprecation warning: This agent makes use of Linux SCSI hot-plug functionality which has been superseded by SCSI reservations. It is deprecated and may be removed from a future release. See the scsi2reservation and sfex agents for alternatives. -- This is a resource agent for LinuxSCSI. It manages the availability of a SCSI device from the point of view of the linux kernel. It make Linux believe the device has gone away, and it can make it come back again.

Supported Parameters

`OCF_RESKEY_scsi=`SCSI instance
: The SCSI instance to be managed.

`OCF_RESKEY_ignore_deprecation=`Suppress deprecation warning
: If set to true, suppresses the deprecation warning for this agent.

ocf:LVM (7)

ocf:LVM — Controls the availability of an LVM Volume Group

Synopsis

[OCF_RESKEY_volgrpname=string] [OCF_RESKEY_exclusive=string] LVM [start | stop | status | monitor | methods | meta-data | validate-all]

Description

Resource script for LVM. It manages an Linux Volume Manager volume (LVM) as an HA resource.

Supported Parameters

OCF_RESKEY_volgrpname=Volume group name
The name of volume group.

OCF_RESKEY_exclusive=Exclusive activation
If set, the volume group will be activated exclusively.

ocf:MailTo (7)

ocf:MailTo — Notifies recipients by email in the event of resource takeover

Synopsis

[`OCF_RESKEY_email`=string] [`OCF_RESKEY_subject`=string] `MailTo` [start | stop | status | monitor | meta-data | validate-all]

Description

This is a resource agent for MailTo. It sends email to a sysadmin whenever a takeover occurs.

Supported Parameters

`OCF_RESKEY_email`=Email address
  The email address of sysadmin.

`OCF_RESKEY_subject`=Subject
  The subject of the email.

ocf:ManageRAID (7)

ocf:ManageRAID — Manages RAID devices

Synopsis

[`OCF_RESKEY_raidname`=string] `ManageRAID` [start | stop | status | monitor | validate-all | meta-data]

Description

Manages starting, stopping and monitoring of RAID devices which are preconfigured in /etc/conf.d/HB-ManageRAID.

Supported Parameters

`OCF_RESKEY_raidname`=RAID name
: Name (case sensitive) of RAID to manage. (preconfigured in /etc/conf.d/HB-ManageRAID)

ocf:ManageVE (7)

ocf:ManageVE — Manages an OpenVZ Virtual Environment (VE)

Synopsis

[`OCF_RESKEY_veid`=integer] `ManageVE` [start | stop | status | monitor | validate-all | meta-data]

Description

This OCF complaint resource agent manages OpenVZ VEs and thus requires a proper OpenVZ installation including a recent vzctl util.

Supported Parameters

`OCF_RESKEY_veid`=OpenVZ ID of VE
: OpenVZ ID of virtual environment (see output of vzlist -a for all assigned IDs)

ocf:mysql-proxy (7)

ocf:mysql-proxy — Manages a MySQL Proxy daemon

Synopsis

[OCF_RESKEY_binary=string] OCF_RESKEY_defaults_file=string
[OCF_RESKEY_proxy_backend_addresses=string]
[OCF_RESKEY_proxy_read_only_backend_addresses=string]
[OCF_RESKEY_proxy_address=string] [OCF_RESKEY_log_level=string]
[OCF_RESKEY_keepalive=string] [OCF_RESKEY_admin_address=string]
[OCF_RESKEY_admin_username=string]
[OCF_RESKEY_admin_password=string]
[OCF_RESKEY_admin_lua_script=string]
[OCF_RESKEY_parameters=string] OCF_RESKEY_pidfile=string
mysql-proxy [start | stop | reload | monitor | validate-all | meta-data]

Description

This script manages MySQL Proxy as an OCF resource in a high-availability setup. Tested with MySQL Proxy 0.7.0 on Debian 5.0.

Supported Parameters

OCF_RESKEY_binary=Full path to MySQL Proxy binary
Full path to the MySQL Proxy binary. For example, "/usr/sbin/mysql-proxy".

OCF_RESKEY_defaults_file=Full path to configuration file
Full path to a MySQL Proxy configuration file. For example, "/etc/mysql-proxy.conf".

OCF_RESKEY_proxy_backend_addresses=MySQL Proxy backend-servers
Address:port of the remote backend-servers (default: 127.0.0.1:3306).

`OCF_RESKEY_proxy_read_only_backend_addresses`=MySql Proxy read only backend-servers

Address:port of the remote (read only) slave-server (default:).

`OCF_RESKEY_proxy_address`=MySQL Proxy listening address

Listening address:port of the proxy-server (default: :4040). You can also specify a socket like "/tmp/mysql-proxy.sock".

`OCF_RESKEY_log_level`=MySQL Proxy log level.

Log all messages of level (error|warning|info|message|debug|) or higher. An empty value disables logging.

`OCF_RESKEY_keepalive`=Use keepalive option

Try to restart the proxy if it crashed (default:). Valid values: true or false. An empty value equals "false".

`OCF_RESKEY_admin_address`=MySQL Proxy admin-server address

Listening address:port of the admin-server (default: 127.0.0.1:4041).

`OCF_RESKEY_admin_username`=MySQL Proxy admin-server username

Username to allow to log in (default:).

`OCF_RESKEY_admin_password`=MySQL Proxy admin-server password

Password to allow to log in (default:).

`OCF_RESKEY_admin_lua_script`=MySQL Proxy admin-server lua script

Script to execute by the admin plugin.

`OCF_RESKEY_parameters`=MySQL Proxy additional parameters

The MySQL Proxy daemon may be called with additional parameters. Specify any of them here.

`OCF_RESKEY_pidfile`=PID file

PID file

ocf:mysql (7)

ocf:mysql — Manages a MySQL database instance

Synopsis

[OCF_RESKEY_binary=string] [OCF_RESKEY_client_binary=string] [OCF_RESKEY_config=string] [OCF_RESKEY_datadir=string] [OCF_RESKEY_user=string] [OCF_RESKEY_group=string] [OCF_RESKEY_log=string] [OCF_RESKEY_pid=string] [OCF_RESKEY_socket=string] [OCF_RESKEY_test_table=string] [OCF_RESKEY_test_user=string] [OCF_RESKEY_test_passwd=string] [OCF_RESKEY_enable_creation=integer] [OCF_RESKEY_additional_parameters=string] [OCF_RESKEY_replication_user=string] [OCF_RESKEY_replication_passwd=string] [OCF_RESKEY_replication_port=string] [OCF_RESKEY_max_slave_lag=integer] [OCF_RESKEY_evict_outdated_slaves=boolean] mysql [start | stop | status | monitor | monitor | monitor | promote | demote | notify | validate-all | meta-data]

Description

Resource script for MySQL. May manage a standalone MySQL database, a clone set with externally managed replication, or a complete master/slave replication setup.

Supported Parameters

OCF_RESKEY_binary=MySQL server binary
Location of the MySQL server binary

OCF_RESKEY_client_binary=MySQL client binary
Location of the MySQL client binary

`OCF_RESKEY_config`=MySQL config
    Configuration file

`OCF_RESKEY_datadir`=MySQL datadir
    Directory containing databases

`OCF_RESKEY_user`=MySQL user
    User running MySQL daemon

`OCF_RESKEY_group`=MySQL group
    Group running MySQL daemon (for logfile and directory permissions)

`OCF_RESKEY_log`=MySQL log file
    The logfile to be used for mysqld.

`OCF_RESKEY_pid`=MySQL pid file
    The pidfile to be used for mysqld.

`OCF_RESKEY_socket`=MySQL socket
    The socket to be used for mysqld.

`OCF_RESKEY_test_table`=MySQL test table
    Table to be tested in monitor statement (in database.table notation)

`OCF_RESKEY_test_user`=MySQL test user
    MySQL test user

`OCF_RESKEY_test_passwd`=MySQL test user password
    MySQL test user password

`OCF_RESKEY_enable_creation`=Create the database if it does not exist
    If the MySQL database does not exist, it will be created

`OCF_RESKEY_additional_parameters`=Additional parameters to pass to mysqld
    Additional parameters which are passed to the mysqld on startup. (e.g. --skip-external-locking or --skip-grant-tables) On M/S setup --skip-slave-start is needed (or in config file).

`OCF_RESKEY_replication_user`=MySQL replication user

MySQL replication user. This user is used for starting and stopping MySQL replication, for setting and resetting the master host, and for setting and unsetting read-only mode. Because of that, this user must have SUPER, REPLICATION SLAVE, REPLICATION CLIENT, and PROCESS privileges on all nodes within the cluster.

`OCF_RESKEY_replication_passwd`=MySQL replication user password

MySQL replication password. Used for replication client and slave.

`OCF_RESKEY_replication_port`=MySQL replication port

The port on which the Master MySQL instance is listening.

`OCF_RESKEY_max_slave_lag`=Maximum time (seconds) a MySQL slave is allowed to lag behind a master

The maximum number of seconds a replication slave is allowed to lag behind its master. Do not set this to zero. What the cluster manager does in case a slave exceeds this maximum lag is determined by the evict_outdated_slaves parameter.

`OCF_RESKEY_evict_outdated_slaves`=Determines whether to shut down badly lagging slaves

If set to true, any slave which is more than max_slave_lag seconds behind the master has its MySQL instance shut down. If this parameter is set to false in a primitive or clone resource, it is simply ignored. If set to false in a master/slave resource, then exceeding the maximum slave lag will merely push down the master preference so the lagging slave is never promoted to the new master.

ocf:nfsserver (7)

ocf:nfsserver — Manages an NFS server

Synopsis

[OCF_RESKEY_nfs_init_script=string]
[OCF_RESKEY_nfs_notify_cmd=string]
[OCF_RESKEY_nfs_shared_infodir=string] [OCF_RESKEY_nfs_ip=string]
nfsserver [start | stop | monitor | meta-data | validate-all]

Description

Nfsserver helps to manage the Linux nfs server as a failover-able resource in Linux-HA. It depends on Linux specific NFS implementation details, so is considered not portable to other platforms yet.

Supported Parameters

OCF_RESKEY_nfs_init_script= Init script for nfsserver
The default init script shipped with the Linux distro. The nfsserver resource agent offloads the start/stop/monitor work to the init script because the procedure to start/stop/monitor nfsserver varies on different Linux distro.

OCF_RESKEY_nfs_notify_cmd= The tool to send out notification.
The tool to send out NSM reboot notification. Failover of nfsserver can be considered as rebooting to different machines. The nfsserver resource agent use this command to notify all clients about the happening of failover.

OCF_RESKEY_nfs_shared_infodir= Directory to store nfs server related information.
The nfsserver resource agent will save nfs related information in this specific directory. And this directory must be able to fail-over before nfsserver itself.

`OCF_RESKEY_nfs_ip=` IP address.
The floating IP address used to access the nfs service

ocf:oracle (7)

ocf:oracle — Manages an Oracle Database instance

Synopsis

OCF_RESKEY_sid=string [OCF_RESKEY_home=string] [OCF_RESKEY_user=string] [OCF_RESKEY_ipcrm=string] [OCF_RESKEY_clear_backupmode=boolean] [OCF_RESKEY_shutdown_method=string] oracle [start | stop | status | monitor | validate-all | methods | meta-data]

Description

Resource script for oracle. Manages an Oracle Database instance as an HA resource.

Supported Parameters

OCF_RESKEY_sid=sid
The Oracle SID (aka ORACLE_SID).

OCF_RESKEY_home=home
The Oracle home directory (aka ORACLE_HOME). If not specified, then the SID along with its home should be listed in /etc/oratab.

OCF_RESKEY_user=user
The Oracle owner (aka ORACLE_OWNER). If not specified, then it is set to the owner of file $ORACLE_HOME/dbs/*${ORACLE_SID}.ora. If this does not work for you, just set it explicitely.

OCF_RESKEY_ipcrm=ipcrm
Sometimes IPC objects (shared memory segments and semaphores) belonging to an Oracle instance might be left behind which prevents the instance from starting. It is not easy to figure out which shared segments belong to which instance, in particular when more instances are running as same user. What we use here is the

"oradebug" feature and its "ipc" trace utility. It is not optimal to parse the debugging information, but I am not aware of any other way to find out about the IPC information. In case the format or wording of the trace report changes, parsing might fail. There are some precautions, however, to prevent stepping on other peoples toes. There is also a dumpinstipc option which will make us print the IPC objects which belong to the instance. Use it to see if we parse the trace file correctly. Three settings are possible: - none: don't mess with IPC and hope for the best (beware: you'll probably be out of luck, sooner or later) - instance: try to figure out the IPC stuff which belongs to the instance and remove only those (default; should be safe) - orauser: remove all IPC belonging to the user which runs the instance (don't use this if you run more than one instance as same user or if other apps running as this user use IPC) The default setting "instance" should be safe to use, but in that case we cannot guarantee that the instance will start. In case IPC objects were already left around, because, for instance, someone mercilessly killing Oracle processes, there is no way any more to find out which IPC objects should be removed. In that case, human intervention is necessary, and probably _all_ instances running as same user will have to be stopped. The third setting, "orauser", guarantees IPC objects removal, but it does that based only on IPC objects ownership, so you should use that only if every instance runs as separate user. Please report any problems. Suggestions/fixes welcome.

`OCF_RESKEY_clear_backupmode`=clear_backupmode
The clear of the backup mode of ORACLE.

`OCF_RESKEY_shutdown_method`=shutdown_method
How to stop Oracle is a matter of taste it seems. The default method ("checkpoint/abort") is: alter system checkpoint; shutdown abort; This should be the fastest safe way bring the instance down. If you find "shutdown abort" distasteful, set this attribute to "immediate" in which case we will shutdown immediate; If you still think that there's even better way to shutdown an Oracle instance we are willing to listen.

ocf:oralsnr (7)

ocf:oralsnr — Manages an Oracle TNS listener

Synopsis

`OCF_RESKEY_sid`=string [`OCF_RESKEY_home`=string] [`OCF_RESKEY_user`=string] `OCF_RESKEY_listener`=string `oralsnr` [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for Oracle Listener. It manages an Oracle Listener instance as an HA resource.

Supported Parameters

`OCF_RESKEY_sid`=sid
: The Oracle SID (aka ORACLE_SID). Necessary for the monitor op, i.e. to do tnsping SID.

`OCF_RESKEY_home`=home
: The Oracle home directory (aka ORACLE_HOME). If not specified, then the SID should be listed in /etc/oratab.

`OCF_RESKEY_user`=user
: Run the listener as this user.

`OCF_RESKEY_listener`=listener
: Listener instance to be started (as defined in listener.ora). Defaults to LISTENER.

ocf:pgsql (7)

ocf:pgsql — Manages a PostgreSQL database instance

Synopsis

[OCF_RESKEY_pgctl=string] [OCF_RESKEY_start_opt=string] [OCF_RESKEY_ctl_opt=string] [OCF_RESKEY_psql=string] [OCF_RESKEY_pgdata=string] [OCF_RESKEY_pgdba=string] [OCF_RESKEY_pghost=string] [OCF_RESKEY_pgport=integer] [OCF_RESKEY_config=integer] [OCF_RESKEY_pgdb=string] [OCF_RESKEY_logfile=string] [OCF_RESKEY_socketdir=string] [OCF_RESKEY_stop_escalate=integer] pgsql [start | stop | status | monitor | meta-data | validate-all | methods]

Description

Resource script for PostgreSQL. It manages a PostgreSQL as an HA resource.

Supported Parameters

OCF_RESKEY_pgctl=pgctl
: Path to pg_ctl command.

OCF_RESKEY_start_opt=start_opt
: Start options (-o start_opt in pg_ctl). "-i -p 5432" for example.

OCF_RESKEY_ctl_opt=ctl_opt
: Additional pg_ctl options (-w, -W etc..).

OCF_RESKEY_psql=psql
: Path to psql command.

OCF_RESKEY_pgdata=pgdata
: Path to PostgreSQL data directory.

`OCF_RESKEY_pgdba`=pgdba
: User that owns PostgreSQL.

`OCF_RESKEY_pghost`=pghost
: Hostname/IP address where PostgreSQL is listening

`OCF_RESKEY_pgport`=pgport
: Port where PostgreSQL is listening

`OCF_RESKEY_config`=Configuration file
: Path to the PostgreSQL configuration file for the instance

`OCF_RESKEY_pgdb`=pgdb
: Database that will be used for monitoring.

`OCF_RESKEY_logfile`=logfile
: Path to PostgreSQL server log output file.

`OCF_RESKEY_socketdir`=socketdir
: Unix socket directory for PostgeSQL

`OCF_RESKEY_stop_escalate`=stop escalation
: Number of shutdown retries (using -m fast) before resorting to -m immediate

ocf:pingd (7)

ocf:pingd — Monitors connectivity to specific hosts or IP addresses ("ping nodes") (deprecated)

Synopsis

[OCF_RESKEY_pidfile=string] [OCF_RESKEY_user=string] [OCF_RESKEY_dampen=integer] [OCF_RESKEY_set=integer] [OCF_RESKEY_name=integer] [OCF_RESKEY_section=integer] [OCF_RESKEY_multiplier=integer] [OCF_RESKEY_host_list=integer] [OCF_RESKEY_ignore_deprecation=boolean] pingd [start | stop | monitor | meta-data | validate-all]

Description

Deprecation warning: This agent is deprecated and may be removed from a future release. See the ocf:pacemaker:pingd resource agent for a supported alternative. -- This is a pingd Resource Agent. It records (in the CIB) the current number of ping nodes a node can connect to.

Supported Parameters

OCF_RESKEY_pidfile=PID file
PID file

OCF_RESKEY_user=The user we want to run pingd as
The user we want to run pingd as

OCF_RESKEY_dampen=Dampening interval
The time to wait (dampening) further changes occur

OCF_RESKEY_set=Set name
The name of the instance_attributes set to place the value in. Rarely needs to be specified.

`OCF_RESKEY_name`=Attribute name
: The name of the attributes to set. This is the name to be used in the constraints.

`OCF_RESKEY_section`=Section name
: The section place the value in. Rarely needs to be specified.

`OCF_RESKEY_multiplier`=Value multiplier
: The number by which to multiply the number of connected ping nodes by

`OCF_RESKEY_host_list`=Host list
: The list of ping nodes to count. Defaults to all configured ping nodes. Rarely needs to be specified.

`OCF_RESKEY_ignore_deprecation`=Suppress deprecation warning
: If set to true, suppresses the deprecation warning for this agent.

ocf:portblock (7)

ocf:portblock — Block and unblocks access to TCP and UDP ports

Synopsis

[OCF_RESKEY_protocol=string] [OCF_RESKEY_portno=integer]
[OCF_RESKEY_action=string] [OCF_RESKEY_ip=string]
[OCF_RESKEY_tickle_dir=string] [OCF_RESKEY_sync_script=string]
portblock [start | stop | status | monitor | meta-data | validate-all]

Description

Resource script for portblock. It is used to temporarily block ports using iptables. In addition, it may allow for faster TCP reconnects for clients on failover. Use that if there are long lived TCP connections to an HA service. This feature is enabled by setting the tickle_dir parameter and only in concert with action set to unblock. Note that the tickle ACK function is new as of version 3.0.2 and hasn't yet seen widespread use.

Supported Parameters

OCF_RESKEY_protocol=protocol
The protocol used to be blocked/unblocked.

OCF_RESKEY_portno=portno
The port number used to be blocked/unblocked.

OCF_RESKEY_action=action
The action (block/unblock) to be done on the protocol::portno.

OCF_RESKEY_ip=ip
The IP address used to be blocked/unblocked.

`OCF_RESKEY_tickle_dir`=Tickle directory
The shared or local directory (_must_ be absolute path) which stores the established TCP connections.

`OCF_RESKEY_sync_script`=Connection state file synchronization script
If the tickle_dir is a local directory, then the TCP connection state file has to be replicated to other nodes in the cluster. It can be csync2 (default), some wrapper of rsync, or whatever. It takes the file name as a single argument. For csync2, set it to "csync2 -xv".

ocf:postfix (7)

ocf:postfix — Manages a highly available Postfix mail server instance

Synopsis

[`OCF_RESKEY_binary`=string] `OCF_RESKEY_config_dir`=string [`OCF_RESKEY_parameters`=string] `postfix` [start | stop | reload | monitor | validate-all | meta-data]

Description

This script manages Postfix as an OCF resource in a high-availability setup. Tested with Postfix 2.5.5 on Debian 5.0.

Supported Parameters

`OCF_RESKEY_binary`=Full path to Postfix binary
: Full path to the Postfix binary. For example, "/usr/sbin/postfix".

`OCF_RESKEY_config_dir`=Full path to configuration directory
: Full path to a Postfix configuration directory. For example, "/etc/postfix".

`OCF_RESKEY_parameters`=
: The Postfix daemon may be called with additional parameters. Specify any of them here.

ocf:proftpd (7)

ocf:proftpd — OCF Resource Agent compliant FTP script.

Synopsis

[OCF_RESKEY_binary=string] [OCF_RESKEY_conffile=string] [OCF_RESKEY_pidfile=string] [OCF_RESKEY_curl_binary=string] [OCF_RESKEY_curl_url=string] [OCF_RESKEY_test_user=string] [OCF_RESKEY_test_pass=string] proftpd [start | stop | monitor | monitor | validate-all | meta-data]

Description

This script manages Proftpd in an Active-Passive setup

Supported Parameters

OCF_RESKEY_binary=The Proftpd binary
The Proftpd binary

OCF_RESKEY_conffile=Configuration file name with full path
The Proftpd configuration file name with full path. For example, "/etc/proftpd.conf"

OCF_RESKEY_pidfile=PID file
The Proftpd PID file. The location of the PID file is configured in the Proftpd configuration file.

OCF_RESKEY_curl_binary=The absolut path to the curl binary
The absolut path to the curl binary for monitoring with OCF_CHECK_LEVEL greater zero.

OCF_RESKEY_curl_url=The URL which is checked by curl
The URL which is checked by curl with OCF_CHECK_LEVEL greater zero.

OCF_RESKEY_test_user=The name of the ftp user
The name of the ftp user for monitoring with OCF_CHECK_LEVEL greater zero.

OCF_RESKEY_test_pass=The password of the ftp user
The password of the ftp user for monitoring with OCF_CHECK_LEVEL greater zero.

ocf:Pure-FTPd (7)

ocf:Pure-FTPd — Manages a Pure-FTPd FTP server instance

Synopsis

`OCF_RESKEY_script`=string `OCF_RESKEY_conffile`=string `OCF_RESKEY_daemon_type`=string [`OCF_RESKEY_pidfile`=string] `Pure-FTPd` [start | stop | monitor | validate-all | meta-data]

Description

This script manages Pure-FTPd in an Active-Passive setup

Supported Parameters

`OCF_RESKEY_script`=Script name with full path
The full path to the Pure-FTPd startup script. For example, "/sbin/pure-config.pl"

`OCF_RESKEY_conffile`=Configuration file name with full path
The Pure-FTPd configuration file name with full path. For example, "/etc/pure-ftpd/pure-ftpd.conf"

`OCF_RESKEY_daemon_type`=Configuration file name with full path
The Pure-FTPd daemon to be called by pure-ftpd-wrapper. Valid options are "" for pure-ftpd, "mysql" for pure-ftpd-mysql, "postgresql" for pure-ftpd-postgresql and "ldap" for pure-ftpd-ldap

`OCF_RESKEY_pidfile`=PID file
PID file

ocf:Raid1 (7)

ocf:Raid1 — Manages a software RAID1 device on shared storage

Synopsis

[`OCF_RESKEY_raidconf=`string] [`OCF_RESKEY_raiddev=`string] [`OCF_RESKEY_homehost=`string] `Raid1` [start | stop | status | monitor | validate-all | meta-data]

Description

Resource script for RAID1. It manages a software Raid1 device on a shared storage medium.

Supported Parameters

`OCF_RESKEY_raidconf=`RAID config file
: The RAID configuration file. e.g. /etc/raidtab or /etc/mdadm.conf.

`OCF_RESKEY_raiddev=`block device
: The block device to use.

`OCF_RESKEY_homehost=`Homehost for mdadm
: The value for the homehost directive; this is an mdadm feature to protect RAIDs against being activated by accident. It is recommended to create RAIDs managed by the cluster with "homehost" set to a special value, so they are not accidentially auto-assembled by nodes not supposed to own them.

ocf:Route (7)

ocf:Route — Manages network routes

Synopsis

`OCF_RESKEY_destination=`string `OCF_RESKEY_device=`string `OCF_RESKEY_gateway=`string `OCF_RESKEY_source=`string [`OCF_RESKEY_table=`string] `Route` [start | stop | monitor | reload | meta-data | validate-all]

Description

Enables and disables network routes. Supports host and net routes, routes via a gateway address, and routes using specific source addresses. This resource agent is useful if a node's routing table needs to be manipulated based on node role assignment. Consider the following example use case: - One cluster node serves as an IPsec tunnel endpoint. - All other nodes use the IPsec tunnel to reach hosts in a specific remote network. Then, here is how you would implement this scheme making use of the Route resource agent: - Configure an ipsec LSB resource. - Configure a cloned Route OCF resource. - Create an order constraint to ensure that ipsec is started before Route. - Create a colocation constraint between the ipsec and Route resources, to make sure no instance of your cloned Route resource is started on the tunnel endpoint itself.

Supported Parameters

`OCF_RESKEY_destination=`Destination network
: The destination network (or host) to be configured for the route. Specify the netmask suffix in CIDR notation (e.g. "/24"). If no suffix is given, a host route will be created. Specify "0.0.0.0/0" or "default" if you want this resource to set the system default route.

`OCF_RESKEY_device=`Outgoing network device
: The outgoing network device to use for this route.

`OCF_RESKEY_gateway`=Gateway IP address
: The gateway IP address to use for this route.

`OCF_RESKEY_source`=Source IP address
: The source IP address to be configured for the route.

`OCF_RESKEY_table`=Routing table
: The routing table to be configured for the route.

ocf:rsyncd (7)

ocf:rsyncd — Manages an rsync daemon

Synopsis

[OCF_RESKEY_binpath=string] [OCF_RESKEY_conffile=string] [OCF_RESKEY_bwlimit=string] rsyncd [start | stop | monitor | validate-all | meta-data]

Description

This script manages rsync daemon

Supported Parameters

OCF_RESKEY_binpath=Full path to the rsync binary
The rsync binary path. For example, "/usr/bin/rsync"

OCF_RESKEY_conffile=Configuration file name with full path
The rsync daemon configuration file name with full path. For example, "/etc/rsyncd.conf"

OCF_RESKEY_bwlimit=limit I/O bandwidth, KBytes per second
This option allows you to specify a maximum transfer rate in kilobytes per second. This option is most effective when using rsync with large files (several megabytes and up). Due to the nature of rsync transfers, blocks of data are sent, then if rsync determines the transfer was too fast, it will wait before sending the next data block. The result is an average transfer rate equaling the specified limit. A value of zero specifies no limit.

ocf:SAPDatabase (7)

ocf:SAPDatabase — Manages any SAP database (based on Oracle, MaxDB, or DB2)

Synopsis

OCF_RESKEY_SID=string OCF_RESKEY_DIR_EXECUTABLE=string
OCF_RESKEY_DBTYPE=string OCF_RESKEY_NETSERVICENAME=string
OCF_RESKEY_DBJ2EE_ONLY=boolean OCF_RESKEY_JAVA_HOME=string
OCF_RESKEY_STRICT_MONITORING=boolean
OCF_RESKEY_AUTOMATIC_RECOVER=boolean
OCF_RESKEY_DIR_BOOTSTRAP=string OCF_RESKEY_DIR_SECSTORE=string
OCF_RESKEY_DB_JARS=string OCF_RESKEY_PRE_START_USEREXIT=string
OCF_RESKEY_POST_START_USEREXIT=string
OCF_RESKEY_PRE_STOP_USEREXIT=string
OCF_RESKEY_POST_STOP_USEREXIT=string SAPDatabase [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for SAP databases. It manages a SAP database of any type as an HA resource.

Supported Parameters

OCF_RESKEY_SID=SAP system ID
    The unique SAP system identifier. e.g. P01

OCF_RESKEY_DIR_EXECUTABLE=path of sapstartsrv and sapcontrol
    The full qualified path where to find sapstartsrv and sapcontrol.

OCF_RESKEY_DBTYPE=database vendor
    The name of the database vendor you use. Set either: ORA,DB6,ADA

`OCF_RESKEY_NETSERVICENAME`=listener name
: The Oracle TNS listener name.

`OCF_RESKEY_DBJ2EE_ONLY`=only JAVA stack installed
: If you do not have a ABAP stack installed in the SAP database, set this to TRUE

`OCF_RESKEY_JAVA_HOME`=Path to Java SDK
: This is only needed if the DBJ2EE_ONLY parameter is set to true. Enter the path to the Java SDK which is used by the SAP WebAS Java

`OCF_RESKEY_STRICT_MONITORING`=Activates application level monitoring
: This controls how the resource agent monitors the database. If set to true, it will use SAP tools to test the connect to the database. Do not use with Oracle, because it will result in unwanted failovers in case of an archiver stuck

`OCF_RESKEY_AUTOMATIC_RECOVER`=Enable or disable automatic startup recovery
: The SAPDatabase resource agent tries to recover a failed start attempt automaticaly one time. This is done by running a forced abort of the RDBMS and/or executing recovery commands.

`OCF_RESKEY_DIR_BOOTSTRAP`=path to j2ee bootstrap directory
: The full qualified path where to find the J2EE instance bootstrap directory. e.g. /usr/sap/P01/J00/j2ee/cluster/bootstrap

`OCF_RESKEY_DIR_SECSTORE`=path to j2ee secure store directory
: The full qualified path where to find the J2EE security store directory. e.g. /usr/sap/P01/SYS/global/security/lib/tools

`OCF_RESKEY_DB_JARS`=file name of the jdbc driver
: The full qualified filename of the jdbc driver for the database connection test. It will be automaticaly read from the bootstrap.properties file in Java engine 6.40 and 7.00. For Java engine 7.10 the parameter is mandatory.

`OCF_RESKEY_PRE_START_USEREXIT`=path to a pre-start script
: The full qualified path where to find a script or program which should be executed before this resource gets started.

`OCF_RESKEY_POST_START_USEREXIT`=path to a post-start script
: The full qualified path where to find a script or program which should be executed after this resource got started.

OCF_RESKEY_PRE_STOP_USEREXIT=path to a pre-start script

The full qualified path where to find a script or program which should be executed before this resource gets stopped.

OCF_RESKEY_POST_STOP_USEREXIT=path to a post-start script

The full qualified path where to find a script or program which should be executed after this resource got stopped.

ocf:SAPInstance (7)

ocf:SAPInstance — Manages a SAP instance

Synopsis

OCF_RESKEY_InstanceName=string OCF_RESKEY_DIR_EXECUTABLE=string OCF_RESKEY_DIR_PROFILE=string OCF_RESKEY_START_PROFILE=string OCF_RESKEY_START_WAITTIME=string OCF_RESKEY_AUTOMATIC_RECOVER=boolean OCF_RESKEY_MONITOR_SERVICES=string OCF_RESKEY_ERS_InstanceName=string OCF_RESKEY_ERS_START_PROFILE=string OCF_RESKEY_PRE_START_USEREXIT=string OCF_RESKEY_POST_START_USEREXIT=string OCF_RESKEY_PRE_STOP_USEREXIT=string OCF_RESKEY_POST_STOP_USEREXIT=string SAPInstance [start | stop | status | monitor | promote | demote | validate-all | meta-data | methods]

Description

Resource script for SAP. It manages a SAP Instance as an HA resource.

Supported Parameters

OCF_RESKEY_InstanceName=instance name: SID_INSTANCE_VIR-HOSTNAME
The full qualified SAP instance name. e.g. P01_DVEBMGS00_sapp01ci

OCF_RESKEY_DIR_EXECUTABLE=path of sapstartsrv and sapcontrol
The full qualified path where to find sapstartsrv and sapcontrol.

OCF_RESKEY_DIR_PROFILE=path of start profile
The full qualified path where to find the SAP START profile.

`OCF_RESKEY_START_PROFILE`=start profile name
: The name of the SAP START profile.

`OCF_RESKEY_START_WAITTIME`=Check the successful start after that time (do not wait for J2EE-Addin)
: After that time in seconds a monitor operation is executed by the resource agent. Does the monitor return SUCCESS, the start is handled as SUCCESS. This is useful to resolve timing problems with e.g. the J2EE-Addin instance.

`OCF_RESKEY_AUTOMATIC_RECOVER`=Enable or disable automatic startup recovery
: The SAPInstance resource agent tries to recover a failed start attempt automaticaly one time. This is done by killing runing instance processes and executing cleanipc.

`OCF_RESKEY_MONITOR_SERVICES`=

`OCF_RESKEY_ERS_InstanceName`=

`OCF_RESKEY_ERS_START_PROFILE`=

`OCF_RESKEY_PRE_START_USEREXIT`=path to a pre-start script
: The full qualified path where to find a script or program which should be executed before this resource gets started.

`OCF_RESKEY_POST_START_USEREXIT`=path to a post-start script
: The full qualified path where to find a script or program which should be executed after this resource got started.

`OCF_RESKEY_PRE_STOP_USEREXIT`=path to a pre-start script
: The full qualified path where to find a script or program which should be executed before this resource gets stopped.

`OCF_RESKEY_POST_STOP_USEREXIT`=path to a post-start script
: The full qualified path where to find a script or program which should be executed after this resource got stopped.

ocf:scsi2reservation (7)

ocf:scsi2reservation — scsi-2 reservation

Synopsis

[OCF_RESKEY_scsi_reserve=string] [OCF_RESKEY_sharedisk=string] [OCF_RESKEY_start_loop=string] scsi2reservation [start | stop | monitor | meta-data | validate-all]

Description

The scsi-2-reserve resource agent is a place holder for SCSI-2 reservation. A healthy instance of scsi-2-reserve resource, indicates the own of the specified SCSI device. This resource agent depends on the scsi_reserve from scsires package, which is Linux specific.

Supported Parameters

OCF_RESKEY_scsi_reserve=Manages exclusive access to shared storage media thrugh SCSI-2 reservations

> The scsi_reserve is a command from scsires package. It helps to issue SCSI-2 reservation on SCSI devices.

OCF_RESKEY_sharedisk= Shared disk.

> The shared disk that can be reserved.

OCF_RESKEY_start_loop= Times to re-try before giving up.

> We are going to try several times before giving up. Start_loop indicates how many times we are going to re-try.

ocf:SendArp (7)

ocf:SendArp — Broadcasts unsolicited ARP announcements

Synopsis

[`OCF_RESKEY_ip`=string] [`OCF_RESKEY_nic`=string] `SendArp` [start | stop | monitor | meta-data | validate-all]

Description

This script send out gratuitous Arp for an IP address

Supported Parameters

`OCF_RESKEY_ip`=IP address
: The IP address for sending arp package.

`OCF_RESKEY_nic`=NIC
: The nic for sending arp package.

ocf:ServeRAID (7)

ocf:ServeRAID — Enables and disables shared ServeRAID merge groups

Synopsis

[`OCF_RESKEY_serveraid`=integer] [`OCF_RESKEY_mergegroup`=integer] `ServeRAID` [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for ServeRAID. It enables/disables shared ServeRAID merge groups.

Supported Parameters

`OCF_RESKEY_serveraid`=serveraid
: The adapter number of the ServeRAID adapter.

`OCF_RESKEY_mergegroup`=mergegroup
: The logical drive under consideration.

ocf:sfex (7)

ocf:sfex — Manages exclusive acess to shared storage using Shared Disk File EXclusiveness (SF-EX)

Synopsis

[OCF_RESKEY_device=string] [OCF_RESKEY_index=integer]
[OCF_RESKEY_collision_timeout=integer]
[OCF_RESKEY_monitor_interval=integer]
[OCF_RESKEY_lock_timeout=integer] sfex [start | stop | monitor | meta-data]

Description

Resource script for SF-EX. It manages a shared storage medium exclusively .

Supported Parameters

OCF_RESKEY_device=block device
Block device path that stores exclusive control data.

OCF_RESKEY_index=index
Location in block device where exclusive control data is stored. 1 or more is specified. Default is 1.

OCF_RESKEY_collision_timeout=waiting time for lock acquisition
Waiting time when a collision of lock acquisition is detected. Default is 1 second.

OCF_RESKEY_monitor_interval=monitor interval
Monitor interval(sec). Default is 10 seconds

OCF_RESKEY_lock_timeout=Valid term of lock
Valid term of lock(sec). Default is 20 seconds.

ocf:SphinxSearchDaemon (7)

ocf:SphinxSearchDaemon — Manages the Sphinx search daemon.

Synopsis

`OCF_RESKEY_config`=string [`OCF_RESKEY_searchd`=string] [`OCF_RESKEY_search`=string] [`OCF_RESKEY_testQuery`=string] `SphinxSearchDaemon` [start | stop | monitor | meta-data | validate-all]

Description

This is a searchd Resource Agent. It manages the Sphinx Search Daemon.

Supported Parameters

`OCF_RESKEY_config`=Configuration file
searchd configuration file

`OCF_RESKEY_searchd`=searchd binary
searchd binary

`OCF_RESKEY_search`=search binary
Search binary for functional testing in the monitor action.

`OCF_RESKEY_testQuery`=test query
Test query for functional testing in the monitor action. The query does not need to match any documents in the index. The purpose is merely to test whether the search daemon is is able to query its indices and respond properly.

ocf:Squid (7)

ocf:Squid — Manages a Squid proxy server instance

Synopsis

[`OCF_RESKEY_squid_exe=`string] `OCF_RESKEY_squid_conf=`string `OCF_RESKEY_squid_pidfile=`string `OCF_RESKEY_squid_port=`integer [`OCF_RESKEY_squid_stop_timeout=`integer] [`OCF_RESKEY_debug_mode=`string] [`OCF_RESKEY_debug_log=`string] `Squid` [start | stop | status | monitor | meta-data | validate-all]

Description

The resource agent of Squid. This manages a Squid instance as an HA resource.

Supported Parameters

`OCF_RESKEY_squid_exe=`Executable file
: This is a required parameter. This parameter specifies squid's executable file.

`OCF_RESKEY_squid_conf=`Configuration file
: This is a required parameter. This parameter specifies a configuration file for a squid instance managed by this RA.

`OCF_RESKEY_squid_pidfile=`Pidfile
: This is a required parameter. This parameter specifies a process id file for a squid instance managed by this RA.

`OCF_RESKEY_squid_port=`Port number
: This is a required parameter. This parameter specifies a port number for a squid instance managed by this RA. If plural ports are used, you must specifiy the only one of them.

`OCF_RESKEY_squid_stop_timeout`=Number of seconds to await to confirm a normal stop method

This is an omittable parameter. On a stop action, a normal stop method is firstly used. and then the confirmation of its completion is awaited for the specified seconds by this parameter. The default value is 10.

`OCF_RESKEY_debug_mode`=Debug mode

This is an optional parameter. This RA runs in debug mode when this parameter includes 'x' or 'v'. If 'x' is included, both of STDOUT and STDERR redirect to the logfile specified by "debug_log", and then the builtin shell option 'x' is turned on. It is similar about 'v'.

`OCF_RESKEY_debug_log`=A destination of the debug log

This is an optional and omittable parameter. This parameter specifies a destination file for debug logs and works only if this RA run in debug mode. Refer to "debug_mode" about debug mode. If no value is given but it's requied, it's made by the following rules: "/var/log/" as a directory part, the basename of the configuration file given by "syslog_ng_conf" as a basename part, ".log" as a suffix.

ocf:Stateful (7)

ocf:Stateful — Example stateful resource agent

Synopsis

`OCF_RESKEY_state=`string `Stateful` [start | stop | monitor | meta-data | validate-all]

Description

This is an example resource agent that impliments two states

Supported Parameters

`OCF_RESKEY_state=`State file
: Location to store the resource state in

ocf:SysInfo (7)

ocf:SysInfo — Records various node attributes in the CIB

Synopsis

[`OCF_RESKEY_pidfile=`string] [`OCF_RESKEY_delay=`string] `SysInfo` [start | stop | monitor | meta-data | validate-all]

Description

This is a SysInfo Resource Agent. It records (in the CIB) various attributes of a node Sample Linux output: arch: i686 os: Linux-2.4.26-gentoo-r14 free_swap: 1999 cpu_info: Intel(R) Celeron(R) CPU 2.40GHz cpu_speed: 4771.02 cpu_cores: 1 cpu_load: 0.00 ram_total: 513 ram_free: 117 root_free: 2.4 Sample Darwin output: arch: i386 os: Darwin-8.6.2 cpu_info: Intel Core Duo cpu_speed: 2.16 cpu_cores: 2 cpu_load: 0.18 ram_total: 2016 ram_free: 787 root_free: 13 Units: free_swap: Mb ram_*: Mb root_free: Gb cpu_speed (Linux): bogomips cpu_speed (Darwin): Ghz

Supported Parameters

`OCF_RESKEY_pidfile=`PID file
: PID file

`OCF_RESKEY_delay=`Dampening Delay
: Interval to allow values to stabilize

ocf:syslog-ng (7)

ocf:syslog-ng — Syslog-ng resource agent

Synopsis

[`OCF_RESKEY_configfile`=string]
[`OCF_RESKEY_syslog_ng_binary`=string]
[`OCF_RESKEY_start_opts`=string]
[`OCF_RESKEY_kill_term_timeout`=integer] `syslog-ng` [start | stop | status | monitor | meta-data | validate-all]

Description

This script manages a syslog-ng instance as an HA resource.

Supported Parameters

`OCF_RESKEY_configfile`=Configuration file
: This parameter specifies a configuration file for a syslog-ng instance managed by this RA.

`OCF_RESKEY_syslog_ng_binary`=syslog-ng executable
: This parameter specifies syslog-ng's executable file.

`OCF_RESKEY_start_opts`=Start options
: This parameter specifies startup options for a syslog-ng instance managed by this RA. When no value is given, no startup options is used. Don't use option '-F'. It causes a stuck of a start action.

`OCF_RESKEY_kill_term_timeout`=Number of seconds to await to confirm a normal stop method
: On a stop action, a normal stop method(pkill -TERM) is firstly used. And then the confirmation of its completion is waited for the specified seconds by this parameter. The default value is 10.

ocf:tomcat (7)

ocf:tomcat — Manages a Tomcat servlet environment instance

Synopsis

`OCF_RESKEY_tomcat_name`=string `OCF_RESKEY_script_log`=string [`OCF_RESKEY_tomcat_stop_timeout`=integer] [`OCF_RESKEY_tomcat_suspend_trialcount`=integer] [`OCF_RESKEY_tomcat_user`=string] [`OCF_RESKEY_statusurl`=string] [`OCF_RESKEY_java_home`=string] `OCF_RESKEY_catalina_home`=string `OCF_RESKEY_catalina_pid`=string [`OCF_RESKEY_tomcat_start_opts`=string] [`OCF_RESKEY_catalina_opts`=string] [`OCF_RESKEY_catalina_rotate_log`=string] [`OCF_RESKEY_catalina_rotatetime`=integer] `tomcat` [start | stop | status | monitor | meta-data | validate-all]

Description

Resource script for Tomcat. It manages a Tomcat instance as a cluster resource.

Supported Parameters

`OCF_RESKEY_tomcat_name`=The name of the resource
: The name of the resource, added as a Java parameter in JAVA_OPTS: -Dname=<tomcat_name> to Tomcat process on start. Used to ensure process is still running and must be unique.

`OCF_RESKEY_script_log`=Log file
: Log file, used during start and stop operations.

`OCF_RESKEY_tomcat_stop_timeout`=Time-out for the stop operation
: Time-out for stop operation.

`OCF_RESKEY_tomcat_suspend_trialcount`=Max retry count for stop operation
Maximum number of times to retry stop operation before suspending and killing Tomcat.

`OCF_RESKEY_tomcat_user`=The user who starts Tomcat
The user who starts Tomcat.

`OCF_RESKEY_statusurl`=URL for state confirmation
URL for state confirmation.

`OCF_RESKEY_java_home`=Home directory of Java
Home directory of Java.

`OCF_RESKEY_catalina_home`=Home directory of Tomcat
Home directory of Tomcat.

`OCF_RESKEY_catalina_pid`=A PID file name for Tomcat
A PID file name for Tomcat.

`OCF_RESKEY_tomcat_start_opts`=Tomcat start options
Tomcat start options.

`OCF_RESKEY_catalina_opts`=Catalina options
Catalina options, for the start operation only.

`OCF_RESKEY_catalina_rotate_log`=Rotate catalina.out flag
Rotate catalina.out flag.

`OCF_RESKEY_catalina_rotatetime`=catalina.out rotation interval (seconds)
catalina.out rotation interval (seconds).

ocf:VIPArip (7)

ocf:VIPArip — Manages a virtual IP address through RIP2

Synopsis

`OCF_RESKEY_ip`=string [`OCF_RESKEY_nic`=string] [`OCF_RESKEY_zebra_binary`=string] [`OCF_RESKEY_ripd_binary`=string] `VIPArip` [start | stop | monitor | validate-all | meta-data]

Description

Virtual IP Address by RIP2 protocol. This script manages IP alias in different subnet with quagga/ripd. It can add an IP alias, or remove one.

Supported Parameters

`OCF_RESKEY_ip`=The IP address in different subnet
: The IPv4 address in different subnet, for example "192.168.1.1".

`OCF_RESKEY_nic`=The nic for broadcast the route information
: The nic for broadcast the route information. The ripd uses this nic to broadcast the route informaton to others

`OCF_RESKEY_zebra_binary`=zebra binary
: Absolute path to the zebra binary.

`OCF_RESKEY_ripd_binary`=ripd binary
: Absolute path to the ripd binary.

ocf:VirtualDomain (7)

ocf:VirtualDomain — Manages virtual domains through the libvirt virtualization framework

Synopsis

`OCF_RESKEY_config`=string [`OCF_RESKEY_hypervisor`=string] [`OCF_RESKEY_force_stop`=boolean] [`OCF_RESKEY_migration_transport`=string] [`OCF_RESKEY_monitor_scripts`=string] `VirtualDomain` [start | stop | status | monitor | migrate_from | migrate_to | meta-data | validate-all]

Description

Resource agent for a virtual domain (a.k.a. domU, virtual machine, virtual environment etc., depending on context) managed by libvirtd.

Supported Parameters

`OCF_RESKEY_config`=Virtual domain configuration file
: Absolute path to the libvirt configuration file, for this virtual domain.

`OCF_RESKEY_hypervisor`=Hypervisor URI
: Hypervisor URI to connect to. See the libvirt documentation for details on supported URI formats. The default is system dependent.

`OCF_RESKEY_force_stop`=Always force shutdown on stop
: Always forcefully shut down ("destroy") the domain on stop. The default behavior is to resort to a forceful shutdown only after a graceful shutdown attempt has failed. You should only set this to true if your virtual domain (or your virtualization backend) does not support graceful shutdown.

`OCF_RESKEY_migration_transport`=Remote hypervisor transport
: Transport used to connect to the remote hypervisor while migrating. Please refer to the libvirt documentation for details on transports available. If this parameter is omitted, the resource will use libvirt's default transport to connect to the remote hypervisor.

`OCF_RESKEY_monitor_scripts`=space-separated list of monitor scripts
: To additionally monitor services within the virtual domain, add this parameter with a list of scripts to monitor. Note: when monitor scripts are used, the start and migrate_from operations will complete only when all monitor scripts have completed successfully. Be sure to set the timeout of these operations to accommodate this delay.

ocf:vmware (7)

ocf:vmware — Manages VMWare Server 2.0 virtual machines

Synopsis

[OCF_RESKEY_vmxpath=string] [OCF_RESKEY_vimshbin=string] vmware
[start | stop | monitor | meta-data]

Description

OCF compliant script to control vmware server 2.0 virtual machines.

Supported Parameters

OCF_RESKEY_vmxpath=VMX file path
VMX configuration file path

OCF_RESKEY_vimshbin=vmware-vim-cmd path
vmware-vim-cmd executable path

ocf:WAS6 (7)

ocf:WAS6 — Manages a WebSphere Application Server 6 instance

Synopsis

[`OCF_RESKEY_profile=`string] `WAS6` [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for WAS6. It manages a Websphere Application Server (WAS6) as an HA resource.

Supported Parameters

`OCF_RESKEY_profile=`profile name
    The WAS profile name.

ocf:WAS (7)

ocf:WAS — Manages a WebSphere Application Server instance

Synopsis

[`OCF_RESKEY_config=`string] [`OCF_RESKEY_port=`integer] `WAS` [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for WAS. It manages a Websphere Application Server (WAS) as an HA resource.

Supported Parameters

`OCF_RESKEY_config=`configration file
    The WAS-configuration file.

`OCF_RESKEY_port=`port
    The WAS-(snoop)-port-number.

ocf:WinPopup (7)

ocf:WinPopup — Sends an SMB notification message to selected hosts

Synopsis

[`OCF_RESKEY_hostfile=`string] `WinPopup` [start | stop | status | monitor | validate-all | meta-data]

Description

Resource script for WinPopup. It sends WinPopups message to a sysadmin's workstation whenever a takeover occurs.

Supported Parameters

`OCF_RESKEY_hostfile=`Host file
: The file containing the hosts to send WinPopup messages to.

ocf:Xen (7)

ocf:Xen — Manages Xen unprivileged domains (DomUs)

Synopsis

[OCF_RESKEY_xmfile=string] [OCF_RESKEY_name=string] [OCF_RESKEY_shutdown_timeout=boolean] [OCF_RESKEY_allow_mem_management=boolean] [OCF_RESKEY_reserved_Dom0_memory=string] [OCF_RESKEY_monitor_scripts=string] Xen [start | stop | migrate_from | migrate_to | monitor | meta-data | validate-all]

Description

Resource Agent for the Xen Hypervisor. Manages Xen virtual machine instances by mapping cluster resource start and stop, to Xen create and shutdown, respectively. A note on names We will try to extract the name from the config file (the xmfile attribute). If you use a simple assignment statement, then you should be fine. Otherwise, if there's some python acrobacy involved such as dynamically assigning names depending on other variables, and we will try to detect this, then please set the name attribute. You should also do that if there is any chance of a pathological situation where a config file might be missing, for example if it resides on a shared storage. If all fails, we finally fall back to the instance id to preserve backward compatibility. Para-virtualized guests can also be migrated by enabling the meta_attribute allow-migrate.

Supported Parameters

OCF_RESKEY_xmfile=Xen control file
: Absolute path to the Xen control file, for this virtual machine.

OCF_RESKEY_name=Xen DomU name
: Name of the virtual machine.

`OCF_RESKEY_shutdown_timeout`=Shutdown escalation timeout
: The Xen agent will first try an orderly shutdown using xm shutdown. Should this not succeed within this timeout, the agent will escalate to xm destroy, forcibly killing the node. If this is not set, it will default to two-third of the stop action timeout. Setting this value to 0 forces an immediate destroy.

`OCF_RESKEY_allow_mem_management`=Use dynamic memory management
: This parameter enables dynamic adjustment of memory for start and stop actions used for Dom0 and the DomUs. The default is to not adjust memory dynamically.

`OCF_RESKEY_reserved_Dom0_memory`=Minimum Dom0 memory
: In case memory management is used, this parameter defines the minimum amount of memory to be reserved for the dom0. The default minimum memory is 512MB.

`OCF_RESKEY_monitor_scripts`=list of space separated monitor scripts
: To additionally monitor services within the unprivileged domain, add this parameter with a list of scripts to monitor.

ocf:Xinetd (7)

ocf:Xinetd — Manages an Xinetd service

Synopsis

[`OCF_RESKEY_service`=string] `Xinetd` [start | stop | restart | status | monitor | validate-all | meta-data]

Description

Resource script for Xinetd. It starts/stops services managed by xinetd. Note that the xinetd daemon itself must be running: we are not going to start it or stop it ourselves. Important: in case the services managed by the cluster are the only ones enabled, you should specify the -stayalive option for xinetd or it will exit on Heartbeat stop. Alternatively, you may enable some internal service such as echo.

Supported Parameters

`OCF_RESKEY_service`=service name
: The service name managed by xinetd.

Part V. Appendix

Example of Setting Up a Simple Testing Resource

A

This chapter provides a basic example for the configuration of a simple resource: an IP address. It demonstrates both approaches to do so, using either the Pacemaker GUI or the `crm` command line tool.

For the following example, we assume that you have set up your cluster as described in Chapter 3, *Installation and Basic Setup with YaST* (page 21) and that your cluster consists of at least two nodes. For an introduction and overview of how to configure cluster resources with the Pacemaker GUI and the `crm` shell, refer to the following chapters:

- *Configuring and Managing Cluster Resources (GUI)* (page 57)
- *Configuring and Managing Cluster Resources (Command Line)* (page 89)

A.1 Configuring a Resource with the GUI

Creating a sample cluster resource and migrating it to another server can help you test to ensure your cluster is functioning properly. A simple resource to configure and migrate is an IP address.

Procedure A.1 *Creating an IP Address Cluster Resource*

1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Connecting to a Cluster” (page 58).

2 In the left pane, switch to the *Resources* view and in the right pane, select the group to modify and click *Edit*. The next window shows the basic group parameters and the meta attributes and primitives already defined for that resource.

3 Click the *Primitives* tab and click *Add*.

4 In the next dialog, set the following parameters to add an IP address as sub-resource of the group:

> **4a** Enter a unique `ID`. For example, `myIP`.
>
> **4b** From the *Class* list, select *ocf* as resource agent class.
>
> **4c** As *Provider* of your OCF resource agent, select *heartbeat*.
>
> **4d** From the *Type* list, select *IPaddr* as resource agent.
>
> **4e** Click *Forward*.
>
> **4f** In the *Instance Attribute* tab, select the *IP* entry and click *Edit* (or double-click the *IP* entry).
>
> **4g** As *Value*, enter the desired IP address, for example, `10.10.0.1` and click *OK*.
>
> **4h** *Add* a new instance attribute and specify `nic` as *Name* and `eth0` as *Value*, then click *OK*.
>
> The name and value are dependent on your hardware configuration and what you chose for the media configuration during the installation of the High Availability Extension software.

5 Once all parameters are set according to your wishes, click *OK* to finish the configuration of that resource. The configuration dialog is closed and the main window shows the modified resource.

To start the resource with the Pacemaker GUI, select *Management* in the left pane. In the right pane, right-click the resource and select *Start* (or start it from the toolbar).

To migrate the IP address resource to another node (`saturn)` proceed as follows:

Procedure A.2 *Migrating Resources to Another Node*

1 Switch to the *Management* view in the left pane, then right-click the IP address resource in the right pane and select *Migrate Resource*.

2 In the new window, select `saturn` from the *To Node* drop-down list to move the selected resource to the node `saturn`.

3 If you want to migrate the resource only temporarily, activate *Duration* and enter the time frame for which the resource should migrate to the new node.

4 Click *OK* to confirm the migration.

A.2 Configuring a Resource Manually

Resources are any type of service that a computer provides. Resources are known to High Availability when they may be controlled by RAs (Resource Agents), which are LSB scripts, OCF scripts, or legacy Heartbeat 1 resources. All resources can be configured with the `crm` command or as XML in the CIB (Cluster Information Base) in the `resources` section. For an overview of available resources, look at Chapter 19, *HA OCF Agents* (page 269).

To add an IP address `10.10.0.1` as a resource to the current configuration, use the `crm` command:

Procedure A.3 *Creating an IP Address Cluster Resource*

1 Open a shell and become `root`.

2 Enter `crm configure` to open the internal shell.

3 Create an IP address resource:

```
crm(live)configure# resource
primitive myIP ocf:heartbeat:IPaddr params ip=10.10.0.1
```

NOTE

When configuring a resource with High Availability, the same resource should not be initialized by `init`. High availability is be responsible for all service start or stop actions.

If the configuration was successful, a new resource appears in `crm_mon` that is started on a random node of your cluster.

To migrate a resource to another node, do the following:

Procedure A.4 *Migrating Resources to Another Node*

1 Start a shell and become the user `root`.

2 Migrate your resource `myip` to node `saturn`:

```
crm resource migrate myIP saturn
```

Example Configuration for OCFS2 and cLVM

B

The following is an example configuration that can help you setting up your resources for use of either OCFS2, cLVM, or both. The configuration below does not represent a complete cluster configuration but is only an extract, including all resources needed for OCFS2 and cLVM, and ignoring any other resources that you might need. Attributes and attribute values may need adjustment to your specific setup.

Example B.1 *Cluster Configuration for OCFS2 and cLVM*

```
primitive clvm ocf:lvm2:clvmd \
     params daemon_timeout="30" \
     op monitor interval="60" timeout="60"
primitive dlm ocf:pacemaker:controld \
     op monitor interval="60" timeout="60"
primitive o2cb ocf:ocfs2:o2cb \
     op monitor interval="60" timeout="60"
primitive ocfs2-1 ocf:heartbeat:Filesystem \
     params device="/dev/sdb1" directory="/mnt/shared" fstype="ocfs2"
options="acl" \
     op monitor interval="20" timeout="40"
primitive sbd_stonith stonith:external/sbd \
     meta target-role="Started" op monitor interval="15" \
     timeout="15" start-delay="15" \
     params sbd_device="/dev/sdb2"
primitive vg1 ocf:heartbeat:LVM \
     params volgrpname="cluster-vg" \
     op monitor interval="60" timeout="60"
group base-group dlm o2cb clvm vg1 ocfs2-1
clone base-clone base-group \
     meta interleave="true"
```

The configuration with a base group (including several primitives) and a base clone simplifies the overall setup: The base group has internal collocation and ordering and can always remain the same, apart from two resources:

- `vg1`—the resource for the volume group. Only configure this resource if your setup includes cVLM. Otherwise omit it from the cluster configuration and from the base group.

- `ocfs2-1`—the resource for mounting the OCFS2 file system. Only configure this resource if your setup includes OCFS2. Otherwise omit it from the cluster configuration and from the base group.

All of the other resources mentioned in Example B.1, “Cluster Configuration for OCFS2 and cLVM” (page 371) can be configured and be running in the cluster regardless of your setup.

Upgrading Your Cluster to the Latest Product Version

C

If you have an existing cluster based on SUSE® Linux Enterprise Server 10, you can update your cluster to run with the High Availability Extension on SUSE Linux Enterprise Server 11 or 11 SP1.

For migrating from SUSE Linux Enterprise Server 10 to SUSE Linux Enterprise Server 11 or 11 SP1, all cluster nodes must be offline and the cluster must be migrated as a whole—mixed clusters running on SUSE Linux Enterprise Server 10/SUSE Linux Enterprise Server 11 are not supported.

C.1 Upgrading from SLES 10 to SLEHA 11

For convenience, SUSE® Linux Enterprise High Availability Extension includes a `hb2openais.sh` script with which to convert your data while moving from the Heartbeat to the OpenAIS cluster stack. The script parses the configuration stored in `/etc/ha.d/ha.cf` and generates a new configuration file for the OpenAIS cluster stack. Furthermore, it adjusts the CIB to match the OpenAIS conventions, converts the OCFS2 file system and replaces EVMS with cLVM. Any EVMS2 containers are converted to cLVM2 volumes. For volume groups referenced in existing resources in the CIB, new LVM resources are created.

To successfully migrate your cluster from SUSE Linux Enterprise Server 10 SP3 to SUSE Linux Enterprise Server 11, you need to execute the following steps:

1. Preparing your SUSE Linux Enterprise Server 10 SP3 Cluster (page 374)

2. Updating to SUSE Linux Enterprise 11 (page 375)

3. Testing the Conversion (page 376)

4. Converting the Data (page 377)

After the conversion has been successfully completed, you can bring the updated cluster online again.

NOTE: Reverting after Update

After the update process to SUSE Linux Enterprise Server 11, reverting back to SUSE Linux Enterprise Server 10 is *not* supported.

C.1.1 Preparation and Backup

Before updating your cluster to the next product version and converting the data accordingly, you need to prepare your current cluster.

Procedure C.1 *Preparing your SUSE Linux Enterprise Server 10 SP3 Cluster*

1 Log in to the cluster.

2 Review the Heartbeat configuration file `/etc/ha.d/ha.cf` and check that all communication media support multicasting.

3 Make sure the following files are equal on all nodes: `/etc/ha.d/ha.cf` and `/var/lib/heartbeat/crm/cib.xml`.

4 Take all nodes offline by executing `rcheartbeat stop` on each node.

5 In addition to the general system backup recommended before updating to the latest version, back up the following files, as you need them for running the conversion script after the update to SUSE Linux Enterprise Server 11:

- `/var/lib/heartbeat/crm/cib.xml`

- `/var/lib/heartbeat/hostcache`

- /etc/ha.d/ha.cf

- /etc/logd.cf

6 If you have EVMS2 resources, convert non-LVM EVMS2 volumes to compatibility volumes on SUSE Linux Enterprise Server 10. During the conversion process (see Section C.1.3, “Data Conversion” (page 375)), these are then turned into LVM2 volume groups. After conversion, make sure to mark each volume group as a member of the High Availability cluster with `vgchange -c y`.

C.1.2 Update/Installation

After preparing the cluster and backing up the files, you can start updating the cluster nodes to the next product version. Instead of running an update, you can also do a fresh installation of SUSE Linux Enterprise 11 on your cluster nodes.

Procedure C.2 *Updating to SUSE Linux Enterprise 11*

1 On all cluster nodes, perform an update from SUSE Linux Enterprise Server 10 SP3 to SUSE Linux Enterprise Server 11. For information on how to update your product, refer to the SUSE Linux Enterprise Server 11 *Deployment Guide*, chapter *Updating SUSE Linux Enterprise*.

   Conversely, you can also freshly install SUSE Linux Enterprise Server 11 on all cluster nodes.

2 On all cluster nodes, install SUSE Linux Enterprise High Availability Extension 11 as add-on on top of SUSE Linux Enterprise Server. For detailed information, see Section 3.1, “Installing the High Availability Extension” (page 21).

C.1.3 Data Conversion

After having installed SUSE Linux Enterprise Server 11 and the High Availability Extension, you can start with the data conversion. The conversion script shipped with the High Availability Extension has been set up with care, but it cannot handle all setups in fully automatic mode. It alerts you of the changes it makes, but needs interaction and decisions from your side. You need to know your cluster in detail—it is up to you to verify that the changes are meaningful. The conversion script is located in `/usr/lib/heartbeat` (or in `/usr/lib64/heartbeat`, if you are using a 64-bit system).

NOTE: Executing Test Runs

To make yourself familiar with the conversion process, we highly recommend that you test the conversion first (without making any changes). You can use the same test directory to do repeated test runs, but you only need to copy the files once.

Procedure C.3 *Testing the Conversion*

1 On one of the nodes, create a test directory and copy the backup files to the test directory:

```
$ mkdir /tmp/hb2openais-testdir
$ cp /etc/ha.d/ha.cf /tmp/hb2openais-testdir
$ cp /var/lib/heartbeat/hostcache /tmp/hb2openais-testdir
$ cp /etc/logd.cf /tmp/hb2openais-testdir
$ sudo cp /var/lib/heartbeat/crm/cib.xml /tmp/hb2openais-testdir
```

2 Start the test run with

```
$ /usr/lib/heartbeat/hb2openais.sh -T /tmp/hb2openais-testdir -U
```

or with the following command, if you are using a 64-bit system:

```
$ /usr/lib64/heartbeat/hb2openais.sh -T /tmp/hb2openais-testdir -U
```

3 Read and verify the resulting `openais.conf` and `cib-out.xml` files:

```
$ cd  /tmp/hb2openais-testdir
$ less openais.conf
$ crm_verify -V -x cib-out.xml
```

For detailed information about the conversion stages, refer to `/usr/share/doc/packages/pacemaker/README.hb2openais` in your installed High Availability Extension.

Procedure C.4 *Converting the Data*

After doing a test run and checking the output, you can now start with the data conversion. You only need to run the conversion on *one* node. The main cluster configuration (the CIB) is automatically replicated to the other nodes. All other files that need to be replicated are automatically copied by the conversion script.

1 Make sure that `sshd` is running on all nodes with access allowed for `root` in order for the conversion script to successfully copy the files to the other cluster nodes.

2 Make sure that all ocfs2 filesystems are unmounted.

3 The High Availability Extension ships with a default OpenAIS configuration file. If you want to prevent the default configuration from being overwritten during the following steps, make a copy of the `/etc/ais/openais.conf` configuration file.

4 Start the conversion script as `root`. If using sudo, specify the privileged user using the `-u` option:

```
$ /usr/lib/heartbeat/hb2openais.sh -u root
```

Based on the configuration stored in `/etc/ha.d/ha.cf`, the script will generate a new configuration file for the OpenAIS cluster stack, `/etc/ais/openais.conf`. It will also analyze the CIB configuration and let you know if your cluster configuration requires changes, due to the change from Heartbeat to OpenAIS. All file processing is done on the node where conversion runs and replicated to the other nodes.

5 Follow the instructions on the screen.

After the conversion has been finished successfully, start the new cluster stack as described in Section 3.3, “Bringing the Cluster Online” (page 30).

After the upgrade process, reverting back to SUSE Linux Enterprise Server 10 is not supported.

C.1.4 For More Information

For more details about the conversion script and the stages of the conversion, refer to /usr/share/doc/packages/pacemaker/README.hb2openais in your installed High Availability Extension.

C.2 Upgrading from SLEHA 11 to SLEHA 11 SP1

To successfully migrate an existing cluster from SUSE Linux Enterprise High Availability Extension 11 to 11 SP1 you can do a “rolling upgrade”, meaning upgrading one node after the other. As the main cluster configuration file has changed from /etc/ais/openais.conf to /etc/corosync/corosync.conf with SUSE Linux Enterprise High Availability Extension 11 SP1, a script takes care of the necessary conversions. They are executed automatically when the openais package is updated.

Procedure C.5 *Performing a Rolling Upgrade*

IMPORTANT: Updating Software Packages

If you want to update any software packages on a node that is part of a running cluster, stop the cluster stack on that node before starting the software update. To stop the cluster stack, log in to the node as root and enter rcopenais stop.

If OpenAIS/Corosync is running during the software update, this can lead to unpredictable results like fencing of active nodes.

1 Log in as root on the node that you want to upgrade and stop OpenAIS:

```
rcopenais stop
```

2 Check that your system backup is up-to-date and restorable.

3 Perform an upgrade from SUSE Linux Enterprise Server 11 to SUSE Linux Enterprise Server 11 SP1 and from SUSE Linux Enterprise High Availability Extension 11 to SUSE Linux Enterprise High Availability Extension 11 SP1. For information on how to update your product, refer to the SUSE Linux Enterprise Server 11 SP1 *Deployment Guide*, chapter *Updating SUSE Linux Enterprise*.

4 Restart OpenAIS/Corosync on the upgraded node to make the node rejoin the cluster:

```
rcopenais start
```

5 Take the next node offline and repeat the procedure for that node.

D

What's New?

The software modifications from version to version are outlined in detail in the following sections. This summary indicates, for example, whether basic settings have been completely reconfigured, configuration files have been moved to other places, or other significant changes happened.

D.1 Version 10 SP3 to Version 11

With SUSE Linux Enterprise Server 11, the cluster stack has changed from Heartbeat to OpenAIS. OpenAIS implements an industry standard API, the Application Interface Specification (AIS), published by the Service Availability Forum. The cluster resource manager from SUSE Linux Enterprise Server 10 has been retained but has been significantly enhanced, ported to OpenAIS and is now known as Pacemaker.

For more details what changed in the High Availability components from SUSE® Linux Enterprise Server 10 SP3 to SUSE Linux Enterprise Server 11, refer to the following sections.

D.1.1 New Features and Functions Added

Migration Threshold and Failure Timeouts

> The High Availability Extension now comes with the concept of a migration threshold and failure timeout. You can define a number of failures for resources, after which they will migrate to a new node. By default, the node will no longer be allowed to run the failed resource until the administrator manually resets the

resource's failcount. However it is also possible to expire them by setting the resource's `failure-timeout` option.

Resource and Operation Defaults

You can now set global defaults for resource options and operations.

Support for Offline Configuration Changes

Often it is desirable to preview the effects of a series of changes before updating the configuration atomically. You can now create a “shadow” copy of the configuration that can be edited with the command line interface, before committing it and thus changing the active cluster configuration atomically.

Reusing Rules, Options and Sets of Operations

Rules, instance_attributes, meta_attributes and sets of operations can be defined once and referenced in multiple places.

Using XPath Expressions for Certain Operations in the CIB

The CIB now accepts XPath-based `create`, `modify`, `delete` operations. For more information, refer to the `cibadmin` help text.

Multi-dimensional Collocation and Ordering Constraints

For creating a set of collocated resources, previously you could either define a resource group (which could not always accurately express the design) or you could define each relationship as an individual constraint—causing a constraint explosion as the number of resources and combinations grew. Now you can also use an alternate form of collocation constraints by defining `resource_sets`.

Connection to the CIB From Non-cluster Machines

Provided Pacemaker is installed on a machine, it is possible to connect to the cluster even if the machine itself is not a part of it.

Triggering Recurring Actions at Known Times

By default, recurring actions are scheduled relative to when the resource started, but this is not always desirable. To specify a date/time that the operation should be relative to, set the operation's interval-origin. The cluster uses this point to calculate the correct start-delay such that the operation will occur at origin + (interval * N).

D.1.2 Changed Features and Functions

Naming Conventions for Resource and Custer Options

All resource and cluster options now use dashes (-) instead of underscores (_). For example, the `master_max` meta option has been renamed to `master-max`.

Renaming of `master_slave` Resource

The `master_slave` resource has been renamed to `master`. Master resources are a special type of clone that can operate in one of two modes.

Container Tag for Attributes

The `attributes` container tag has been removed.

Operation Field for Prerequisites

The `pre-req` operation field has been renamed `requires`.

Interval for Operations

All operations must have an interval. For start/stop actions the interval must be set to `0` (zero).

Attributes for Collocation and Ordering Constraints

The attributes of collocation and ordering constraints were renamed for clarity.

Cluster Options for Migration Due to Failure

The `resource-failure-stickiness` cluster option has been replaced by the `migration-threshold` cluster option. See also Migration Threshold and Failure Timeouts (page 381).

Arguments for Command Line Tools

The arguments for command-line tools have been made consistent. See also Naming Conventions for Resource and Custer Options (page 383).

Validating and Parsing XML

The cluster configuration is written in XML. Instead of a Document Type Definition (DTD), now a more powerful RELAX NG schema is used to define the pattern for the structure and content. `libxml2` is used as parser.

`id` Fields

`id` fields are now XML IDs which have the following limitations:

- IDs cannot contain colons.
- IDs cannot begin with a number.
- IDs must be globally unique (not just unique for that tag).

References to Other Objects
: Some fields (such as those in constraints that refer to resources) are IDREFs. This means that they must reference existing resources or objects in order for the configuration to be valid. Removing an object which is referenced elsewhere will therefore fail.

D.1.3 Removed Features and Functions

Setting Resource Meta Options
: It is no longer possible to set resource meta-options as top-level attributes. Use meta attributes instead. See also crm_resource(8) (page 242).

Setting Global Defaults
: Resource and operation defaults are no longer read from `crm_config`.

D.2 Version 11 to Version 11 SP1

Cluster Configuration File
: The main cluster configuration file has changed from `/etc/ais/openais.conf` to `/etc/corosync/corosync.conf`. Both files are very similar. When upgrading from SUSE Linux Enterprise High Availability Extension 11 to SP1, a script takes care of the minor differences between those files. For more information about the relationship between OpenAIS and Corosync, see . [`http://www.corosync.org/doku.php?id=faq:why`]

Rolling Upgrade
: In order to migrate existing clusters with minimal downtime, SUSE Linux Enterprise High Availability Extension allows you to perform a “rolling upgrade” from SUSE Linux Enterprise High Availability Extension 11 to 11 SP1. The cluster is still online while you upgrade one node after the other.

Automatic Cluster Deployment

For easier cluster deployment, AutoYaST allows you to clone existing nodes. AutoYaST is a system for installing one or more SUSE Linux Enterprise systems automatically and without user intervention, using an AutoYaST profile that contains installation and configuration data. The profile tells AutoYaST what to install and how to configure the installed system to get a completely ready-to-use system in the end. This profile can be used for mass deployment in different ways.

Transfer of Configuration Files

SUSE Linux Enterprise High Availability Extension ships with Csync2, a tool for replication of configuration files across all nodes in the cluster. It can handle any number of hosts and it is also possible to synchronize files among certain subgroups of hosts only. Use YaST to configure the hostnames and the files that should be synchronized with Csync2.

Web-Interface for Cluster Management

The High Availability Extension now also includes the HA Web Konsole, a Web-based user interface for management tasks. It allows you to monitor and administer your Linux cluster also from non-Linux machines. It is also an ideal solution in case your system does not provide or allow a graphical user interface.

Templates for Resource Configuration

When using the command line interface to create and configure resources, you can now choose from various resource templates for quicker and easier configuration.

Load-based Placement of Resources

By defining the capacity a certain node *provides*, the capacity a certain resource *requires* and by choosing one of several placement strategies in the cluster, resources can be placed according to their load impact to prevent decrease of cluster performance.

Cluster-aware Active/Active RAID1

It is now possible to create disaster-resilient storage configurations from two independent SANs, using `cmirrord`.

Read-only GFS2 Support

For easier migration from GFS2 to OCFS2, you can mount your GFS2 file systems in read-only mode to copy the data to an OCFS2 file system. OCFS2 is fully supported by SUSE Linux Enterprise High Availability Extension.

SCTP Support for OCFS2
: If redundant rings are configured, OCFS2 and DLM can automatically use redundant communication paths via SCTP, independent of network device bonding.

Storage Protection
: For additional layers of security in protecting your storage from data corruption, you can use a combination of IO fencing (with the `external/sbd` fencing device) and the `sfex` resource agent to ensure exclusive storage access.

Samba Clustering
: The High Availability Extension now supports CTDB, the cluster implementation of the trivial database. This allows you configure a clustered Samba server—providing an High Availability solution also for heterogeneous environments.

YaST Module for IP Load Balancing
: The new module allows configuration of kernel-based load balancing with a graphical user interface. It is a front-end for `ldirectord`, a user-space daemon for managing Linux Virtual Server and monitoring the real servers.

E

GNU Licenses

This appendix contains the GNU General Public License version 2 and the GNU Free Documentation License version 1.2.

GNU General Public License

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation’s software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author’s protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors’ reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone’s free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The “Program”, below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for non-commercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the

Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and an idea of what it does.
Copyright (C) yyyy name of author
```

```
This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.


This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.


You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type `show w'. This is free software, and you are welcome
to redistribute it under certain conditions; type `show c'
for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright
interest in the program `Gnomovision'
(which makes passes at compilers) written
by James Hacker.


signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License [http://www.fsf.org/licenses/lgpl.html] instead of this License.

GNU Free Documentation License

Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
   Copyright (c) YEAR YOUR NAME.
```

```
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled “GNU
Free Documentation License”.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Terminology

active/active, active/passive

A concept about how services are running on nodes. An active-passive scenario means that one or more services are running on the active node and the passive nodes waits for the active node to fail. Active-active means that each node is active and passive at the same time.

cluster

A high-performance cluster is a group of computers (real or virtual) sharing the application load in order to achieve faster results. A high-availability cluster is designed primarily to secure the highest possible availability of services.

cluster information base (CIB)

A representation of the whole cluster configuration and status (node membership, resources, constraints, etc.) written in XML and residing in memory. A master CIB is kept and maintained on the designated coordinator (DC) (page 396) and replicated to the other nodes.

cluster partition

Whenever communication fails between one or more nodes and the rest of the cluster, a cluster partition occurs. The nodes of a cluster partition are still active and able to communicate with each other, but they are unaware of the nodes with which they cannot communicate. As the loss of the other partition cannot be confirmed, a split brain scenario develops (see also split brain (page 398)).

cluster resource manager (CRM)

The main management entity responsible for coordinating all non-local interactions. Each node of the cluster has its own CRM, but the one running on the DC is the one elected to relay decisions to the other non-local CRMs and process their input. A CRM interacts with a number of components: local resource managers both on its own node and on the other nodes, non-local CRMs, administrative commands, the fencing functionality, and the membership layer.

consensus cluster membership (CCM)

The CCM determines which nodes make up the cluster and shares this information across the cluster. Any new addition and any loss of nodes or quorum is delivered by the CCM. A CCM module runs on each node of the cluster.

designated coordinator (DC)

The “master” node. This node is where the master copy of the CIB is kept. All other nodes get their configuration and resource allocation information from the current DC. The DC is elected from all nodes in the cluster after a membership change.

distributed lock manager (DLM)

DLM coordinates disk access for clustered file systems and administers file locking to increase performance and availability.

distributed replicated block device (drbd)

DRBD is a block device designed for building high availability clusters. The whole block device is mirrored via a dedicated network and is seen as a network RAID-1.

failover

Occurs when a resource or node fails on one machine and the affected resources are started on another node.

fencing

Describes the concept of preventing access to a shared resource by non-cluster members. It can be archived by killing (shutting down) a “misbehaving” node to prevent it from causing trouble, locking resources away from a node whose status is uncertain, or in several other ways. Furthermore, fencing is distinguished between node and resource fencing.

Heartbeat resource agent

Heartbeat resource agents were widely used with Heartbeat version 1. Their use is deprecated, but still supported in version 2. A Heartbeat resource agent can perform `start`, `stop`, and `status` operations and resides under `/etc/ha.d/resource.d` or `/etc/init.d`. For more information about Heartbeat resource agents, refer to http://www.linux-ha.org/HeartbeatResourceAgent (see also OCF resource agent (page 397)).

local resource manager (LRM)

The local resource manager (LRM) is responsible for performing operations on resources. It uses the resource agent scripts to carry out the work. The LRM is “dumb” in that it does not know of any policy by itself. It needs the DC to tell it what to do.

LSB resource agent

LSB resource agents are standard LSB init scripts. LSB init scripts are not limited to use in a high availability context. Any LSB-compliant Linux system uses LSB init scripts to control services. Any LSB resource agent supports a `start`, `stop`, `restart`, `status` and `force-reload` option and may optionally provide `try-restart` and `reload` as well. LSB resource agents are located in `/etc/init.d`. Find more information about LSB resource agents and the actual specification at `http://www.linux-ha.org/LSBResourceAgent` and `http://www.linux-foundation.org/spec/refspecs/LSB_3.0.0/LSB-Core-generic/LSB-Core-generic/iniscrptact.html` (see also OCF resource agent (page 397) and Heartbeat resource agent (page 396)).

node

Any computer (real or virtual) that is a member of a cluster and invisible to the user.

pingd

The ping daemon. It continuously contacts one or more servers outside the cluster with ICMP pings.

policy engine (PE)

The policy engine computes the actions that need to be taken to implement policy changes in the CIB. This information is then passed on to the transaction engine, which in turn implements the policy changes in the cluster setup. The PE always runs on the DC.

OCF resource agent

OCF resource agents are similar to LSB resource agents (init scripts). Any OCF resource agent must support `start`, `stop`, and `status` (sometimes called `monitor`) options. Additionally, they support a `metadata` option that returns the description of the resource agent type in XML. Additional options may be supported, but are not mandatory. OCF resource agents reside in `/usr/lib/ocf/resource.d/`*`provider`*. Find more information about OCF resource agents and a draft of the specification at `http://www.linux-ha.org/OCFResourceAgent` and `http://www.opencf.org/cgi-bin/viewcvs.cgi/specs/ra/resource-agent-api.txt?rev=HEAD` (see also Heartbeat resource agent (page 396)).

quorum
: In a cluster, a cluster partition is defined to have quorum (is “quorate”) if it has the majority of nodes (or votes). Quorum distinguishes exactly one partition. It is part of the algorithm to prevent several disconnected partitions or nodes from proceeding and causing data and service corruption (split brain). Quorum is a prerequisite for fencing, which then ensures that quorum is indeed unique.

resource
: Any type of service or application that is known to Heartbeat. Examples include an IP address, a file system, or a database.

resource agent (RA)
: A resource agent (RA) is a script acting as a proxy to manage a resource. There are three different kinds of resource agents: OCF (Open Cluster Framework) resource agents, LSB resource agents (Standard LSB init scripts), and Heartbeat resource agents (Heartbeat v1 resources).

Single Point of Failure (SPOF)
: A single point of failure (SPOF) is any component of a cluster that, should it fail, triggers the failure of the entire cluster.

split brain
: A scenario in which the cluster nodes are divided into two or more groups that do not know of each other (either through a software or hardware failure). STONITH prevents a split brain situation from badly affecting the entire cluster. Also known as a “partitioned cluster” scenario.

  The term split brain is also used in DRBD but means, the two nodes contain different data.

STONITH
: The acronym for “Shoot the other node in the head” which is basically bringing down a misbehaving node to prevent it from causing trouble in the cluster.

transition engine (TE)
: The transition engine (TE) takes the policy directives from the PE and carries them out. The TE always runs on the DC. From there, it instructs the local resource managers on the others nodes of which actions to take.