

SUSE Linux Enterprise High Availability Extension

www.suse.com

September 27, 2012

High Availability Guide



High Availability Guide

List of Authors: Tanja Roth, Thomas Schraitle

Copyright © 2006–2012 Novell, Inc. and contributors. All rights reserved.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or (at your option) version 1.3; with the Invariant Section being this copyright notice and license. A copy of the license version 1.2 is included in the section entitled “GNU Free Documentation License”.

For Novell trademarks, see the Novell Trademark and Service Mark list <http://www.novell.com/company/legal/trademarks/tmlist.html>. All other third party trademarks are the property of their respective owners. A trademark symbol (®, ™ etc.) denotes a Novell trademark; an asterisk (*) denotes a third party trademark.

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 | xiii |
| 1 Feedback | xiv |
| 2 Documentation Conventions | xv |
| 3 About the Making of This Manual | xv |
| | |
| I Installation and Setup | 1 |
| 1 Product Overview | 3 |
| 1.1 Key Features | 3 |
| 1.2 Benefits | 7 |
| 1.3 Cluster Configurations: Storage | 11 |
| 1.4 Architecture | 13 |
| | |
| 2 System Requirements | 19 |
| 2.1 Hardware Requirements | 19 |
| 2.2 Software Requirements | 20 |
| 2.3 Shared Disk System Requirements | 20 |
| 2.4 Other Requirements | 21 |
| | |
| 3 Installation and Basic Setup | 23 |
| 3.1 Definition of Terms | 23 |
| 3.2 Overview | 26 |
| 3.3 Installation as Add-on | 27 |

| | |
|---------------------------------------------------------------------|------------|
| 3.4 Automatic Cluster Setup (sleha-bootstrap) | 28 |
| 3.5 Manual Cluster Setup (YaST) | 31 |
| 3.6 Mass Deployment with AutoYaST | 44 |
| II Configuration and Administration | 47 |
| 4 Configuration and Administration Basics | 49 |
| 4.1 Global Cluster Options | 49 |
| 4.2 Cluster Resources | 51 |
| 4.3 Resource Monitoring | 65 |
| 4.4 Resource Constraints | 66 |
| 4.5 For More Information | 74 |
| 5 Configuring and Managing Cluster Resources (GUI) | 75 |
| 5.1 Pacemaker GUI—Overview | 76 |
| 5.2 Configuring Global Cluster Options | 79 |
| 5.3 Configuring Cluster Resources | 80 |
| 5.4 Managing Cluster Resources | 102 |
| 6 Configuring and Managing Cluster Resources (Web Interface) | 109 |
| 6.1 Hawk—Overview | 110 |
| 6.2 Configuring Global Cluster Options | 113 |
| 6.3 Configuring Cluster Resources | 115 |
| 6.4 Managing Cluster Resources | 134 |
| 6.5 Multi-Site Clusters (Geo Clusters) | 144 |
| 6.6 Troubleshooting | 149 |

7 Configuring and Managing Cluster Resources (Command Line) **151**

| | |
|-----------------------------------------------------------------|-----|
| 7.1 <code>crm</code> Shell—Overview | 152 |
| 7.2 Configuring Global Cluster Options | 158 |
| 7.3 Configuring Cluster Resources | 159 |
| 7.4 Managing Cluster Resources | 170 |
| 7.5 Setting Passwords Independent of <code>cib.xml</code> | 172 |
| 7.6 Retrieving History Information | 173 |
| 7.7 For More Information | 174 |

8 Adding or Modifying Resource Agents **175**

| | |
|-------------------------------------------------|-----|
| 8.1 STONITH Agents | 175 |
| 8.2 Writing OCF Resource Agents | 176 |
| 8.3 OCF Return Codes and Failure Recovery | 177 |

9 Fencing and STONITH **181**

| | |
|--------------------------------------|-----|
| 9.1 Classes of Fencing | 181 |
| 9.2 Node Level Fencing | 182 |
| 9.3 STONITH Configuration | 184 |
| 9.4 Monitoring Fencing Devices | 188 |
| 9.5 Special Fencing Devices | 189 |
| 9.6 Basic Recommendations | 190 |
| 9.7 For More Information | 191 |

10 Access Control Lists **193**

| | |
|-------------------------------------------------------------|-----|
| 10.1 Requirements and Prerequisites | 193 |
| 10.2 The Basics of ACLs | 194 |
| 10.3 Configuring ACLs with the Pacemaker GUI | 197 |
| 10.4 Configuring ACLs with the <code>crm</code> Shell | 198 |

| | |
|--------------------------------------------------------------|------------|
| 10.5 For More Information | 200 |
| 11 Network Device Bonding | 201 |
| 11.1 Configuring Bonding Devices with YaST | 201 |
| 11.2 For More Information | 203 |
| 12 Load Balancing with Linux Virtual Server | 205 |
| 12.1 Conceptual Overview | 205 |
| 12.2 Configuring IP Load Balancing with YaST | 208 |
| 12.3 Further Setup | 213 |
| 12.4 For More Information | 214 |
| 13 Multi-Site Clusters (Geo Clusters) | 215 |
| 13.1 Challenges for Multi-Site Clusters | 215 |
| 13.2 Conceptual Overview | 216 |
| 13.3 Requirements | 218 |
| 13.4 Basic Setup | 219 |
| 13.5 Managing Multi-Site Clusters | 224 |
| III Storage and Data Replication | 227 |
| 14 OCFS2 | 229 |
| 14.1 Features and Benefits | 229 |
| 14.2 OCFS2 Packages and Management Utilities | 230 |
| 14.3 Configuring OCFS2 Services and a STONITH Resource | 231 |
| 14.4 Creating OCFS2 Volumes | 233 |
| 14.5 Mounting OCFS2 Volumes | 236 |
| 14.6 Using Quotas on OCFS2 File Systems | 238 |
| 14.7 For More Information | 238 |

| | |
|-----------------------------------------------------------------------|------------|
| 15 Distributed Replicated Block Device (DRBD) | 241 |
| 15.1 Conceptual Overview | 241 |
| 15.2 Installing DRBD Services | 243 |
| 15.3 Configuring the DRBD Service | 244 |
| 15.4 Testing the DRBD Service | 250 |
| 15.5 Tuning DRBD | 252 |
| 15.6 Troubleshooting DRBD | 252 |
| 15.7 For More Information | 254 |
| 16 Cluster Logical Volume Manager (cLVM) | 257 |
| 16.1 Conceptual Overview | 257 |
| 16.2 Configuration of cLVM | 258 |
| 16.3 Configuring Eligible LVM2 Devices Explicitly | 267 |
| 16.4 For More Information | 268 |
| 17 Storage Protection | 269 |
| 17.1 Storage-based Fencing | 270 |
| 17.2 Ensuring Exclusive Storage Activation | 276 |
| 18 Samba Clustering | 279 |
| 18.1 Conceptual Overview | 279 |
| 18.2 Basic Configuration | 281 |
| 18.3 Debugging and Testing Clustered Samba | 283 |
| 18.4 Joining Active Directory Domains | 285 |
| 18.5 For More Information | 289 |
| 19 Disaster Recovery with ReaR | 291 |
| 19.1 Terminology | 291 |
| 19.2 Conceptual Overview | 292 |
| 19.3 Preparing for the Worst Scenarios: Disaster Recovery Plans | 293 |

| | |
|-----------------------------------------------|-----|
| 19.4 Setting Up ReaR | 294 |
| 19.5 Setting Up rear-SUSE with AutoYaST | 295 |
| 19.6 For More Information | 297 |

IV Troubleshooting and Reference **299**

20 Troubleshooting **301**

| | |
|-----------------------------------------|-----|
| 20.1 Installation and First Steps | 301 |
| 20.2 Logging | 302 |
| 20.3 Resources | 304 |
| 20.4 STONITH and Fencing | 305 |
| 20.5 Miscellaneous | 306 |
| 20.6 Fore More Information | 309 |

21 HA OCF Agents **311**

| | |
|------------------------|-----|
| ocf:anything | 312 |
| ocf:AoEtarget | 314 |
| ocf:apache | 315 |
| ocf:AudibleAlarm | 318 |
| ocf:ClusterMon | 319 |
| ocf:conntrackd | 320 |
| ocf:CTDB | 321 |
| ocf:db2 | 325 |
| ocf:Delay | 327 |
| ocf:drbd | 328 |
| ocf:Dummy | 329 |
| ocf:eDir88 | 330 |
| ocf:ethmonitor | 332 |
| ocf:Evmsd | 334 |

| | |
|----------------------------|-----|
| ocf:EvmsSCC | 335 |
| ocf:exportfs | 336 |
| ocf:Filesystem | 338 |
| ocf:fio | 340 |
| ocf:ICP | 341 |
| ocf:ids | 342 |
| ocf:IPAddr2 | 344 |
| ocf:IPAddr | 347 |
| ocf:IPsrcaddr | 349 |
| ocf:IPv6addr | 350 |
| ocf:iSCSILogicalUnit | 351 |
| ocf:iSCSITarget | 353 |
| ocf:iscsi | 355 |
| ocf:jboss | 356 |
| ocf:ldirectord | 358 |
| ocf:LinuxSCSI | 359 |
| ocf:LVM | 360 |
| ocf:lxc | 361 |
| ocf:MailTo | 362 |
| ocf:ManageRAID | 363 |
| ocf:ManageVE | 364 |
| ocf:mysql-proxy | 365 |
| ocf:mysql | 367 |
| ocf:named | 370 |
| ocf:nfsserver | 372 |
| ocf:nginx | 374 |
| ocf:oracle | 377 |
| ocf:oralsnr | 379 |

| | |
|------------------------|-----|
| ocf:pgsql | 380 |
| ocf:pingd | 382 |
| ocf:porthblock | 384 |
| ocf:postfix | 386 |
| ocf:proftpd | 387 |
| ocf:Pure-FTPd | 389 |
| ocf:Raid1 | 390 |
| ocf:Route | 391 |
| ocf:rsyncd | 393 |
| ocf:rsyslog | 394 |
| ocf:SAPDatabase | 395 |
| ocf:scsi2reservation | 398 |
| ocf:SendArp | 399 |
| ocf:ServeRAID | 400 |
| ocf:sfex | 401 |
| ocf:slapd | 403 |
| ocf:SphinxSearchDaemon | 405 |
| ocf:Squid | 406 |
| ocf:Stateful | 408 |
| ocf:symlink | 409 |
| ocf:SysInfo | 410 |
| ocf:syslog-ng | 411 |
| ocf:tomcat | 412 |
| ocf:VIPArp | 415 |
| ocf:VirtualDomain | 416 |
| ocf:vmware | 418 |
| ocf:WAS6 | 419 |
| ocf:WAS | 420 |

| | |
|---------------------------------------------------------------|------------|
| ocf:WinPopup | 421 |
| ocf:Xen | 422 |
| ocf:Xinetd | 424 |
| V Appendix | 425 |
| A Example of Setting Up a Simple Testing Resource | 427 |
| A.1 Configuring a Resource with the GUI | 427 |
| A.2 Configuring a Resource Manually | 429 |
| B Example Configuration for OCFS2 and cLVM | 431 |
| C Cluster Management Tools | 433 |
| D Upgrading Your Cluster to the Latest Product Version | 437 |
| D.1 Upgrading from SLES 10 to SLE HA 11 | 437 |
| D.2 Upgrading from SLE HA 11 to SLE HA 11 SP1 | 442 |
| D.3 Upgrading from SLE HA 11 SP1 to SLE HA 11 SP2 | 443 |
| E What's New? | 445 |
| E.1 Version 10 SP3 to Version 11 | 445 |
| E.2 Version 11 to Version 11 SP1 | 448 |
| E.3 Version 11 SP1 to Version 11 SP2 | 450 |
| Terminology | 453 |
| F GNU Licenses | 459 |
| F.1 GNU Free Documentation License | 459 |

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 (Hawk), 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.

This 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 cluster resources, using either the graphical user interface (Pacemaker GUI), the Web interface (HA Web Konsole), or the command line interface (`crm` shell). To avoid unauthorized access to the cluster configuration, define roles and assign them to certain users for fine-grained control. Learn how to make use of load balancing and fencing or how to set up a multi-site cluster. 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 a High Availability service from the active node of a cluster to its standby node. Furthermore, a clustered

Samba server also provides a High Availability solution 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 OCF agents shipped with this product.

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.suse.com/documentation/sle_ha.

1 Feedback

Several feedback channels are available:

Bugs and Enhancement Requests

For services and support options available for your product, refer to <http://www.suse.com/support/>.

To report bugs for a product component, log into the Novell Customer Center from <http://www.suse.com/support/> and select *My Support > Service Request*.

User Comments

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

Mail

For feedback on the documentation of this product, you can also send a mail to `doc-team@suse.de`. Make sure to include the document title, the product version and the publication date of the documentation. To report errors or suggest enhancements, provide a concise description of the problem and refer to the respective section number and page (or URL).

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 architectures `amd64`, `em64t`, and `ipf`. 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.

3 About the Making of This Manual

This book is written in Novdoc, a subset of DocBook (see <http://www.docbook.org>). The XML source files were validated by `xmllint`, processed by `xsltproc`,

and converted into XSL-FO using a customized version of Norman Walsh's stylesheets. The final PDF is formatted through XEP from RenderX.

Part I. Installation and Setup

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, fallback, and migration (load balancing) of individually managed cluster resources). The High Availability Extension is available as add-on to SUSE Linux Enterprise Server 11 SP2.

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 453).

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 service availability and resource utilization.
- Local clusters
- Metro clusters (“stretched” local clusters)
- Multi-site clusters (geographically dispersed clusters)

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 SP2 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 Support of Local, Metro, and Multi-Site Clusters

SUSE Linux Enterprise High Availability Extension has been extended to support different geographical scenarios. Support for multi-site clusters is available as a separate option to SUSE Linux Enterprise High Availability Extension.

Local Clusters

A single cluster in one location (for example, all nodes are located in one data center). The cluster uses multicast or unicast for communication between the nodes and manages failover internally. Network latency can be neglected. Storage is typically accessed synchronously by all nodes.

Metro Clusters

A single cluster that can stretch over multiple buildings or data centers, with all sites connected by fibre channel. The cluster uses multicast or unicast for communication between the nodes and manages failover internally. Network latency is usually low (<5 ms for distances of approximately 20 miles). Storage is frequently replicated (mirroring or synchronous replication).

Multi-Site Clusters (Geo Clusters) (page 215)

Multiple, geographically dispersed sites with a local cluster each. The sites communicate via IP. Failover across the sites is coordinated by a higher-level entity.

Multi-site clusters have to cope with limited network bandwidth and high latency. Storage is replicated asynchronously.

The greater the geographical distance between individual cluster nodes, the more factors may potentially disturb the high availability of services the cluster provides. Network latency, limited bandwidth and access to storage are the main challenges for long-distance clusters.

1.1.6 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 21, *HA OCF Agents* (page 311).

1.1.7 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 3.3, “Installation as Add-on” (page 27). YaST also provides the following modules in the High Availability category to help configure your cluster or individual components:

- Cluster: Basic cluster setup. For details, refer to Section 3.5, “Manual Cluster Setup (YaST)” (page 31).
- DRBD: Configuration of a Distributed Replicated Block Device.

- IP Load Balancing: Configuration of load balancing with Linux Virtual Server. For details, refer to Chapter 12, *Load Balancing with Linux Virtual Server* (page 205).

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 75).

HA Web Konsole (Hawk)

A Web-based user interface with which you can administer your Linux cluster from non-Linux machines. It is also an ideal solution in case your system does not provide a graphical user interface. It 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 6, *Configuring and Managing Cluster Resources (Web Interface)* (page 109).

crm Shell

A powerful unified command line interface to configure resources and execute all monitoring or administration tasks. For details, refer to Chapter 7, *Configuring and Managing Cluster Resources (Command Line)* (page 151).

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

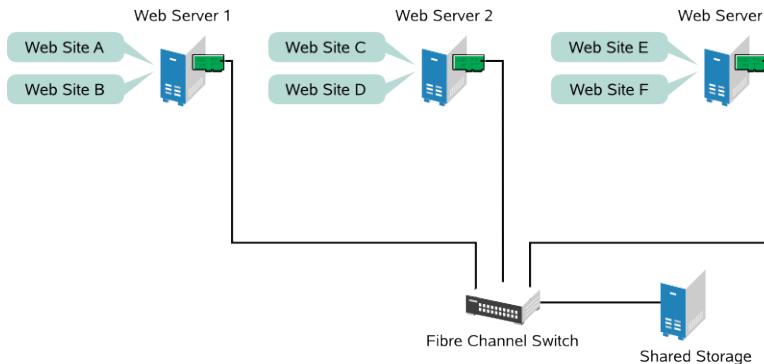
Shared disk fault tolerance can be obtained by implementing RAID on the shared disk subsystem.

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.

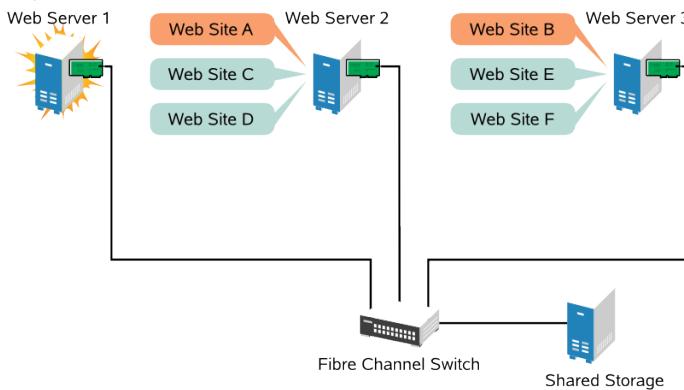
Figure 1.1: Three-Server Cluster



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

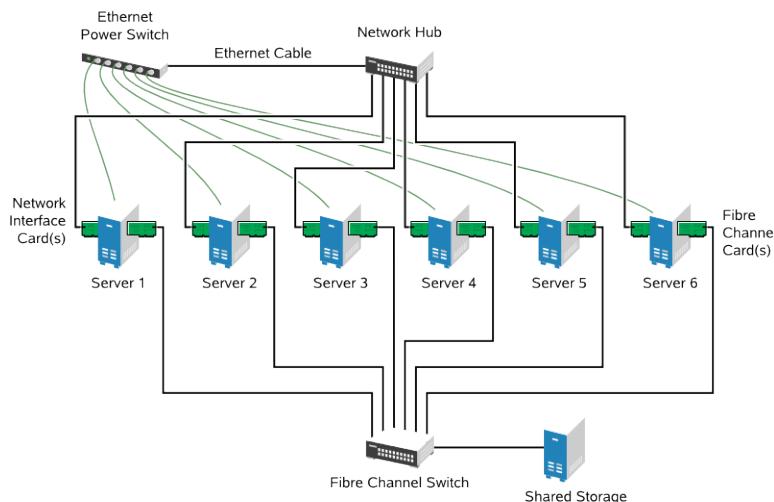
Cluster configurations with the High Availability Extension might or might not include a shared disk subsystem. The shared disk subsystem can be connected via high-speed Fibre Channel cards, cables, and switches, or it can be configured to use iSCSI. If a server fails, another designated server in the 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.

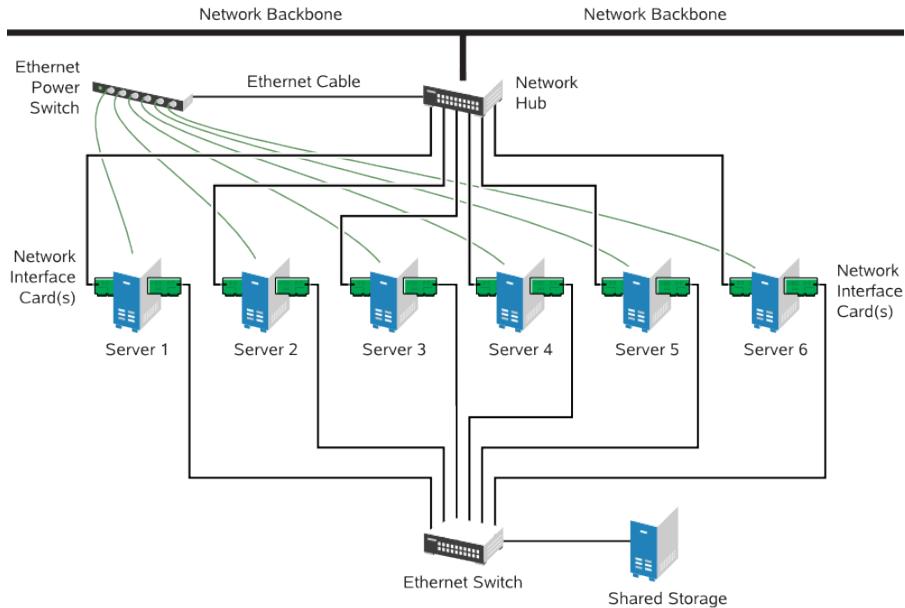
Typical resources might include data, applications, and services. The following figure shows how a typical Fibre Channel cluster configuration might look.

Figure 1.3: Typical Fibre Channel Cluster Configuration



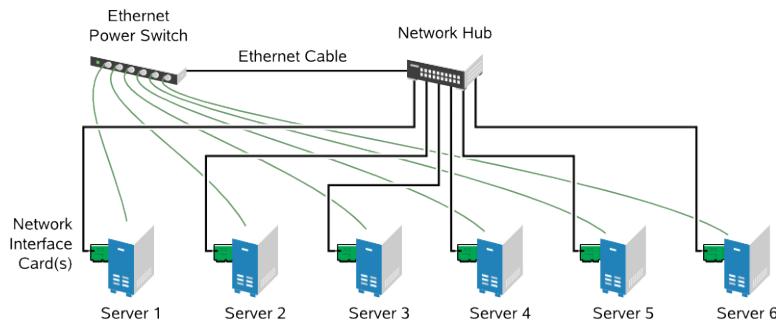
Although Fibre Channel provides the best performance, you can also configure your cluster to use iSCSI. iSCSI is an alternative to Fibre Channel that can be used to create a low-cost Storage Area Network (SAN). The following figure shows how a typical iSCSI cluster configuration might look.

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



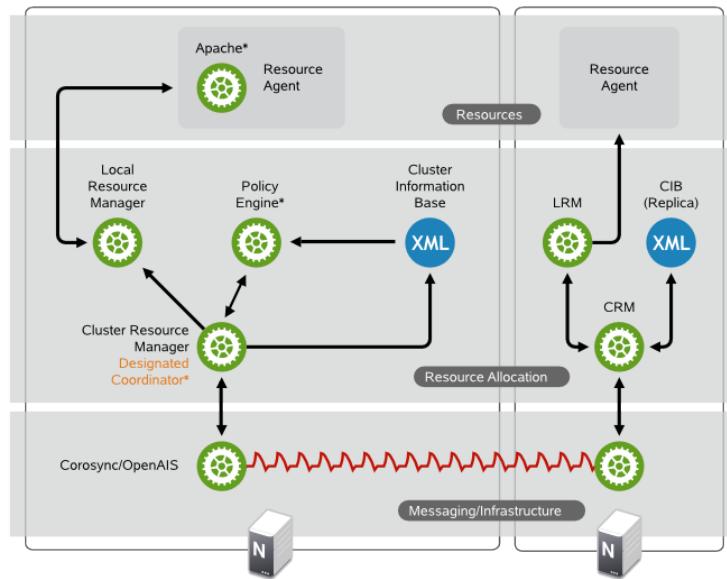
1.4 Architecture

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 14) illustrates the different layers and their associated components.

Figure 1.6: Architecture



1.4.1.1 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.

1.4.1.2 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)”.

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 “Designated Coordinator (DC)”. All other nodes contain a CIB replica.

Designated Coordinator (DC)

One CRM in the cluster is elected as DC. 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. The DC is also the node 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.

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 always runs on the DC.

Local Resource Manager (LRM)

The LRM calls the local Resource Agents (see Section 1.4.1.3, “Resource Layer” (page 15)) 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.

1.4.1.3 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 its LRM (implemented as `lrmd`) to perform resource modifications. The `lrmd` is not cluster-aware and interacts directly with resource agents (scripts).

All peer nodes 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.

System Requirements

The following section informs you about system requirements and some prerequisites for SUSE® Linux Enterprise High Availability Extension, including hardware, software, shared disk system, and other essentials.

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 20). The servers do not require identical hardware (memory, disk space, etc.), but they must have the same architecture. Cross-platform clusters are not supported.
- At least two TCP/IP communication media. Cluster nodes use multicast or unicast for communication so the network equipment must support the communication means you want to use. 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.

IMPORTANT: No Support Without STONITH

A cluster without STONITH is not supported.

For more information, refer to Chapter 9, *Fencing and STONITH* (page 181).

2.2 Software Requirements

Ensure that the following software requirements are met:

- SUSE® Linux Enterprise Server 11 SP2 with all available online updates installed on all nodes that will be part of the cluster.
- SUSE Linux Enterprise High Availability Extension 11 SP2 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 device provided by DRBD, and never the backing device. Use the same (bonded) NICs that the rest of the cluster uses to leverage the redundancy provided there.

2.4 Other Requirements

Time Synchronization

Cluster nodes should synchronize to an NTP server outside the cluster. For more information, see the *SUSE Linux Enterprise Server Administration Guide*, available at <http://www.suse.com/documentation/sles11>. Refer to the chapter *Time Synchronization with NTP*.

If nodes are not synchronized, log files or cluster reports are very hard to analyze.

NIC Names

Must be identical on all nodes.

Hostname and IP Address

Configure hostname resolution by editing the `/etc/hosts` file on *each* server in the cluster. To ensure that cluster communication is not slowed down or tampered with by any DNS:

- Use static IP addresses.
- List all cluster nodes in this file with their fully qualified hostname and short hostname. 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.

For more information, see the *SUSE Linux Enterprise Server Administration Guide*, available at <http://www.suse.com/documentation>. Refer to chapter *Basic Networking > Configuring Hostname and DNS*.

SSH

All cluster nodes must be able to access each other via SSH. Tools like `hb_report` (for troubleshooting) and the history explorer require passwordless SSH access between the nodes, otherwise they can only collect data from the current node.

NOTE: Regulatory Requirements

If passwordless SSH access does not comply with regulatory requirements, you can use the following work-around for `hb_report`:

Create a user that can log in without a password (for example, using public key authentication). Configure `sudo` for this user so it does not require a root password. Start `hb_report` from command line with the `-u` option to specify the user's name. For more information, see the `hb_report` man page.

For the history explorer there is currently no alternative for passwordless login.

Installation and Basic Setup

This chapter describes how to install and set up SUSE® Linux Enterprise High Availability Extension 11 SP2 from scratch. Choose between an automatic setup which allows you to have a cluster up and running within a few minutes (with the choice to adjust any options later on) or decide for a manual setup, allowing you to set your individual options right at the beginning.

Refer to chapter Appendix D, *Upgrading Your Cluster to the Latest Product Version* (page 437) 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 Definition of Terms

Existing Cluster

The term “existing cluster” is used to refer to any cluster which consists of at least one node. Existing clusters have a basic Corosync configuration that defines the communication channels, but they do not necessarily have resource configuration yet.

Multicast

A technology used for a one-to-many communication within a network that can be used for cluster communication. If multicast does not comply with your corporate IT policy, use unicast instead.

NOTE: Switches and Multicast

If you want to use multicast for cluster communication, make sure your switches support multicast.

Multicast Address (`mcastaddr`)

IP address to use for multicasting (either IPv4 or IPv6). You can use any multicast address in your private network.

Multicast Port (`mcastport`)

The port to use for cluster communication.

Unicast

A technology for sending messages to a single network destination. In Corosync, unicast is implemented as UDP-unicast (UDPU).

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.

NOTE: Network Address for All Nodes

As the same Corosync configuration will be used on all nodes, make sure to use a network address as `bindnetaddr`, not the address of a specific network interface.

Redundant Ring Protocol (RRP)

Corosync supports the Totem Redundant Ring Protocol. It allows the use of multiple redundant local-area networks for resilience against partial or total network faults. This way, cluster communication can still be kept up as long as a single network is operational. For more information, refer to <http://www.rcsc.de/pdf/icdcs02.pdf>.

When having defined redundant communication channels in Corosync, use RRP to tell the cluster how to use these interfaces. RRP can have three modes (`rrp_mode`): if set to `active`, Corosync uses both interfaces actively. If set to `passive`, Corosync sends messages alternatively over the available networks. If `rrp_mode` is set to `none`, RRP is disabled.

Csync2

A synchronization tool that can be used to replicate configuration files 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/exclude 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>

conntrack Tools

To synchronize the connection status between cluster nodes, the High Availability Extension uses the `conntrack-tools`. They allow to interact with the in-kernel Connection Tracking System for enabling *stateful* packet inspection for iptables. For detailed information, refer to <http://conntrack-tools.netfilter.org/>.

AutoYaST

AutoYaST is a system for installing one or more SUSE Linux Enterprise systems automatically and without user intervention. On SUSE Linux Enterprise you can create 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 ready-to-use system in the end. This profile can then be used for mass deployment in different ways (for example, to clone existing cluster nodes).

For detailed instructions on how to use AutoYaST in various scenarios, see the SUSE Linux Enterprise 11 SP2 *Deployment Guide*, available at <http://www.suse.com/documentation>. Refer to chapter *Automated Installation*.

3.2 Overview

The following basic steps are needed for installation and initial cluster setup.

1 Installation as Add-on (page 27):

Install the software packages with YaST. Alternatively, you can install them from the command line with `zypper`:

```
zypper in -t pattern sles_ha
```

2 Initial Cluster Setup:

After installing the software on all nodes that will be part of your cluster, the following steps are needed to initially configure the cluster.

2a Defining the Communication Channels (page 32)

2b Optional: Defining Authentication Settings (page 37)

2c Configuring Services (page 38)

2d Transferring the Configuration to All Nodes (page 39). Whereas the configuration of Csync2 is done on one node only, the services Csync2 and `xinetd` need to be started on all nodes.

2e Optional: Synchronizing Connection Status Between Cluster Nodes (page 42)

2f Bringing the Cluster Online (page 43). The OpenAIS/Corosync service needs to be started on all nodes.

The cluster setup steps can either be executed automatically (with bootstrap scripts) or manually (with the YaST cluster module or from command line).

- If you decide for an automatic cluster setup, refer to Section 3.4, “Automatic Cluster Setup (`sleha-bootstrap`)” (page 28).
- For a manual setup (or for adjusting any options after the automatic setup), refer to Section 3.5, “Manual Cluster Setup (YaST)” (page 31).

You can also use a combination of both setup methods, for example: set up one node with YaST cluster and then use `sleha-join` to integrate more nodes.

Existing nodes can also be cloned for mass deployment with AutoYaST. The cloned nodes will have the same packages installed and the same system configuration. For details, refer to Section 3.6, “Mass Deployment with AutoYaST” (page 44).

3.3 Installation as Add-on

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 to SUSE Linux Enterprise Server. For information on how to install add-on products, see the SUSE Linux Enterprise 11 SP2 *Deployment Guide*, available at <http://www.suse.com/documentation/sles11>. Refer to chapter *Installing Add-On Products*.

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 installing the packages.

NOTE

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

- 4 Install the High Availability pattern on *all* machines that will be part of your cluster.

If you do not want to install SUSE® Linux Enterprise Server 11 SP2 and SUSE Linux Enterprise High Availability Extension 11 SP2 manually on all nodes that

will be part of your cluster, use AutoYaST to clone existing nodes. For more information, refer to Section 3.6, “Mass Deployment with AutoYaST” (page 44).

3.4 Automatic Cluster Setup (`sleha-bootstrap`)

The `sleha-bootstrap` package provides everything you need to get a one-node cluster up and running and to make other nodes join with the following basic steps:

Automatically Setting Up the First Node (page 29)

With `sleha-init`, define the basic parameters needed for cluster communication and (optionally) set up a STONITH mechanism to protect your shared storage. This leaves you with a running one-node cluster.

Adding Nodes to an Existing Cluster (page 30)

With `sleha-join`, add more nodes to your cluster.

Both commands execute bootstrap scripts that require only a minimum of time and manual intervention. Any options set during the bootstrap process can be modified later with the YaST cluster module.

Before starting the automatic setup, make sure that the following prerequisites are fulfilled on all nodes that will participate in the cluster:

Prerequisites

- The requirements listed in Section 2.2, “Software Requirements” (page 20) and Section 2.4, “Other Requirements” (page 21) are fulfilled.
- The `sleha-bootstrap` package is installed.
- The network is configured according to your needs. For example, a private network is available for cluster communication and network device bonding is configured. For information on bonding, refer to Chapter 11, *Network Device Bonding* (page 201).
- If you want to use SBD for your shared storage, you need one shared block device for SBD. The block device need not be formatted. For more information, refer to Chapter 17, *Storage Protection* (page 269).

- All nodes must be able to see the shared storage via the same paths (`/dev/disk/by-path/...` or `/dev/disk/by-id/...`).

Procedure 3.2: Automatically Setting Up the First Node

The `sleha-init` command checks for configuration of NTP and guides you through configuration of the cluster communication layer (Corosync), and (optionally) through the configuration of SBD to protect your shared storage:

- 1 Log in as `root` to the physical or virtual machine you want to use as cluster node.
- 2 Start the bootstrap script by executing

```
sleha-init
```

If NTP has not been configured to start at boot time, a message appears.

If you decide to continue anyway, the script will automatically generate keys for SSH access and for the Csync2 synchronization tool and start the services needed for both.

- 3 To configure the cluster communication layer (Corosync):
 - 3a Enter a network address to bind to. By default, the script will propose the network address of `eth0`. Alternatively, enter a different network address, for example the address of `bond0`.
 - 3b Enter a multicast address. The script proposes a random address that you can use as default.
 - 3c Enter a multicast port. The script proposes `5405` as default.
 - 4 To configure SBD (optional), enter a persistent path to the partition of your block device that you want to use for SBD. The path must be consistent across all nodes in the cluster.
- Finally, the script will start the OpenAIS service to bring the one-node cluster online and enable the Web management interface Hawk. The URL to use for Hawk is displayed on the screen.
- 5 For any details of the setup process, check `/var/log/sleha-bootstrap.log`.

You now have a running one-node cluster. If you want to check the cluster status or start managing resources, proceed by logging in to one of the user interfaces, Hawk or the Pacemaker GUI. For more information, refer to Chapter 6, *Configuring and Managing Cluster Resources (Web Interface)* (page 109) and Chapter 5, *Configuring and Managing Cluster Resources (GUI)* (page 75).

IMPORTANT: Secure Password

The bootstrap procedure creates a linux user named `hacluster` with the password `linux`. You need it for logging in to the Pacemaker GUI or Hawk. Replace the default password with a secure one as soon as possible:

```
passwd hacluster
```

Procedure 3.3: Adding Nodes to an Existing Cluster

If you have a cluster up and running (with one or more nodes), add more cluster nodes with the `sleha-join` bootstrap script. The script only needs access to an existing cluster node and will complete the basic setup on the current machine automatically.

If you have configured the existing cluster nodes with the YaST cluster module, make sure the following prerequisites are fulfilled before you run `sleha-join`:

- The `root` user on the existing nodes has SSH keys in place for passwordless login.
- Csync2 is configured on the existing nodes. For details, refer to Procedure 3.8, “Configuring Csync2 with YaST” (page 39).

If you are logged in to the first node via Hawk, you can follow the changes in cluster status and view the resources being activated in the Web interface.

1 Log in as `root` to the physical or virtual machine supposed to join the cluster.

2 Start the bootstrap script by executing:

```
sleha-join
```

If NTP has not been configured to start at boot time, a message appears.

3 If you decide to continue anyway, you will be prompted for the IP address of an existing node. Enter the IP address.

- 4 If you have not already configured a passwordless SSH access between both machines, you will also be prompted for the `root` password of the existing node.

After logging in to the specified node, the script will copy the Corosync configuration, configure SSH and Csync2, and will bring the current machine online as new cluster node. Apart from that, it will start the service needed for Hawk. If you have configured shared storage with OCFS2, it will also automatically create the mountpoint directory for the OCFS2 file system.

- 5 Repeat the steps above for all machines you want to add to the cluster.
- 6 For details of the process, check `/var/log/sleha-bootstrap.log`.

IMPORTANT: Check no-quorum-policy

After adding all nodes, check if you need to adjust the `no-quorum-policy` in the global cluster options. This is especially important for two-node clusters. For more information, refer to Section 4.1.1, “Option `no-quorum-policy`” (page 50).

3.5 Manual Cluster Setup (YaST)

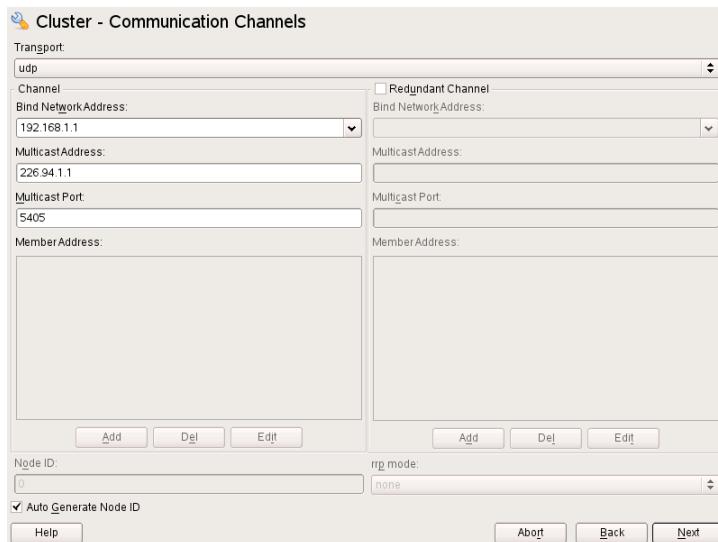
See Section 3.2, “Overview” (page 26) for an overview of all steps for initial setup.

3.5.1 YaST Cluster Module

The following sections guide you through each of the setup steps, using the YaST cluster module. To access it, start YaST as `root` and select *High Availability > Cluster*. Alternatively, start the module from 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.

Figure 3.1: YaST Cluster Module—Overview



The YaST cluster module automatically opens the ports in the firewall that are needed for cluster communication on the current machine. The configuration is written to `/etc/sysconfig/SuSEfirewall2.d/services/cluster`.

Note that some options in the YaST cluster module apply only to the current node, whereas others may automatically be transferred to all nodes. Find detailed information about this in the following sections.

3.5.2 Defining the Communication Channels

For successful communication between the cluster nodes, define at least one communication channel.

IMPORTANT: Redundant Communication Paths

However, it is highly recommended to set up cluster communication via two or more redundant paths. This can be done via:

- *Network Device Bonding* (page 201).

- A second communication channel in Corosync. For details, see Procedure 3.5, “Defining a Redundant Communication Channel” (page 35).

If possible, choose network device bonding.

Procedure 3.4: Defining the First Communication Channel

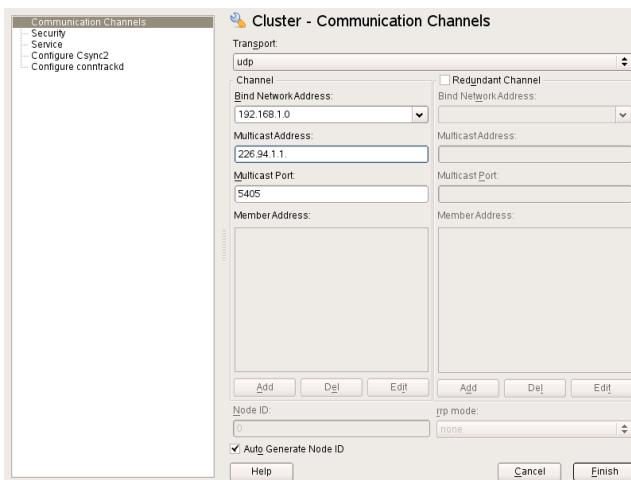
For communication between the cluster nodes, use either multicast (UDP) or unicast (UDPU).

- 1 In the YaST cluster module, switch to the *Communication Channels* category.
- 2 To use multicast:

- 2a Set the *Transport* protocol to UDP.
- 2b Define the *Bind Network Address*. Set the value to the subnet you will use for cluster multicast.
- 2c Define the *Multicast Address*.
- 2d Define the *Multicast Port*.

With the values entered above, you have now defined *one* communication channel for the cluster. In multicast mode, the same `bindnetaddr`, `mcastaddr`, and `mcastport` will be used for all cluster nodes. All nodes in the cluster will know each other by using the same multicast address. For different clusters, use different multicast addresses.

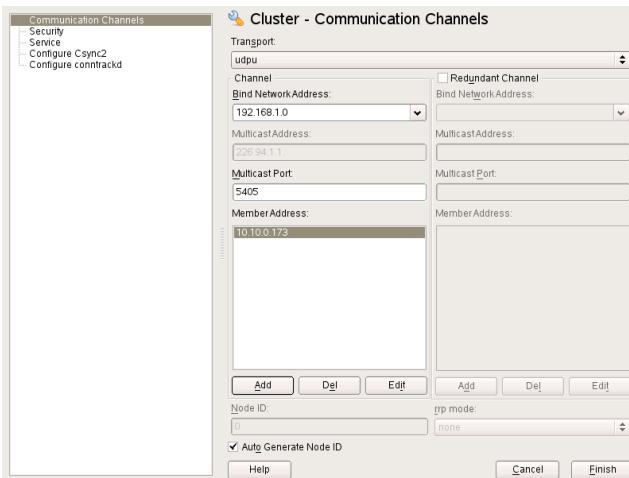
Figure 3.2: YaST Cluster—Multicast Configuration



3 To use unicast:

- 3a** Set the *Transport* protocol to UDP.
- 3b** Define the *Bind Network Address*. Set the value to the subnet you will use for cluster unicast.
- 3c** Define the *Multicast Port*.
- 3d** For unicast communication, Corosync needs to know the IP addresses of all nodes in the cluster. For each node that will be part of the cluster, click *Add* and enter its IP address. To modify or remove any addresses of cluster members, use the *Edit* or *Del* buttons.

Figure 3.3: YaST Cluster—Unicast Configuration



- 4 Activate *Auto Generate Node ID* to automatically generate a unique ID for every cluster node.
- 5 If you modified any options for an existing cluster, confirm your changes and close the cluster module. YaST writes the configuration to `/etc/corosync/corosync.conf`.
- 6 If needed, define a second communication channel as described below. Or click *Next* and proceed with Procedure 3.6, “Enabling Secure Authentication” (page 37).

Procedure 3.5: Defining a Redundant Communication Channel

If network device bonding cannot be used for any reason, the second best choice is to define a redundant communication channel (a second ring) in Corosync. That way, two physically separate networks can be used for communication. In case one network fails, the cluster nodes can still communicate via the other network.

IMPORTANT: Redundant Rings and `/etc/hosts`

If multiple rings are configured, each node can have multiple IP addresses. This needs to be reflected in the `/etc/hosts` file of all nodes.

- 1 In the YaST cluster module, switch to the *Communication Channels* category.
- 2 Activate *Redundant Channel*. The redundant channel must use the same protocol as the first communication channel you defined.
- 3 If you use multicast, define the *Bind Network Address*, the *Multicast Address* and the *Multicast Port* for the redundant channel.

If you use unicast, define the *Bind Network Address*, the *Multicast Port* and enter the IP addresses of all nodes that will be part of the cluster.

Now you have defined an additional communication channel in Corosync that will form a second token-passing ring. In `/etc/corosync/corosync.conf`, the primary ring (the first channel you have configured) gets the ringnumber 0, the second ring (redundant channel) the ringnumber 1.

- 4 To tell Corosync how and when to use the different channels, select the *rrp_mode* you want to use (active or passive). For more information about the modes, refer to “Redundant Ring Protocol (RRP)” (page 24) or click *Help*. As soon as RRP is used, the Stream Control Transmission Protocol (SCTP) is used for communication between the nodes (instead of TCP). The High Availability Extension monitors the status of the current rings and automatically re-enables redundant rings after faults. Alternatively, you can also check the ring status manually with `corosync-cfgtool`. View the available options with `-h`.

If only one communication channel is defined, *rrp_mode* is automatically disabled (value `none`).

- 5 If you modified any options for an existing cluster, confirm your changes and close the cluster module. YaST writes the configuration to `/etc/corosync/corosync.conf`.
- 6 For further cluster configuration, click *Next* and proceed with Section 3.5.3, “Defining Authentication Settings” (page 37).

Find an example file for a UDP setup in `/etc/corosync/corosync.conf.example`. An example for UDPU setup is available in `/etc/corosync/corosync.conf.udpu`.

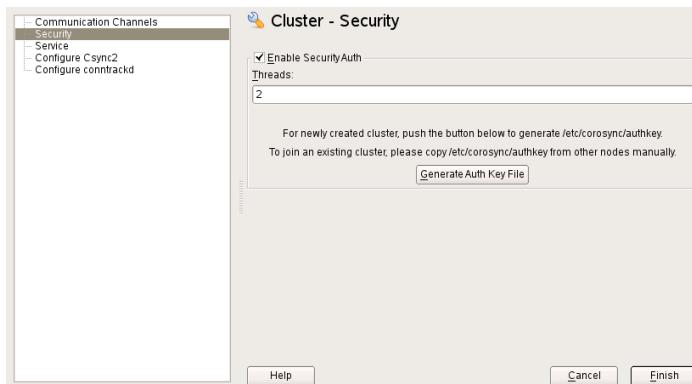
3.5.3 Defining Authentication Settings

The next step is to define the authentication settings for the cluster. You can use HMAC/SHA1 authentication that 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 3.6: *Enabling Secure Authentication*

- 1 In the YaST cluster module, switch to the *Security* category.
- 2 Activate *Enable Security Auth.*
- 3 For a newly created cluster, click *Generate Auth Key File*. An authentication key is created and written to `/etc/corosync/authkey`.

Figure 3.4: YaST Cluster—Security



If you want the current machine to join an existing cluster, do not generate a new key file. Instead, copy the `/etc/corosync/authkey` from one of the nodes to the current machine (either manually or with Csync2).

- 4 If you modified any options for an existing cluster, confirm your changes and close the cluster module. YaST writes the configuration to `/etc/corosync/corosync.conf`.
- 5 For further cluster configuration, proceed with Section 3.5.4, “Configuring Services” (page 38).

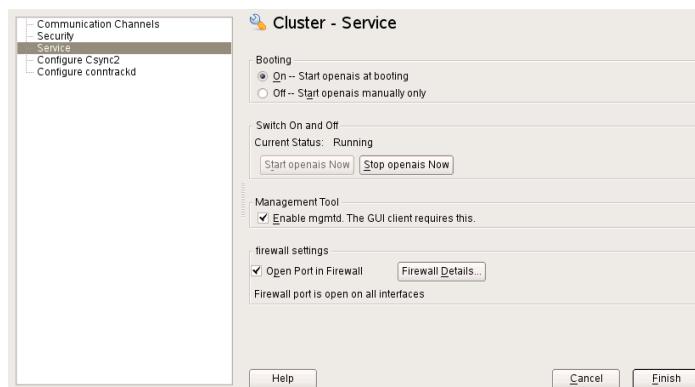
3.5.4 Configuring Services

In the YaST cluster module define whether to start certain services on a node at boot time. You can also use the module to start and stop the services manually. To bring the cluster nodes online and start the cluster resource manager, OpenAIS must be running as a service.

Procedure 3.7: Enabling OpenAIS and mgmtd

- 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. If you select *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 If you want to use the Pacemaker GUI for configuring, managing and monitoring cluster resources, activate *Enable mgmtd*. This daemon is needed for the GUI.
- 4 To start or stop OpenAIS immediately, click the respective button.
- 5 If you modified any options for an existing cluster node, confirm your changes and close the cluster module. Note that the configuration only applies to the current machine, not to all cluster nodes.

Figure 3.5: YaST Cluster—Services



- 6** For further cluster configuration, click *Next* and proceed with Section 3.5.5, “Transferring the Configuration to All Nodes” (page 39).

3.5.5 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.

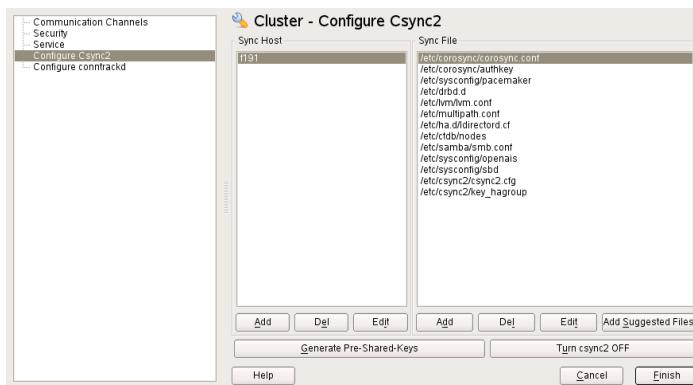
This requires the following basic steps:

- 1** Configuring Csync2 with YaST (page 39).
- 2** Synchronizing the Configuration Files with Csync2 (page 40).

Procedure 3.8: 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*.

Figure 3.6: YaST Cluster—Csync2



- 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 execute `chkconfig csync2` to start Csync2 automatically at boot time.
- 7 If you modified any options for an existing cluster, confirm your changes and close the cluster module. YaST then writes the Csync2 configuration to `/etc/csync2/csync2.cfg`. To start the synchronization process now, proceed with Procedure 3.9, “Synchronizing the Configuration Files with Csync2” (page 40).
- 8 For further cluster configuration, click *Next* and proceed with Section 3.5.6, “Synchronizing Connection Status Between Cluster Nodes” (page 42).

Procedure 3.9: 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 in Procedure 3.8, “Configuring Csync2 with YaST” (page 39). 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 39) to *all* nodes in the cluster as it is needed for authentication by

Csync2. However, do not regenerate the file on the other nodes as it needs to be the same file on all nodes.

- Both Csync2 and `xinetd` must be running on *all* nodes.

NOTE: Starting Services at Boot Time

Execute the following commands on all nodes to make both services start automatically at boot time and to start `xinetd` now:

```
chkconfig csync2 on  
chkconfig xinetd on  
rcxinetd start
```

- 1 On the node that you want to copy the configuration *from*, execute the following command:

```
csync2 -xv
```

This will synchronize all the files once by pushing them to the other nodes. If all files are synchronized successfully, Csync2 will finish with no errors.

If one or several files that are to be synchronized have been modified 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

Csync2 only pushes changes. It does *not* continuously synchronize files between the nodes.

Each time you update files that need to be synchronized, you have 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 with unchanged files, nothing will happen.

3.5.6 Synchronizing Connection Status Between Cluster Nodes

To enable *stateful* packet inspection for iptables, configure and use the conntrack tools with the following basic steps:

- 1 Configuring the conntrackd with YaST (page 42).
- 2 Configuring a resource for conntrackd (class: ocf, provider: heartbeat). If you use Hawk to add the resource, use the default values proposed by Hawk.

After configuring the conntrack tools, you can use them for load balancing with Linux Virtual Server.

Procedure 3.10: Configuring the conntrackd with YaST

Use the YaST cluster module to configure the user-space conntrackd. It needs a dedicated network interface that is not used for other communication channels. The daemon can be started via a resource agent afterward.

- 1 In the YaST cluster module, switch to the *Configure conntrackd* category.
- 2 Select a *Dedicated Interface* for synchronizing the connection status. The IPv4 address of the selected interface is automatically detected and shown in YaST. It must already be configured and it must support multicast.
- 3 Define the *Multicast Address* to be used for synchronizing the connection status.
- 4 In *Group Number*, define a numeric ID for the group to synchronize the connection status to.
- 5 Click *Generate /etc/conntrackd/conntrackd.conf* to create the configuration file for conntrackd.

- 6** Confirm your changes and close the cluster module. If you have done the initial cluster setup exclusively with the YaST cluster module, you have now completed the basic configuration steps. Proceed with Section 3.5.7, “Bringing the Cluster Online” (page 43).

Figure 3.7: YaST Cluster—conntrackd



3.5.7 Bringing the Cluster Online

After the initial cluster configuration is done, start the OpenAIS/Corosync service on *each* cluster node to bring the stack online:

Procedure 3.11: Starting OpenAIS/Corosync and Checking the Status

1 Log in to an existing node.

2 Check if the service is already running:

```
rcopenais status
```

If not, start OpenAIS/Corosync now:

```
rcopenais start
```

3 Repeat the steps above for each of the cluster nodes.

4 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: Thu Nov 10 13:37:48 2011
Last change: Wed Nov  9 17:40:06 2011 by hacluster via cibadmin on auckland
Stack: openais
Current DC: auckland - partition with quorum
Version: 1.1.6-2d8fad5daab7e592069215f144cc765def450309
2 Nodes configured, 2 expected votes
8 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 start to configure cluster resources, using one of the cluster management tools like the `crm` shell, the Pacemaker GUI, or the HA Web Konsole. For more information, refer to the following chapters.

3.6 Mass Deployment 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.

Procedure 3.12: Cloning a Cluster Node with AutoYaST

IMPORTANT: Identical Hardware

This scenario assumes you are rolling out SUSE Linux Enterprise High Availability Extension 11 SP2 to a set of machines with exactly the same hardware 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 SP2 *Deployment Guide*, available at <http://www.suse.com/documentation>.

- 1 Make sure the node you want to clone is correctly installed and configured. For details, refer to Section 3.3, “Installation as Add-on” (page 27), and Section 3.4,

“Automatic Cluster Setup (sleha-bootstrap)” (page 28) or Section 3.5, “Manual Cluster Setup (YaST)” (page 31), respectively.

- 2 Follow the description outlined in the SUSE Linux Enterprise 11 SP2 *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 based on 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 3.13: 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 3.5.5, “Transferring the Configuration to All Nodes” (page 39).
- 2 To bring the node online, start the OpenAIS service on the cloned node as described in Section 3.5.7, “Bringing the Cluster Online” (page 43).

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

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 50)
- Option `stonith-enabled` (page 51)

Learn how to adjust those parameters with the cluster management tools of your choice:

- Hawk: Procedure 6.2, “Modifying Global Cluster Options” (page 114)
- Pacemaker GUI: Procedure 5.1, “Modifying Global Cluster Options” (page 79)
- crm shell: Section 7.2, “Configuring Global Cluster Options” (page 158)

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`.

IMPORTANT: No Support Without STONITH

A cluster without STONITH enabled is not supported.

For an overview of all global cluster options and their default values, see *Pacemaker Explained*, available from “<http://www.clusterlabs.org/doc/>”. 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 with your preferred cluster management tool:

- Hawk: Chapter 6, *Configuring and Managing Cluster Resources (Web Interface)* (page 109)
- Pacemaker GUI: Chapter 5, *Configuring and Managing Cluster Resources (GUI)* (page 75)
- crm shell: Chapter 7, *Configuring and Managing Cluster Resources (Command Line)* (page 151)

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://refspecs.linuxbase.org/LSB_4.1.0/LSB-Core-generic/LSB-Core-generic/iniscriptact.html.

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 177). The cluster follows these specifications exactly. For a detailed list of all available OCF RAs, refer to Chapter 21, *HA OCF Agents* (page 311).

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 181).

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 your preferred cluster management tool:

- Hawk: Procedure 6.4, “Adding Primitive Resources” (page 117)
- Pacemaker GUI: Procedure 5.2, “Adding Primitive Resources” (page 80)
- crm shell: Section 7.3.1, “Creating Cluster Resources” (page 159)

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 4.2.5.1, “Groups” (page 56).

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 4.2.5.2, “Clones” (page 57).

Masters

Masters are a special type of clone resources, they can have multiple modes. For more information, refer to Section 4.2.5.3, “Masters” (page 58).

4.2.4 Resource Templates

If you want to create lots of resources with similar configurations, defining a resource template is the easiest way. Once defined, it can be referenced in primitives—or in certain types of constraints, as described in Section 4.4.3, “Resource Templates and Constraints” (page 68).

If a template is referenced in a primitive, the primitive will inherit all operations, instance attributes (parameters), meta attributes, and utilization attributes defined in the template. Additionally, you can define specific operations or attributes for your primitive. If any of these are defined in both the template and the primitive, the values defined in the primitive will take precedence over the ones defined in the template.

Learn how to define resource templates with your preferred cluster configuration tool:

- Hawk: Section 6.3.4, “Using Resource Templates” (page 120)
- crm shell: Section 7.3.2, “Creating Resource Templates” (page 160)

4.2.5 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.

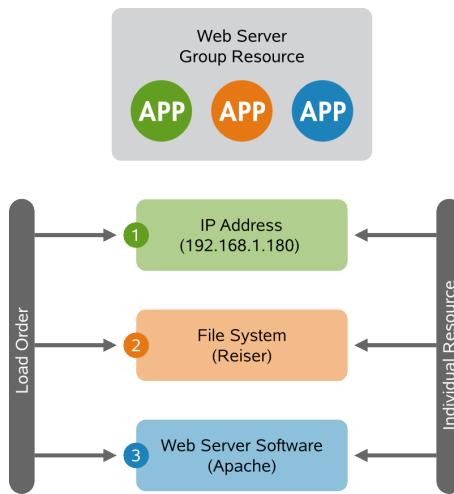
4.2.5.1 Groups

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 with resources it depends on. 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 your preferred cluster management tool:

- Hawk: Procedure 6.14, “Adding a Resource Group” (page 132)
- Pacemaker GUI: Procedure 5.13, “Adding a Resource Group” (page 97)
- `crm shell`: Section 7.3.9, “Configuring a Cluster Resource Group” (page 168)

4.2.5.2 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 4.2.5.3, “Masters” (page 58).

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 Explained*, available from “<http://www.clusterlabs.org/doc/>”. Refer to section *Clones - Resources That Should be Active on Multiple Hosts*.

Learn how to create clones with your preferred cluster management tool:

- Hawk: Procedure 6.15, “Adding or Modifying Clones” (page 133)
- Pacemaker GUI: Procedure 5.15, “Adding or Modifying Clones” (page 101)
- crm shell: Section 7.3.10, “Configuring a Clone Resource” (page 169).

4.2.5.3 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 Explained*, available from

[“<http://www.clusterlabs.org/doc/>”](http://www.clusterlabs.org/doc/). Refer to section *Multi-state - Resources That Have Multiple Modes*.

4.2.6 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 Pacemaker GUI as described in Procedure 5.3, “Adding or Modifying Meta and Instance Attributes” (page 82). Alternatively, use Hawk: Procedure 6.4, “Adding Primitive Resources” (page 117).

Table 4.1: Options for a Primitive Resource

| Option | Description |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>priority</code> | If not all resources can be active, the cluster will stop lower priority resources in order to keep higher priority ones active. |
| <code>target-role</code> | In what state should the cluster attempt to keep this resource? Allowed values: <code>stopped</code> , <code>started</code> . |
| <code>is-managed</code> | Is the cluster allowed to start and stop the resource? Allowed values: <code>true</code> , <code>false</code> . |
| <code>resource-stickiness</code> | How much does the resource prefer to stay where it is? Defaults to the value of <code>default-resource-stickiness</code> . |
| <code>migration-threshold</code> | How many failures should occur for this resource on a node before making the node ineligible to host this resource? Default: <code>none</code> . |

| Option | Description |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| multiple-active | What should the cluster do if it ever finds the resource active on more than one node? Allowed values: <code>block</code> (mark the resource as unmanaged), <code>stop_only</code> , <code>stop_start</code> . |
| 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: <code>never</code> . |
| allow-migrate | Allow resource migration for resources which support <code>migrate_to/migrate_from</code> actions. |

4.2.7 Instance Attributes (Parameters)

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 82). Alternatively, use Hawk: Procedure 6.4, “Adding Primitive Resources” (page 117). In the `crm` command line utility and in Hawk, instance attributes are called `params` or `Parameter`, respectively. 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 (without the optional parts):

```
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.

4.2.8 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 65).

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 |

| Operation | Description |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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 <code>stonith</code> . For STONITH resources, the default is <code>nothing</code> . |
| on-fail | <p>The action to take if this action ever fails. Allowed values:</p> <ul style="list-style-type: none"> • <code>ignore</code>: Pretend the resource did not fail. • <code>block</code>: Do not perform any further operations on the resource. • <code>stop</code>: Stop the resource and do not start it elsewhere. • <code>restart</code>: Stop the resource and start it again (possibly on a different node). • <code>fence</code>: Bring down the node on which the resource failed (STONITH). • <code>standby</code>: Move <i>all</i> resources away from the node on which the resource failed. |

| Operation | Description |
|----------------|------------------------------------------------------------------------------------------------------------------------------------|
| enabled | If <code>false</code> , the operation is treated as if it does not exist. Allowed values: <code>true</code> , <code>false</code> . |
| 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. |

4.2.9 Timeout Values

Timeouts values for resources can be influenced by the following parameters:

- `default-action-timeout` (global cluster option),
- `op_defaults` (global defaults for operations),
- a specific timeout value defined in a resource template,
- a specific timeout value defined for a resource.

Of the default values, `op_defaults` takes precedence over `default-action-timeout`. If a specific value is defined for a resource, it always takes precedence over any of the defaults (and over a value defined in a resource template).

For information on how to set the default parameters, refer to the technical information document *default action timeout and default op timeout*. It is available at <http://www.novell.com/support/search.do?cmd=displayKC&docType=kc&externalId=7009584>.

Getting timeout values right is very important. Setting them too low will result in a lot of (unnecessary) fencing operations for the following reasons:

1. If a resource runs into a timeout, it fails and the cluster will try to stop it.
2. If stopping the resource also fails (for example because the timeout for stopping is set too low), the cluster will fence the node (it considers the node where this happens to be out of control).

The best practice for setting timeout values is as follows:

- 1 Check how long it takes your resources to start and stop (under load).
- 2 Adjust the (default) timeout values accordingly:
 - 2a For example, set the `default-action-timeout` to 120 seconds.
 - 2b For resources that need longer periods of time, define individual timeout values.
- 3 When configuring operations for a resource, add separate `start` and `stop` operations. When configuring operations with Hawk or the Pacemaker GUI, both will provide useful timeout proposals for those operations.

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, Hawk, `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.

Usually, resources are only monitored by the cluster as long as they are running. However, to detect concurrency violations, also configure monitoring for resources which are stopped. For example:

```
primitive dummy1 ocf:heartbeat:Dummy \
    op monitor interval="300s" role="Stopped" timeout="10s" \
    op monitor interval="30s" timeout="10s"
```

This configuration triggers a monitoring operation every 300 seconds for the resource `dummy1` as soon as it is in `role="Stopped"`. When running, it will be monitored every 30 seconds.

Learn how to add monitor operations to resources with your preferred cluster management tool:

- Hawk: Procedure 6.13, “Adding or Modifying Monitor Operations” (page 130)
- Pacemaker GUI: Procedure 5.12, “Adding or Modifying Monitor Operations” (page 95)
- crm shell: Section 7.3.8, “Configuring Resource Monitoring” (page 168)

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 Colocation

Colocational 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 colocation, refer to the following documents. They are available at “<http://www.clusterlabs.org/doc/>” (page 74) and “<http://www.clusterlabs.org/wiki/Documentation>” (page 74), respectively:

- *Pacemaker Explained*, chapter *Resource Constraints*
- *Collocation Explained*
- *Ordering Explained*

Learn how to add the various kinds of constraints with the GUI in Section 5.3.4, “Configuring Resource Constraints” (page 86). If you prefer the command line approach, see Section 7.3.4, “Configuring Resource Constraints” (page 162).

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 Resource Templates and Constraints

If you have defined a resource template, it can be referenced in the following types of constraints:

- order constraints,
- colocation constraints,
- `rsc_ticket` constraints (for multi-site clusters).

However, colocation constraints must not contain more than one reference to a template. Resource sets must not contain a reference to a template.

Resource templates referenced in constraints stand for all primitives which are derived from that template. This means, the constraint applies to all primitive resources referencing the resource template. Referencing resource templates in constraints is an alternative to resource sets and can simplify the cluster configuration considerably. For details about resource sets, refer to Procedure 6.9, “Using Resource Sets for Colocation or Order Constraints” (page 125).

4.4.4 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 (`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 5.3.5, “Specifying Resource Failover Nodes” (page 90). If you prefer the command line approach, see Section 7.3.5, “Specifying Resource Failover Nodes” (page 164).

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 5.3.5, “Specifying Resource Failover Nodes” (page 90). If you prefer the command line approach, see Section 7.3.5, “Specifying Resource Failover Nodes” (page 164).

4.4.5 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.

4.4.6 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.

Learn how to configure these settings with your preferred cluster management tool:

- Pacemaker GUI: Section 5.3.7, “Configuring Placement of Resources Based on Load Impact” (page 91)
- `crm shell`: Section 7.3.7, “Configuring Placement of Resources Based on Load Impact” (page 165)

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 manually 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 High Availability Extension now also provides means to detect and configure both node capacity and resource requirements automatically:

The `NodeUtilization` resource agent checks the capacity of a node (regarding CPU and RAM). To configure automatic detection, create a clone resource of the following class, provider, and type: `ofc:pacemaker:NodeUtilization`. One instance of the clone should be running on each node. After the instance has started, a utilization section will be added to the node's configuration in CIB.

For automatic detection of a resource's minimal requirements (regarding RAM and CPU) the `Xen` resource agent has been improved. Upon start of a `Xen` resource, it will reflect the consumption of RAM and CPU. Utilization attributes will automatically be added to the resource configuration.

Apart from detecting the minimal requirements, the High Availability Extension also allows to monitor the current utilization via the `VirtualDomain` resource agent. It detects CPU and RAM use of the virtual machine. To use this feature, configure a resource of the following class, provider and type:

`ofc:heartbeat:VirtualDomain`. Add the `dynamic_utilization` instance attribute (parameter) and set its value to `1`. This updates the utilization values in the CIB during each monitoring cycle.

Independent of manually or automatically configuring capacity and requirements, the placement strategy must 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 resource'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://www.clusterlabs.org/doc/>

Holds a number of comprehensive manuals, for example:

- *Pacemaker Explained*: Explains the concepts used to configure Pacemaker.
Contains comprehensive and very detailed information for reference.
- *Fencing and Stonith*: How to configure and use STONITH devices.
- *CRM Command Line Interface*: Introduction to the `crm` command line tool.

<http://www.clusterlabs.org/wiki/Documentation>

Features some more useful documentation, such as:

- *Collocation Explained*
- *Ordering Explained*

Configuring and Managing Cluster Resources (GUI)

This chapter introduces the Pacemaker GUI and covers basic tasks needed when configuring and managing cluster resources: modifying global cluster options, creating basic and advanced types of resources (groups and clones), configuring constraints, specifying failover nodes and fallback 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.

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` and adds the user to 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.

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 Logging in to a Cluster

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

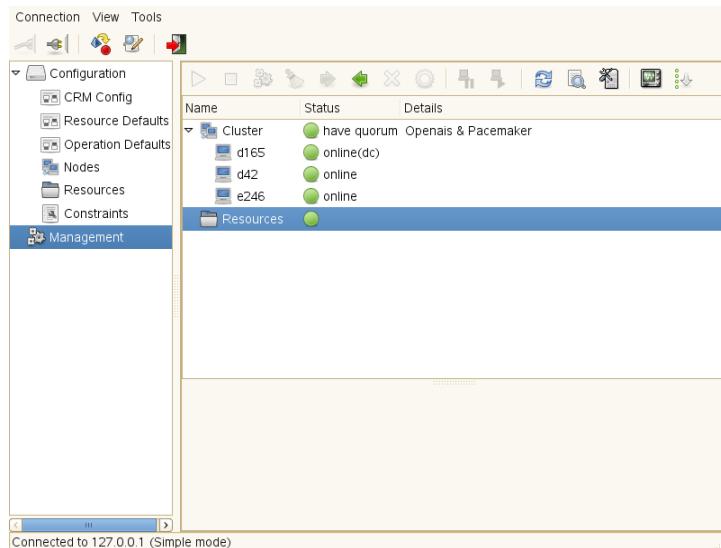


If you are running the Pacemaker GUI remotely, enter the IP address of a cluster node as *Server*. As *User Name*, you can also use any other user belonging to the `haclient` group to connect to the cluster.

5.1.2 Main Window

After being connected, the main window opens:

Figure 5.2: *Pacemaker GUI - Main Window*



NOTE: Available Functions in Pacemaker GUI

By default, users logged in as `root` or `hacluster` have full read-write access to all cluster configuration tasks. However, *Access Control Lists* (page 193) can be used to define fine-grained access permissions.

If ACLs are enabled in the CRM, the available functions in the Pacemaker GUI depend on the user role and access permission assigned to you.

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, it only shows the corresponding tab if a particular type of sub-object already exists in CIB. When adding a new sub-object, you will be 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, it only shows the values of global cluster options that have been actually set. It 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 50)
- Option `stonith-enabled` (page 51)

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, “Logging in to a Cluster” (page 76).
- 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*.

IMPORTANT: No Support Without STONITH

A cluster without STONITH enabled is not supported.

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 54).

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, “Logging in to a Cluster” (page 76).

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 52).

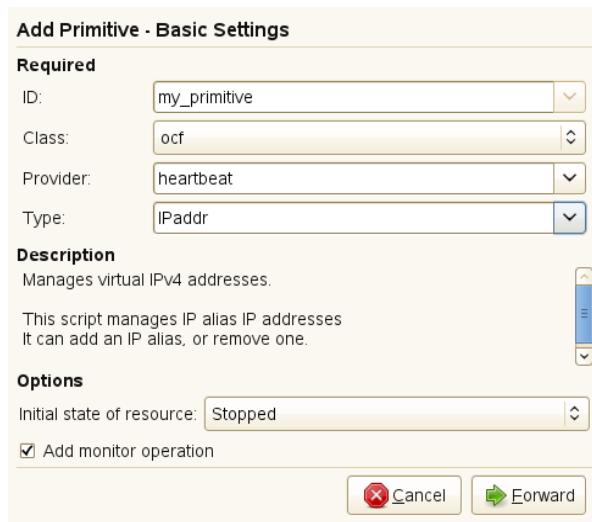
3c If you selected *ocf* as class, also specify the *Provider* of your OCF resource agent. The OCF specification allows multiple vendors to supply the same resource agent.

- 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.



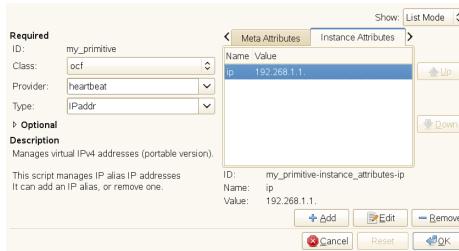
- 4** Click *Forward*. The next window shows a summary of the parameters that you have already defined for that resource. All required *Instance Attributes* for that resource are listed. You need to edit them in order to set them to appropriate values. You may also need to add more attributes, depending on your deployment and settings. For details how to do so, refer to Procedure 5.3, “Adding or Modifying Meta and Instance Attributes” (page 82).
- 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 4.2.7, “Instance Attributes (Parameters)” (page 60).
- **Meta attributes**—they tell the CRM how to treat a specific resource. For more information, refer to Section 4.2.6, “Resource Options (Meta Attributes)” (page 59).
- **Operations**—they are needed for resource monitoring. For more information, refer to Section 4.2.8, “Resource Operations” (page 62).

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.



- 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

IMPORTANT: No Support Without STONITH

A cluster without STONITH running is not supported.

By default, the global cluster option `stonith-enabled` is set to `true`: If no STONITH resources have been defined, the cluster will refuse to start any resources. To complete STONITH setup, you need to configure one or more STONITH resources. While they are configured similar to other resources, the behavior of STONITH resources is different in some respects. For details refer to Section 9.3, “STONITH Configuration” (page 184).

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, “Logging in to a Cluster” (page 76).
- 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 188).
-
- 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 82) and Section 5.3.8, “Configuring Resource Monitoring” (page 95).
 - 5** If all parameters are set according to your wishes, click *Apply* to finish the configuration of that resource. The configuration dialog closes 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 181).

5.3.3 Using Resource Templates

If you want to create several resources with similar configurations, defining a resource template is the easiest way. Once defined, it can be referenced in primitives or in certain types of constraints. For detailed information about function and use of resource templates, refer to Section 4.4.3, “Resource Templates and Constraints” (page 68).

- 1** Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 2** In the left pane, select *Resources* and click *Add > Template*.

- 3** Enter a unique *ID* for the template.
- 4** Specify the resource template as you would specify a primitive. Follow Procedure 5.2: “Adding Primitive Resources”, starting with Step 3b (page 80).
- 5** If all parameters are set according to your wishes, click *Apply* to finish the configuration of the resource template. The configuration dialog is closed and the main window shows the newly added resource template.

Procedure 5.5: *Referencing Resource Templates*

- 1** Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 2** To reference the newly created resource template in a primitive, follow these steps:
 - 2a** In the left pane, select *Resources* and click *Add > Primitive*.
 - 2b** Enter a unique *ID* and specify *Class*, *Provider*, and *Type*.
 - 2c** Select the *Template* to reference.
 - 2d** If you want to set specific instance attributes, operations or meta attributes that deviate from the template, continue to configure the resource as described in Procedure 5.2, “Adding Primitive Resources” (page 80).
- 3** To reference the newly created resource template in colocation or order constraints:
 - 3a** Configure the constraints as described in Procedure 5.7, “Adding or Modifying Colocational Constraints” (page 87) or Procedure 5.8, “Adding or Modifying Ordering Constraints” (page 88), respectively.
 - 3b** For colocation constraints, the *Resources* drop-down list will show the IDs of all resources and resource templates that have been configured. From there, select the template to reference.
 - 3c** Likewise, for ordering constraints, the *First* and *Then* drop-down lists will show both resources and resource templates. From there, select the template to reference.

5.3.4 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 67). 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 67).

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

Procedure 5.6: Adding or Modifying Locational Constraints

- 1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 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 mean it should not run on this node. Setting the score to `INFINITY` forces the resource to run on the node. Setting it to `-INFINITY` means the resources must not run on the node.
- 8 Select the *Node* for the constraint.



- 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 5.7: Adding or Modifying Colocational Constraints

- 1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 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 Colocation* 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 colocation source. The list shows the IDs of all resources and resource templates 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 colocation 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. Setting the score to `INFINITY` forces the resources to run on the same node. Setting it to `-INFINITY` means the resources must not run on the same node. 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 5.8: Adding or Modifying Ordering Constraints

- 1** Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 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:

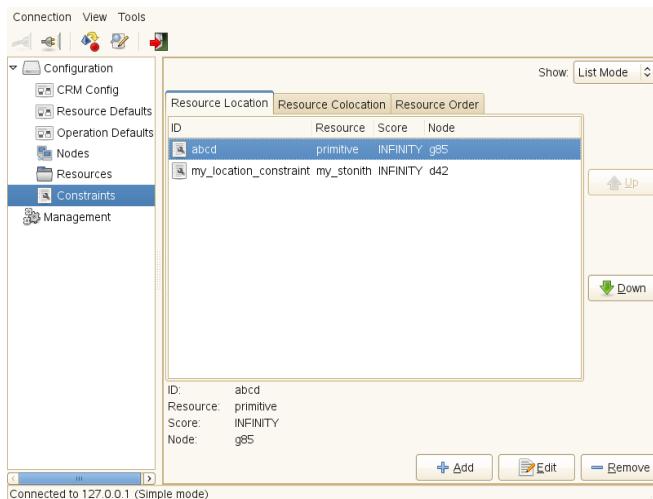
8a Specify a *Score*. If greater than zero, the constraint is mandatory, otherwise it is only a suggestion. The default value is `INFINITY`.

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



5.3.5 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 (`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 5.6, “Adding or Modifying Locational Constraints” (page 86).
- 2 Add the `migration-threshold` meta attribute to that resource as described in Procedure 5.3, “Adding or Modifying Meta and Instance Attributes” (page 82) and enter a *Value* for the `migration-threshold`. The value should be positive and less than 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 82) 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 69).

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 103) for the details.

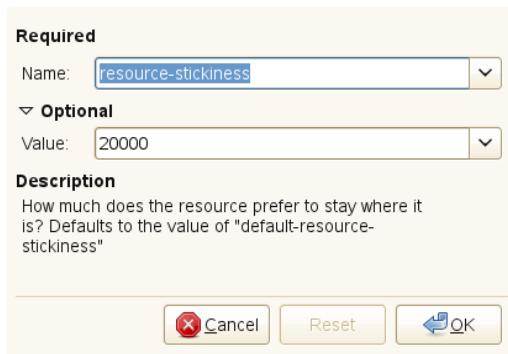
5.3.6 Specifying Resource Failback Nodes (Resource Stickiness)

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 the implications of different resource stickiness values, refer to Section 4.4.5, “Failback Nodes” (page 70).

Procedure 5.9: *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 82).



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

5.3.7 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.

Utilization attributes are used to configure both the resource's requirements and the capacity a node provides. The High Availability Extension now also provides means to detect and configure both node capacity and resource requirements automatically. For more details and a configuration example, refer to Section 4.4.6, “Placing Resources Based on Their Load Impact” (page 71).

To manually configure the resource's requirements and the capacity a node provides, proceed as described in Procedure 5.10, “Adding Or Modifying Utilization Attributes” (page 92). You can name the utilization attributes according to your preferences and define as many name/value pairs as your configuration needs.

Procedure 5.10: 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 92) and Step 3 (page 93).

- 1** Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 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 93) to Step 6 (page 93).

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 94) 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

The screenshot shows a configuration dialog for a node. At the top, there is a "Show: List Mode" dropdown. Below it, the "Required" section contains fields for ID (set to "bourbaki"), Uname (set to "bourbaki"), and Type (set to "normal"). The "Optional" section is expanded, showing an "Utilization" tab selected. This tab displays a table with two rows: "cpu" with value "8" and "memory" with value "16384". Below the table are buttons for "Add", "Edit", and "Remove". At the bottom of the dialog are "Cancel", "Reset", and "OK" buttons.

| Name | Value |
|--------|-------|
| cpu | 8 |
| memory | 16384 |

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

The screenshot shows a configuration dialog for a resource. At the top, there is a "Show: List Mode" dropdown. Below it, the "Required" section contains fields for ID (set to "xen1"), Class (set to "ocf"), Provider (set to "heartbeat"), and Type (set to "Xen"). The "Optional" section is expanded, showing a "Utilization" tab selected. This tab displays a table with two rows: "cpu" with value "4" and "memory" with value "4096". Below the table are buttons for "Add", "Edit", and "Remove". At the bottom of the dialog are "Cancel", "Reset", and "OK" buttons.

| Name | Value |
|--------|-------|
| cpu | 4 |
| memory | 4096 |

After (manual or automatic) configuration of 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 4.4.6, “Placing Resources Based on Their Load Impact” (page 71).

Procedure 5.11: *Setting the Placement Strategy*

- 1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 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*.

5.3.8 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 5.12: *Adding or Modifying Monitor Operations*

- 1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 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.

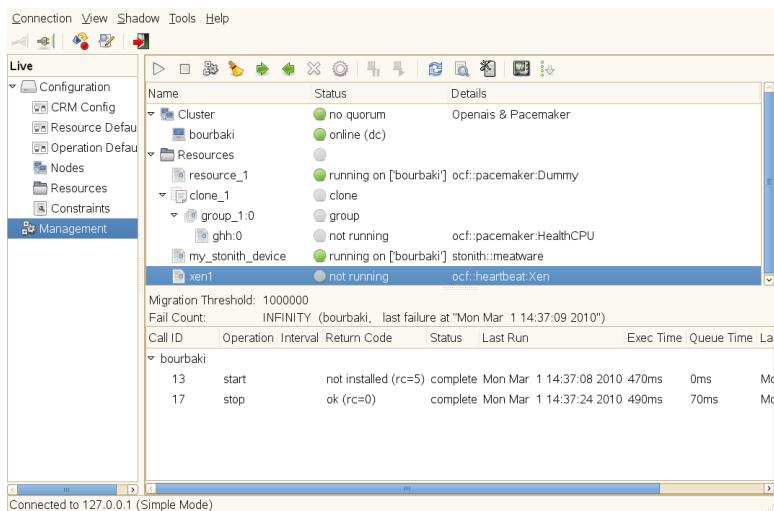


- 6 In the *Timeout* field, enter a value in seconds. After the specified timeout period, the operation will be treated as `failed`. The PE will decide what to do or execute what you specified in the *On Fail* field of the monitor operation.
- 7 If needed, expand the *Optional* section and add parameters, 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 65).

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



5.3.9 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 4.2.5.1, “Groups” (page 56).

NOTE: Empty Groups

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

Procedure 5.13: Adding a Resource Group

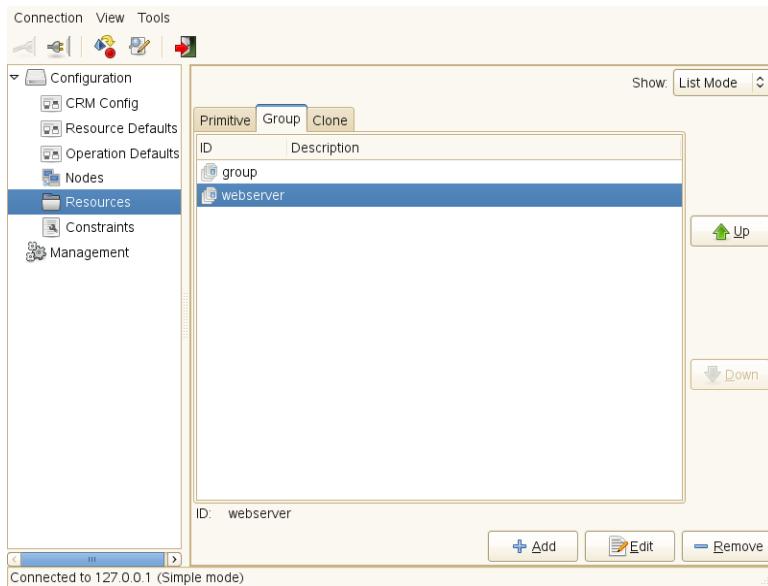
- 1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 2 In the left pane, select *Resources* and click *Add > Group*.
- 3 Enter a unique *ID* for the group.

- 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 80).
- 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

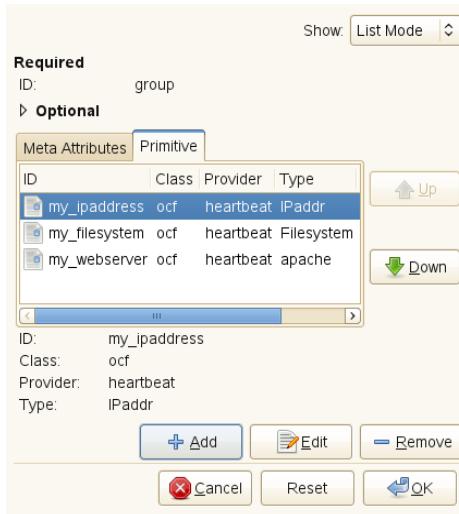


Let us assume you already have created a resource group as explained in Procedure 5.13, “Adding a Resource Group” (page 97). The following procedure shows you how to modify the group to match Example 4.1, “Resource Group for a Web Server” (page 56).

Procedure 5.14: Adding Resources to an Existing Group

- 1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 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, 192.168.1.180.
- 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 99) to Step 4h (page 100), until you have configured all sub-resources for the group.



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.

5.3.10 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 4.2.5.2, “Clones” (page 57).

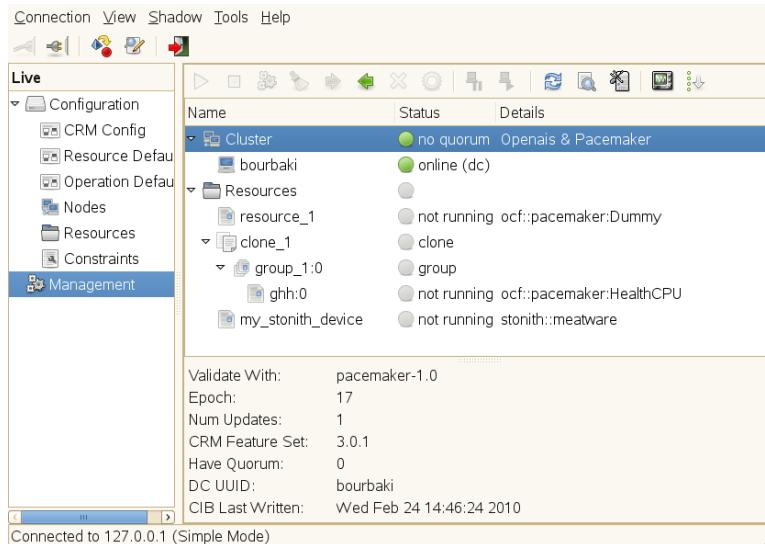
Procedure 5.15: *Adding or Modifying Clones*

- 1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 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 80) or Procedure 5.13, “Adding a Resource Group” (page 97).
- 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



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 107).

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 5.16: *Starting A New Resource*

- 1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 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 by 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 5.17: *Cleaning Up A Resource*

- 1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 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 the man pages of `crm_resource` and `crm_failcount`.

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 5.18: *Removing a Cluster Resource*

- 1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 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 5.17, “Cleaning Up A Resource” (page 104).
- 5 Stop the resource.

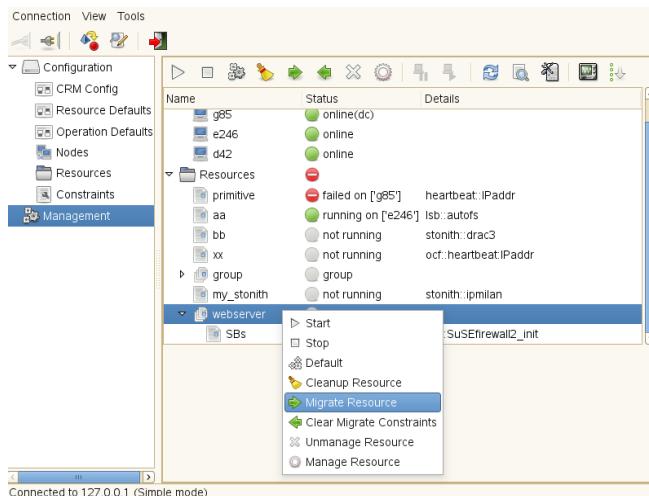
- 6 Remove all constraints that relate to the resource, otherwise removing the resource will not be possible.

5.4.4 Migrating Cluster Resources

As mentioned in Section 5.3.5, “Specifying Resource Failover Nodes” (page 90), 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.

Procedure 5.19: Manually Migrating a Resource

- 1 Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 2 Click *Management* in the left pane.
- 3 Right-click the respective resource in the right pane and select *Migrate Resource*.



- 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 5.20: Clearing a Migration Constraint

- 1** Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 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 the `crm_resource` man page or *Pacemaker Explained*, available from “<http://www.clusterlabs.org/doc/>”. 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 5.21: Changing Management Mode of Resources

- 1** Start the Pacemaker GUI and log in to the cluster as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 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 (Web Interface)

6

In addition to the `crm` command line tool and the Pacemaker GUI, the High Availability Extension also comes with the HA Web Konsole (Hawk), a Web-based user interface for management tasks. It allows you to monitor and administer your Linux cluster from non-Linux machines as well. Furthermore, it is the ideal solution in case your system only provides a minimal graphical user interface.

This chapter introduces Hawk and covers basic tasks for configuring and managing cluster resources: modifying global cluster options, creating basic and advanced types of resources (groups and clones), configuring constraints, specifying failover nodes and fallback nodes, configuring resource monitoring, starting, cleaning up or removing resources, and migrating resources manually. For detailed analysis of the cluster status, Hawk generates a cluster report (`hb_report`). You can view the cluster history or explore potential failure scenarios with the simulator.

The Web interface is included in the `hawk` package. It must be installed on all cluster nodes you want to connect to with Hawk. On the machine from which you want to access a cluster node using Hawk, 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 Hawk, the respective user must be a member of the `haclient` group. The installation creates a Linux user named `hacluster` and adds the user to the `haclient` group.

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

Do this on every node you will connect to with Hawk.

6.1 Hawk—Overview

Learn how to start Hawk, how to log in to the cluster and get to know the Hawk basics.

6.1.1 Starting Hawk and Logging In

Procedure 6.1: Starting Hawk

To use Hawk, the respective Web service must be started on the node that you want to connect to via the Web interface. For communication, the standard HTTPS 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 Hawk to start automatically at boot time, execute the following command:

```
chkconfig hawk on
```

- 1** On any machine, start a Web browser and make sure that JavaScript and cookies are enabled.
- 2** Point it at the IP address or hostname of any cluster node running the Hawk Web service, 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 Hawk 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 149).

-
- 3 On the Hawk login screen, enter the *Username* and *Password* of the hacluster user (or of any other user that is a 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`.

6.1.2 Main Screen: Cluster Status

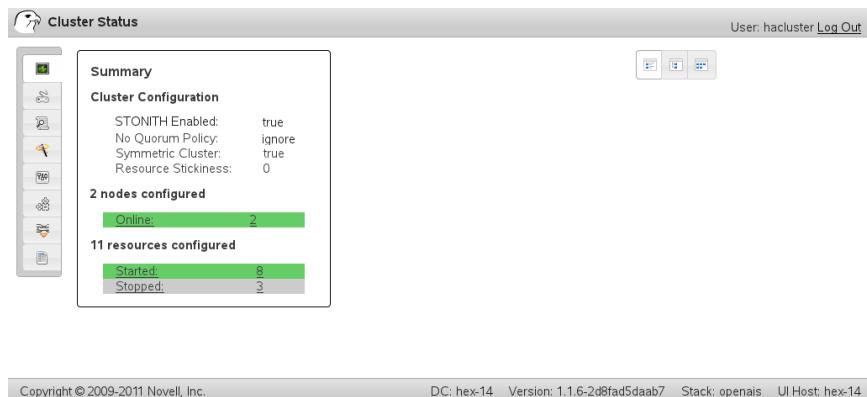
After logging in, Hawk displays the *Cluster Status* screen. It shows a summary with the most important global cluster parameters and the status of your cluster nodes and resources. The following color code is used for status display of nodes and resources:

Hawk Color Code

- 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.
- Gray: 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 gray (if they have been shut down cleanly).

If a resource has failed, a failure message with the details is shown in red at the top of the screen.

Figure 6.1: Hawk—Cluster Status (Summary View)



The *Cluster Status* screen refreshes itself in near real-time. Choose between the following views:

Summary View

Shows the most important global cluster parameters and the status of your cluster nodes and resources at the same time. If your setup includes Multi-Site Clusters (Geo Clusters) (page 144), the summary view also shows tickets. To view details about all elements belonging to a certain category (tickets, nodes, or resources), click the category title, which is marked as link. Otherwise click the individual elements for details.

Tree View

Presents an expandable view of the most important global cluster parameters and the status of your cluster nodes and resources. Click the arrows to expand or collapse the elements belonging to the respective category. In contrast to the *Summary View* this view not only shows the IDs and status of resources but also the type (for example, primitive, clone, or group).

Table View

This view is especially useful for larger clusters, because it shows in a concise way which resources are currently running on which node. Inactive nodes or resources are also displayed.

To perform basic operator tasks on nodes and resources (like starting or stopping resources, bringing nodes online, or viewing details), click the wrench icon next to the respective node or resource to access a context menu.

For more complex tasks like configuring resources, constraints, or global cluster options, use the navigation bar at the left hand side. From there, you can also generate a cluster report (`hb_report`), view the cluster history, or explore potential failure scenarios with the simulator.

NOTE: Available Functions in Hawk

By default, users logged in as `root` or `hacluster` have full read-write access to all cluster configuration tasks. However, *Access Control Lists* (page 193) can be used to define fine-grained access permissions.

If ACLs are enabled in the CRM, the available functions in Hawk depend on the user role and access permissions assigned to you. In addition, the following functions in Hawk can only be executed by the user `hacluster`:

- Generating a `hb_report`.
 - Using the history explorer.
 - Viewing recent events for nodes or resources.
-

6.2 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 cluster management tools like Pacemaker GUI, Hawk, 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 50)
- Option `stonith-enabled` (page 51)

Procedure 6.2: Modifying Global Cluster Options

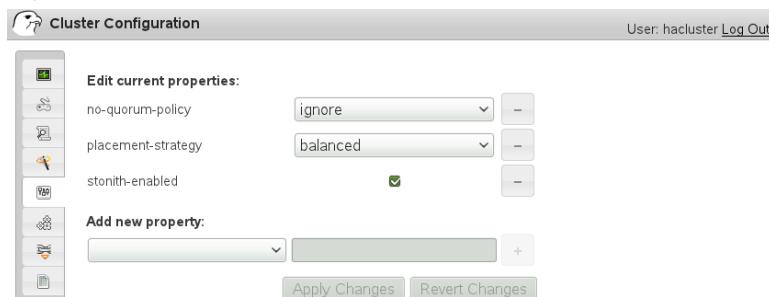
- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 In the left navigation bar, select *Cluster Properties* to view the global cluster options and their current values.
- 3 Depending on your cluster requirements, set *no-quorum-policy* to the appropriate value.
- 4 If you need to disable fencing for any reasons, deselect *stonith-enabled*.

IMPORTANT: No Support Without STONITH

A cluster without STONITH enabled is not supported.

- 5 To remove a cluster property, click the minus icon next to the property.
- 6 To add a new property, choose one from the drop-down list and click the plus icon.
- 7 Confirm your changes.

Figure 6.2: Hawk—Global Cluster Properties



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, 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 54). Apart from the basic specification of a resource (ID, class, provider, and type), you can add or modify the following parameters during or after creation of a resource:

- Instance attributes (parameters) determine which instance of a service the resource controls. For more information, refer to Section 4.2.7, “Instance Attributes (Parameters)” (page 60).

When creating a resource, Hawk automatically shows any required parameters. Edit them to get a valid resource configuration.

- Meta attributes tell the CRM how to treat a specific resource. For more information, refer to Section 4.2.6, “Resource Options (Meta Attributes)” (page 59).

When creating a resource, Hawk automatically lists the important meta attributes for that resource (for example, the `target-role` attribute that defines the initial state of a resource. By default, it is set to `Stopped`, so the resource will not start immediately).

- Operations are needed for resource monitoring. For more information, refer to Section 4.2.8, “Resource Operations” (page 62).

When creating a resource, Hawk displays the most important resource operations (`monitor`, `start`, and `stop`).

6.3.1 Configuring Resources with the Setup Wizard

The High Availability Extension comes with a predefined set of resources for some frequently used cluster scenarios, for example, setting up a highly available Web server. Find the predefined sets in the `hawk-templates` package. You can also define your own wizard templates. For detailed information, refer to <http://hg.clusterlabs.org/pacemaker/hawk/file/tip/doc/wizard.txt>. Hawk provides a wizard that guides you through all configuration steps.

Procedure 6.3: Using the Setup Wizard

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 In the left navigation bar, select *Setup Wizard*. The *Cluster Setup Wizard* shows a list of available resource sets. If you click an entry, Hawk displays a short help text about the resource set.
- 3 Select the resource set you want to configure and click *Next*.
- 4 Follow the instructions on the screen. If you need information about an option, click it to display a short help text in Hawk.

Figure 6.3: Hawk—Setup Wizard



6.3.2 Creating Simple Cluster Resources

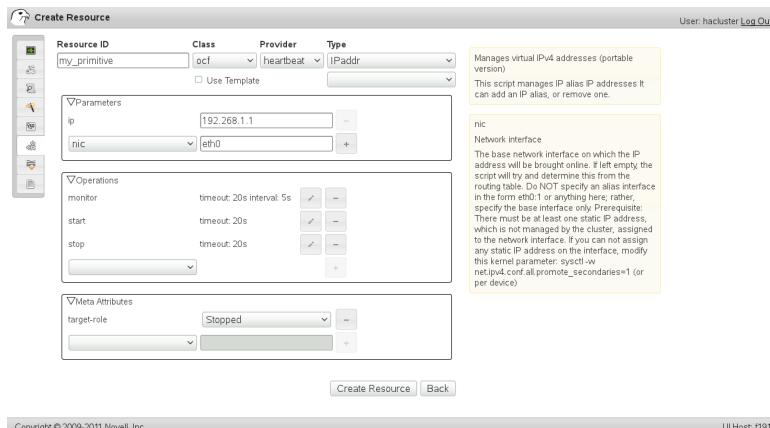
To create the most basic type of a resource, proceed as follows:

Procedure 6.4: Adding Primitive Resources

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
 - 2 In the left navigation bar, select *Resources*. The *Resources* screen shows categories for all types of resources. It lists any resources that are already defined.
 - 3 Select the *Primitive* category and click the plus icon.
 - 4 Specify the resource:
 - 4a Enter a unique *Resource ID*.
 - 4b From the *Class* list, select the resource agent class you want to use for the resource: *heartbeat*, *lsb*, *ocf* or *stonith*. For more information, see Section 4.2.2, “Supported Resource Agent Classes” (page 52).
 - 4c If you selected *ocf* as class, specify the *Provider* of your OCF resource agent. The OCF specification allows multiple vendors to supply the same resource agent.
 - 4d 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.
The selection you get in the *Type* list depends on the *Class* (and for OCF resources also on the *Provider*) you have chosen.
 - 5 Hawk automatically shows any required parameters for the resource plus an empty drop-down list that you can use to specify an additional parameter.
- To define *Parameters* (instance attributes) for the resource:
- 5a Enter values for each required parameter. A short help text is displayed as soon as you click the input field next to a parameter.

- 5b** To completely remove a parameter, click the minus icon next to the parameter.
 - 5c** To add another parameter, click the blank drop-down list, select a parameter and enter a value for it.
- 6** Hawk automatically shows the most important resource *Operations* and proposes default values. If you do not modify any settings here, Hawk will add the proposed operations and their default values as soon as you confirm your changes.
- For details on how to modify, add or remove operations, refer to Procedure 6.13, “Adding or Modifying Monitor Operations” (page 130).
- 7** Hawk automatically lists the most important meta attributes for the resource, for example `target-role`.
- To modify or add *Meta Attributes*:
- 7a** To set a (different) value for an attribute, select one from the drop-down list next to the attribute.
 - 7b** To completely remove a meta attribute, click the minus icon next to it.
 - 7c** To add another meta attribute, click the empty drop-down list and select an attribute. The default value for the attribute is displayed. If needed, change it as described above.
- 8** Click *Create Resource* to finish the configuration. A message at the top of the screen shows if the resource was successfully created or not.

Figure 6.4: Hawk—Primitive Resource



6.3.3 Creating STONITH Resources

IMPORTANT: No Support Without STONITH

A cluster without STONITH is not supported.

By default, the global cluster option `stonith-enabled` is set to `true`: If no STONITH resources have been defined, the cluster will refuse to start any resources. Configure one or more STONITH resources to complete the STONITH setup. While they are configured similar to other resources, the behavior of STONITH resources is different in some respects. For details refer to Section 9.3, “STONITH Configuration” (page 184).

Procedure 6.5: Adding a STONITH Resource

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 In the left navigation bar, select *Resources*. The *Resources* screen shows categories for all types of resources and lists all defined resources.
- 3 Select the *Primitive* category and click the plus icon.
- 4 Specify the resource:

- 4a** Enter a unique *Resource ID*.
 - 4b** From the *Class* list, select the resource agent class *stonith*.
 - 4c** From the *Type* list, select the STONITH plug-in for controlling your STONITH device. A short description for this plug-in is displayed.
-
- 5** Hawk automatically shows the required *Parameters* for the resource. Enter values for each parameter.
 - 6** Hawk displays the most important resource *Operations* and proposes default values. If you do not modify any settings here, Hawk will add the proposed operations and their default values as soon as you confirm.
 - 7** Adopt the default *Meta Attributes* settings if there is no reason to change them.
 - 8** Confirm your changes to create the STONITH resource.

To complete your fencing configuration, add constraints, use clones or both. For more details, refer to Chapter 9, *Fencing and STONITH* (page 181).

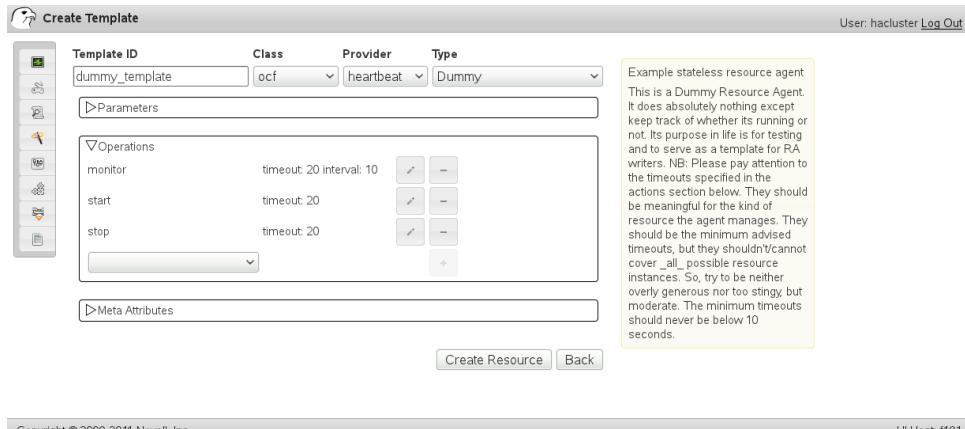
6.3.4 Using Resource Templates

If you want to create lots of resources with similar configurations, defining a resource template is the easiest way. Once defined, it can be referenced in primitives or in certain types of constraints. For detailed information about function and use of resource templates, refer to Section 4.4.3, “Resource Templates and Constraints” (page 68).

- 1** Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2** In the left navigation bar, select *Resources*. The *Resources* screen shows categories for all types of resources plus a *Template* category.
- 3** Select the *Template* category and click the plus icon.
- 4** Enter a *Template ID*.

- 5 Specify the resource template as you would specify a primitive. Follow Procedure 6.4: Adding Primitive Resources, starting with Step 4b (page 117).
- 6 Click *Create Resource* to finish the configuration. A message at the top of the screen shows if the resource template was successfully created.

Figure 6.5: Hawk—Resource Template



Procedure 6.6: Referencing Resource Templates

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 To reference the newly created resource template in a primitive, follow these steps:
 - 2a In the left navigation bar, select *Resources*. The *Resources* screen shows categories for all types of resources. It lists all defined resources.
 - 2b Select the *Primitive* category and click the plus icon.
 - 2c Enter a unique *Resource ID*.
 - 2d Activate *Use Template* and, from the drop-down list, select the template to reference.
 - 2e If needed, specify further *Parameters*, *Operations*, or *Meta Attributes* as described in Procedure 6.4, “Adding Primitive Resources” (page 117).

- 3 To reference the newly created resource template in colocational or order constraints, proceed as described in Procedure 6.8, “Adding or Modifying Colocational or Order Constraints” (page 123).

6.3.5 Configuring Resource Constraints

After you have configured all resources, specify how the cluster should handle them correctly. Resource constraints let you specify on which cluster nodes resources can run, in which order resources will be loaded, and what other resources a specific resource depends on.

For an overview of available types of constraints, refer to Section 4.4.1, “Types of Constraints” (page 67). When defining constraints, you also need to specify scores. For more information on scores and their implications in the cluster, see Section 4.4.2, “Scores and Infinity” (page 67).

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

Procedure 6.7: Adding or Modifying Location Constraints

For location constraints, specify a constraint ID, resource, score and node:

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 In the left navigation bar, select *Constraints*. The *Constraints* screen shows categories for all types of constraints. It lists all defined constraints.
- 3 To add a new *Location* constraint, click the plus icon in the respective category.

To modify an existing constraint, click the wrench icon next to the constraint and select *Edit Constraint*.

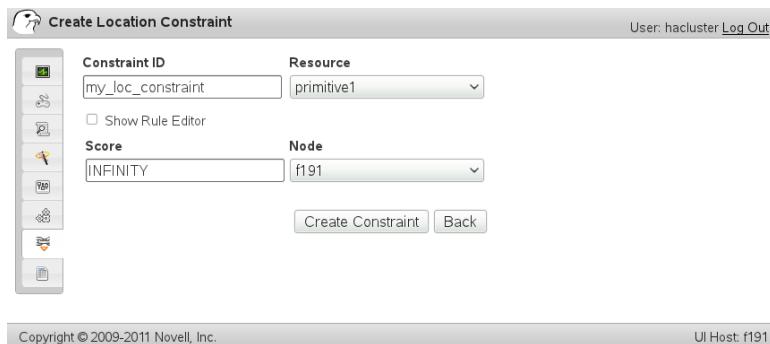
- 4 Enter a unique *Constraint ID*. When modifying existing constraints, the ID is already defined.
- 5 Select the *Resource* for which to define the constraint. The list shows the IDs of all resources that have been configured for the cluster.
- 6 Set the *Score* for the constraint. Positive values indicate the resource can run on the *Node* you specify in the next step. Negative values mean it should not run on that

node. Setting the score to `INFINITY` forces the resource to run on the node. Setting it to `-INFINITY` means the resources must not run on the node.

7 Select the *Node* for the constraint.

8 Click *Create Constraint* to finish the configuration. A message at the top of the screen shows if the constraint was successfully created.

Figure 6.6: Hawk—Location Constraint



Procedure 6.8: Adding or Modifying Colocational or Order Constraints

For both types of constraints specify a constraint ID and a score, then add resources to a dependency chain:

- 1** Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2** In the left navigation bar, select *Constraints*. The *Constraints* screen shows categories for all types of constraints and lists all defined constraints.
- 3** To add a new *Colocation* or *Order* constraint, click the plus icon in the respective category.

To modify an existing constraint, click the wrench icon next to the constraint and select *Edit Constraint*.

- 4** Enter a unique *Constraint ID*. When modifying existing constraints, the ID is already defined.
- 5** Define a *Score*.

For colocation constraints, the score determines the location relationship between the resources. Positive values indicate the resources should run on the same node. Negative values indicate the resources should not run on the same node. Setting the score to `INFINITY` forces the resources to run on the same node. Setting it to `-INFINITY` means the resources must not run on the same node. The score will be combined with other factors to decide where to put the resource.

For order constraints, the constraint is mandatory if the score is greater than zero, otherwise it is only a suggestion. The default value is `INFINITY`.

6 For order constraints, you can usually keep the option *Symmetrical* enabled. This specifies that resources are stopped in reverse order.

7 To define the resources for the constraint, follow these steps:

7a Select a resource from the list *Add resource to constraint*. The list shows the IDs of all resources and all resource templates configured for the cluster.

7b To add the selected resource, click the plus icon next to the list. A new list appears beneath. Select the next resource from the list. As both colocation and order constraints define a dependency between resources, you need at least two resources.

7c Select one of the remaining resources from the list *Add resource to constraint*. Click the plus icon to add the resource.

Now you have two resources in a dependency chain.

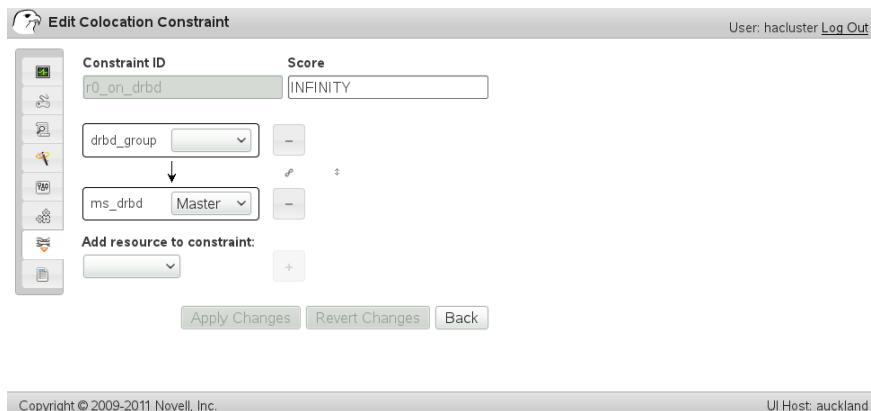
If you have defined an order constraint, the topmost resource will start first, then the second etc. Usually the resources will be stopped in reverse order.

However, if you have defined a colocation constraint, the arrow icons between the resources reflect their dependency, but *not* their start order. As the topmost resource depends on the next resource and so on, the cluster will first decide where to put the last resource, then place the depending ones based on that decision. If the constraint cannot be satisfied, the cluster may decide not to allow the resource to run at all.

7d Add as many resources as needed for your colocation or order constraint.

- 7e** If you want to swap the order of two resources, click the double arrow at the right hand side of the resources to swap the resources in the dependency chain.
- 8** If needed, specify further parameters for each resource, like the role (Master, Slave, Started, or Stopped).
- 9** Click *Create Constraint* to finish the configuration. A message at the top of the screen shows if the constraint was successfully created.

Figure 6.7: Hawk—Colocation Constraint

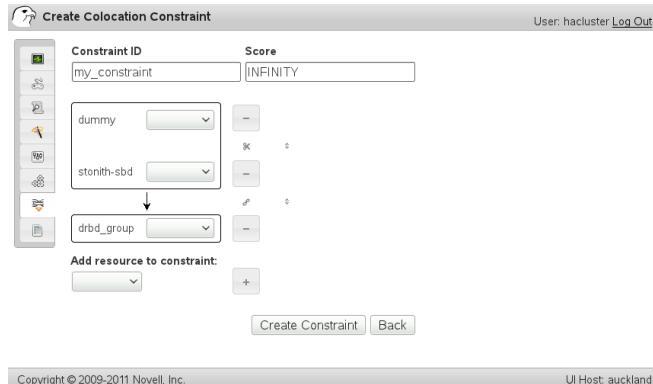


As an alternative format for defining colocation or ordering constraints, you can use resource sets. They have the same ordering semantics as groups.

Procedure 6.9: Using Resource Sets for Colocation or Order Constraints

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 Define colocation or order constraints as described in Procedure 6.8, “Adding or Modifying Colocational or Order Constraints” (page 123).
- 3 When you have added the resources to the dependency chain, you can put them into a resource set by clicking the chain icon at the right hand side. A resource set is visualized by a frame around the resources belonging to a set.

- 4 You can also add multiple resources to a resource set or create multiple resource sets.
- 5 To extract a resource from a resource set, click the scissors icon above the respective resource.



Copyright © 2009-2011 Novell, Inc. UI Host: auckland

The resource will be removed from the set and put back into the dependency chain at its original place.

- 6 Confirm your changes to finish the constraint configuration.

For more information on configuring constraints and detailed background information about the basic concepts of ordering and colocation, refer to the documentation available at “<http://www.clusterlabs.org/doc/>” (page 74) and “<http://www.clusterlabs.org/wiki/Documentation>” (page 74), respectively:

- *Pacemaker Explained*, chapter *Resource Constraints*
- *Collocation Explained*
- *Ordering Explained*

Procedure 6.10: Removing Constraints

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 In the left navigation bar, select *Constraints*. The *Constraints* screen shows categories for all types of constraints and lists all defined constraints.

- 3 Click the wrench icon next to a constraint and select *Remove Constraint*.

6.3.6 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 (`migration-threshold`), after which they will migrate to a new node. If you have more than two nodes in your cluster, the node to which a particular resource fails over is chosen by the High Availability software.

You can specify a specific node to which a resource will fail over by proceeding as follows:

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 Configure a location constraint for the resource as described in Procedure 6.7, “Adding or Modifying Location Constraints” (page 122).
- 3 Add the `migration-threshold` meta attribute to the resource as described in Procedure 6.4: Adding Primitive Resources, Step 7 (page 118) and enter a *Value* for the `migration-threshold`. The value should be positive and less than INFINITY.
- 4 If you want to automatically expire the failcount for a resource, add the `failure-timeout` meta attribute to the resource as described in Procedure 6.4: Adding Primitive Resources, Step 7 (page 118) and enter a *Value* for the `failure-timeout`.
- 5 If you want to specify additional failover nodes with preferences for a resource, create additional location constraints.

The process flow regarding migration thresholds and failcounts is demonstrated in Example 4.2, “Migration Threshold—Process Flow” (page 69).

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 6.4.2, “Cleaning Up Resources” (page 136) for details.

6.3.7 Specifying Resource Failback Nodes (Resource Stickiness)

A resource may fail back to its original node when that node is back online and in the cluster. To prevent this or to specify a different node for the resource to fail back to, change the stickiness value of the resource. You can either specify the resource stickiness when creating it or afterwards.

For the implications of different resource stickiness values, refer to Section 4.4.5, “Failback Nodes” (page 70).

Procedure 6.11: *Specifying Resource Stickiness*

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 Add the `resource-stickiness` meta attribute to the resource as described in Procedure 6.4: Adding Primitive Resources, Step 7 (page 118).
- 3 Specify a value between `-INFINITY` and `INFINITY` for the `resource-stickiness`.

6.3.8 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 so that their combined needs exceed the provided capacity, the performance of the resources diminishes or they fail.

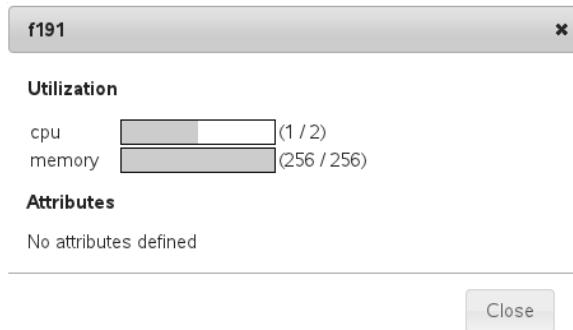
To take this into account, the High Availability Extension allows you to specify the following parameters:

1. The capacity a certain node *provides*.
2. The capacity a certain resource *requires*.
3. An overall strategy for placement of resources.

Utilization attributes are used to configure both the resource's requirements and the capacity a node provides. The High Availability Extension now also provides means to detect and configure both node capacity and resource requirements automatically. For more details and a configuration example, refer to Section 4.4.6, “Placing Resources Based on Their Load Impact” (page 71).

To display a node's capacity values (defined via utilization attributes) as well as the capacity currently consumed by resources running on the node, switch to the *Cluster Status* screen in Hawk and select the node you are interested in. Click the wrench icon next to the node and select *Show Details*.

Figure 6.8: Hawk—Viewing a Node's Capacity Values



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 4.4.6, “Placing Resources Based on Their Load Impact” (page 71).

Procedure 6.12: Setting the Placement Strategy

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 In the left navigation bar, select *Cluster Properties* to view the global cluster options and their current values.
- 3 From the *Add new property* drop-down list, choose *placement-strategy*.

- 4** Depending on your requirements, set *Placement Strategy* to the appropriate value.
- 5** Click the plus icon to add the new cluster property including its value.
- 6** Confirm your changes.

6.3.9 Configuring Resource Monitoring

The High Availability Extension can not only detect a node failure, but also when an individual resource on a node has failed. If you want to ensure that a resource is running, configure resource monitoring for it. For resource monitoring, specify 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 6.13: Adding or Modifying Monitor Operations

- 1** Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2** In the left navigation bar, select *Resources*. The *Resources* screen shows categories for all types of resources and lists all defined resources.
- 3** Select the resource to modify, click the wrench icon next to it and select *Edit Resource*. The resource definition is displayed. Hawk automatically shows the most important resource operations (`monitor`, `start`, `stop`) and proposes default values.
- 4** To change the values for an operation:
 - 4a** Click the pen icon next to the operation.
 - 4b** In the dialog that opens, specify the following values:
 - Enter a `timeout` 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.
 - For monitoring operations, define the monitoring `interval` in seconds.

If needed, use the empty drop-down list at the bottom of the *monitor* dialog to add more parameters, like *On Fail* (what to do if this action fails?) or *Requires* (what conditions need to be fulfilled before this action occurs?).



4c Confirm your changes to close the dialog and to return to the *Edit Resource* screen.

- 5** To completely remove an operation, click the minus icon next to it.
- 6** To add another operation, click the empty drop-down list and select an operation. A default value for the operation is displayed. If needed, change it as described above.
- 7** Click *Apply Changes* to finish the configuration. A message at the top of the screen shows if the resource was successfully updated or not.

For the processes which take place if the resource monitor detects a failure, refer to Section 4.3, “Resource Monitoring” (page 65).

6.3.10 Configuring a Cluster Resource Group

Some cluster resources depend on other components or resources and require that each component or resource starts in a specific order and runs 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 4.2.5.1, “Groups” (page 56).

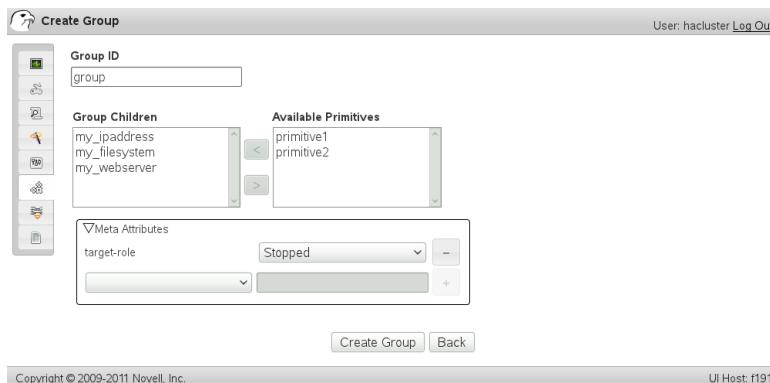
NOTE: Empty Groups

Groups must contain at least one resource, otherwise the configuration is not valid. In Hawk, primitives cannot be created or modified while creating a group. Before adding a group, create primitives and configure them as desired. For details, refer to Procedure 6.4, “Adding Primitive Resources” (page 117).

Procedure 6.14: Adding a Resource Group

- 1** Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2** In the left navigation bar, select *Resources*. The *Resources* screen shows categories for all types of resources and lists all defined resources.
- 3** Select the *Group* category and click the plus icon.
- 4** Enter a unique *Group ID*.
- 5** To define the group members, select one or multiple entries in the list of *Available Primitives* and click the < icon to add them to the *Group Children* list. Any new group members are added to the bottom of the list. To define the order of the group members, you currently need to add and remove them in the order you desire.
- 6** If needed, modify or add *Meta Attributes* as described in Adding Primitive Resources, Step 7 (page 118).
- 7** Click *Create Group* to finish the configuration. A message at the top of the screen shows if the group was successfully created.

Figure 6.9: Hawk—Resource Group



6.3.11 Configuring a Clone Resource

If you want certain resources to run simultaneously on multiple nodes in your cluster, configure these resources as clones. For example, cloning makes sense for resources like STONITH and cluster file systems like OCFS2. You can clone any resource provided. Cloning is supported by the resource’s Resource Agent. Clone resources may be configured differently depending on which nodes they are running on.

For an overview of the available types of resource clones, refer to Section 4.2.5.2, “Clones” (page 57).

NOTE: Sub-resources for Clones

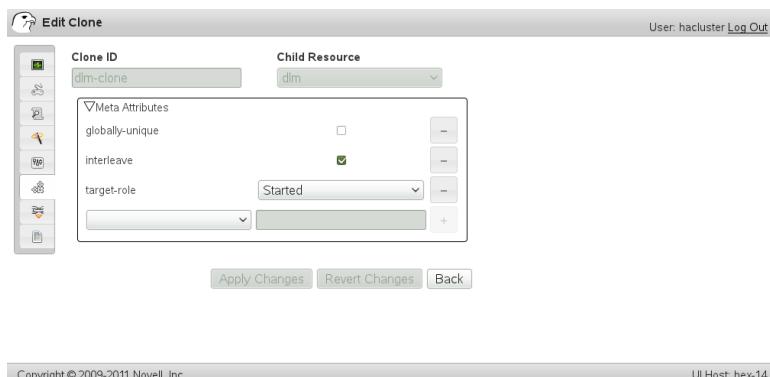
Clones can either contain a primitive or a group as sub-resources. In Hawk, sub-resources cannot be created or modified while creating a clone. Before adding a clone, create sub-resources and configure them as desired. For details, refer to Procedure 6.4, “Adding Primitive Resources” (page 117) or Procedure 6.14, “Adding a Resource Group” (page 132).

Procedure 6.15: Adding or Modifying Clones

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 In the left navigation bar, select *Resources*. The *Resources* screen shows categories for all types of resources and lists all defined resources.

- 3** Select the *Clone* category and click the plus icon.
- 4** Enter a unique *Clone ID*.
- 5** From the *Child Resource* list, select the primitive or group to use as a sub-resource for the clone.
- 6** If needed, modify or add *Meta Attributes* as described in Procedure 6.4: Adding Primitive Resources, Step 7 (page 118).
- 7** Click *Create Clone* to finish the configuration. A message at the top of the screen shows if the clone was successfully created.

Figure 6.10: Hawk—Clone Resource



6.4 Managing Cluster Resources

In addition to configuring your cluster resources, Hawk allows you to manage existing resources from the *Cluster Status* screen. For a general overview of the screen, its different views and the color code used for status information, refer to Section 6.1.2, “Main Screen: Cluster Status” (page 111).

Basic resource operations can be executed from any cluster status view. Both *Tree View* and *Table View* let you access the individual resources directly. However, in the *Summary View* you need to click the links in the resources category first to display the resource details.

6.4.1 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 via 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 Procedure 6.21, “Changing Management Mode of Resources” (page 139).

When creating a resource with Hawk, you can set its initial state with the `target-role` meta attribute. If you set its value to `stopped`, the resource does not start automatically after being created.

Procedure 6.16: Starting A New Resource

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 In the left navigation bar, select *Cluster Status*.
- 3 In one of the individual resource views, click the wrench icon next to the resource and select *Start*. To continue, confirm the message that appears. As soon as the resource has started, Hawk changes the resource's color to green and shows on which node it is running.

6.4.2 Cleaning Up Resources

A resource will be automatically restarted if it fails, but each failure increases the resource's failcount.

If a `migration-threshold` has been set for the resource, the node will no longer run the resource when the number of failures reaches 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 6.17: Cleaning Up A Resource

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 In the left navigation bar, select *Cluster Status*.
- 3 In one of the individual resource views, click the wrench icon next to the failed resource and select *Clean Up*. To continue, confirm the message that appears.

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

For more information, see the man pages of `crm_resource` and `crm_failcount`.

6.4.3 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

A cluster resource cannot be removed if its 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 6.18: *Removing a Cluster Resource*

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 In the left navigation bar, select *Cluster Status*.
- 3 Clean up the resource on all nodes as described in Procedure 6.17, “Cleaning Up A Resource” (page 136).
- 4 In one of the individual resource views, click the wrench icon next to the resource and select *Stop*. To continue, confirm the message that appears.
- 5 Remove all constraints that relate to the resource, otherwise removing the resource will not be possible. For details, refer to Procedure 6.10, “Removing Constraints” (page 126).
- 6 If the resource is stopped, click the wrench icon next to it and select *Delete Resource*.

6.4.4 Migrating Cluster Resources

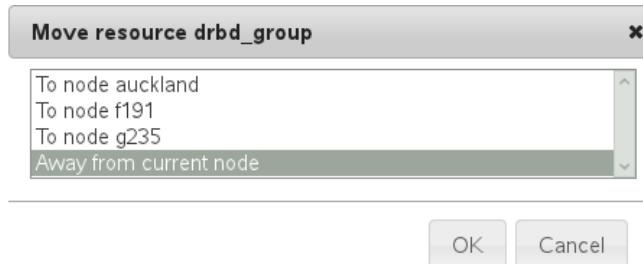
As mentioned in Section 6.3.6, “Specifying Resource Failover Nodes” (page 127), the cluster will fail over (migrate) resources automatically in case of software or hardware failures—according to certain parameters you can define (for example, migration threshold or resource stickiness). Apart from that, you can also manually migrate a resource to another node in the cluster (or decide to just move it away from the current node and leave the decision where to put it to the cluster).

Procedure 6.19: *Manually Migrating a Resource*

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 In the left navigation bar, select *Cluster Status*.
- 3 In one of the individual resource views, click the wrench icon next to the resource and select *Move*.
- 4 In the new window, select the node to which to move the resource.

This creates a location constraint with an `INFINITY` score for the destination node.

- 5** Alternatively, select to move the resource *Away from current node*.



This creates a location constraint with a `-INFINITY` score for the current node.

- 6** Click *OK* to confirm the migration.

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

Procedure 6.20: Clearing a Migration Constraint

- 1** Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2** In the left navigation bar, select *Cluster Status*.
- 3** In one of the individual resource views, click the wrench icon next to the resource and select *Drop Relocation Rule*. To continue, confirm the message that appears.

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 the `crm_resource` man page or *Pacemaker Explained*, available from “<http://www.clusterlabs.org/doc/>”. Refer to section *Resource Migration*.

6.4.5 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 6.21: Changing Management Mode of Resources

- 1** Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2** In the left navigation bar, select *Resources*. Select the resource you want to put in unmanaged mode, click the wrench icon next to the resource and select *Edit Resource*.
- 3** Open the *Meta Attributes* category.
- 4** From the empty drop-down list, select the *is-managed* attribute and click the plus icon to add it.
- 5** Deactivate the checkbox next to *is-managed* to put the resource in unmanaged mode and confirm your changes.
- 6** After you have finished the maintenance task for that resource, reactivate the checkbox next to the *is-managed* attribute for that resource.

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

6.4.6 Viewing the Cluster History

Hawk provides the following possibilities to view past events on the cluster (on different levels and in varying detail).

Procedure 6.22: Viewing Recent Events of Nodes or Resources

- 1** Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).

- 2 In the left navigation bar, select *Cluster Status*.
- 3 In the *Tree View* or *Table View*, click the wrench icon next to the resource or node you are interested in and select *View Recent Events*.

The dialog that opens shows the events of the last hour.

Procedure 6.23: Viewing Transitions with the History Explorer

The *History Explorer* provides transition information for a time frame that you can define:

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 In the left navigation bar, select *History Explorer*.
- 3 By default, the period to explore is set to the last 24 hours. To modify this, set another *Start Time* and *End Time*.
- 4 Click *Display* to start collecting transition data.

Figure 6.11: Hawk—History Report

The screenshot shows the Hawk—History Report interface. At the top, there is a header bar with the title "History Explorer", the user name "User: hacluster", and a "Log Out" link. Below the header is a search bar with fields for "Start Time" (set to "2011-11-18 12:34") and "End Time" (set to "2011-11-18 17:34"), and a "Display" button. To the right of the search bar is a "User: hacluster" link and a "Log Out" link. The main content area contains a table of recent transitions:

| Time | PE Input | Node | | |
|---------------------|-----------------------------|------|-------------------------|-----------------------------|
| 2011-11-18 17:26:20 | pe-input-34 | f191 | Details | Graph (xml) |
| 2011-11-18 17:25:28 | pe-input-32 | f191 | Details | Graph (xml) |
| 2011-11-18 17:24:02 | pe-input-31 | f191 | Details | Graph (xml) |

At the bottom of the interface, there is a footer bar with the copyright notice "Copyright © 2009-2011 Novell, Inc." and the UI host information "UI Host: auckland".

The following information is displayed:

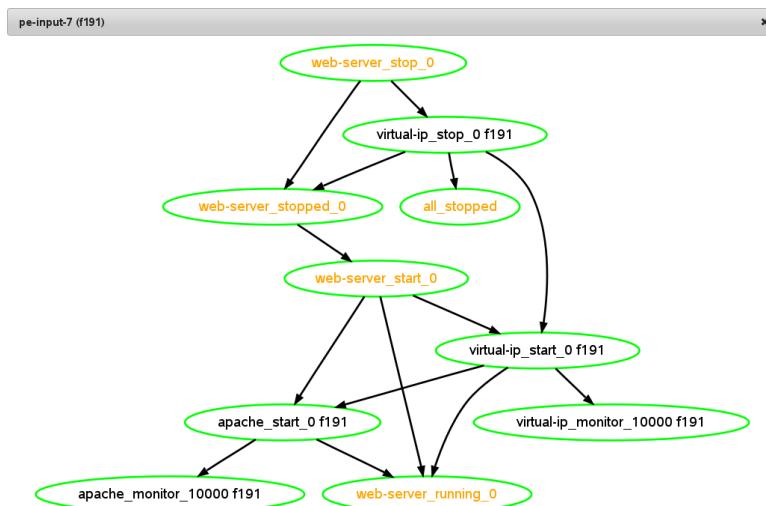
- The time line of all past transitions in the cluster.
- The `pe-input` files for each transition. For each transition, the cluster saves a copy of the state which is provided to the policy engine as input. The path to this archive

is logged. The files are only generated on the Designated Coordinator (DC), but as the DC can change, there may be pe-input files from several nodes listed in the *History Explorer*. The files show what the Policy Engine (PE) *planned* to do.

- Graph, XML representation and details of each transition.

If you choose to show the *Graph*, the PE is reinvoked (using the pe-input files), and generates a graphical visualization of the transition. Alternatively, you can view the XML representation of the graph. Clicking *Details* shows snippets of var/log/ messages that belong to that particular transition.

Figure 6.12: Hawk History Report—Transition Graph



6.4.7 Exploring Potential Failure Scenarios

Hawk provides a *Simulator* that allows you to explore failure scenarios before they happen. After switching to the simulator mode, change the status of nodes or execute multiple resource operations to see how the cluster would behave should these events occur.

Procedure 6.24: Switching to Simulator Mode

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).

2 In the left navigation bar, select *Simulator*.

Hawk's background changes color to indicate the simulator is active. A simulator dialog opens in the bottom right hand corner of the screen. Its title *Simulator (initial state)* indicates that *Cluster Status* screen still reflects the current state of the cluster.

3 To simulate status change of a node:

3a Click *+Node* in the simulator control dialog.

3b Select the *Node* you want to manipulate and select its target *State*.

3c Confirm your changes to add them to the queue of events listed in the controller dialog below *Injected State*.

4 To simulate a resource operation:

4a Click *+Op* in the simulator control dialog.

4b Select the *Resource* to manipulate and the *Operation* to simulate. If necessary, define an *Interval*. Select the *Node* on which to run the operation and the targeted *Result*.

4c Confirm your changes to add them to the queue of events listed in the controller dialog below *Injected State*.

5 Repeat the previous steps for any other events you wish to simulate.

6 To remove any of the events listed in *Injected State*, select the entry and click the minus icon beneath the list.

7 To start the simulation, click *Run* in the simulator control dialog. The *Cluster Status* screen displays the simulated events. For example, if you marked a node as unclean, it will now be shown offline, and all its resources will be stopped. The simulator control dialog changes to *Simulator (final state)*.

8 To view more detailed information:

8a To see log snippets of what occurred, click the *Details* link in the simulator dialog.

8b To see the transition graph, click the *Graph* link.

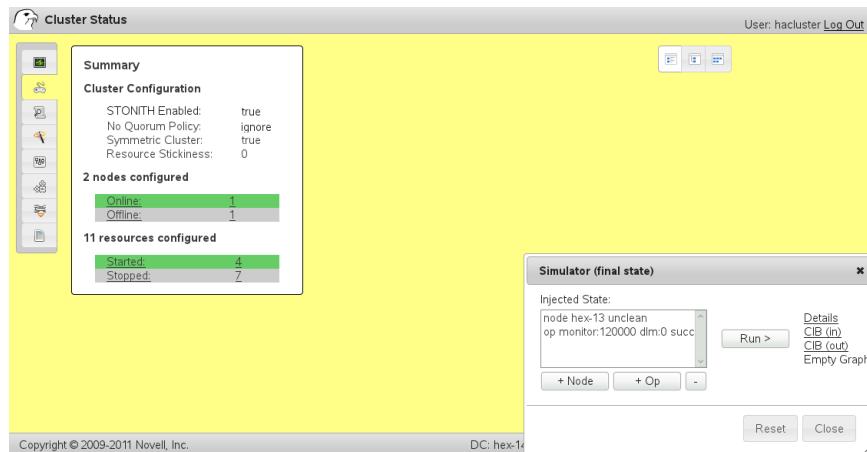
8c To display the initial CIB state, click *CIB (in)*.

8d To see what the CIB would look like after the transition, click *CIB (out)*.

9 To start from scratch with a new simulation, use the *Reset* button.

10 To exit the simulation mode, close the simulator control dialog. The *Cluster Status* screen switches back to its normal color and displays the current cluster state.

Figure 6.13: Hawk—Simulator



6.4.8 Generating a Cluster Report

For analysis and diagnosis of problems occurring on the cluster, Hawk can generate a cluster report that collects information from all nodes in the cluster.

Procedure 6.25: Generating a *hb_report*

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).

- 2** In the left navigation bar, select *Generate hb_report*.
- 3** By default, the period to examine is the last hour. To modify this, set another *Start Time* and *End Time*.
- 4** Click *Generate*.
- 5** After the report has been created, download the `*.tar.bz2` file by clicking the respective link.

For more information about the log files that the `hb_report` covers, refer to “How can I create a report with an analysis of all my cluster nodes?” (page 307).

6.5 Multi-Site Clusters (Geo Clusters)

Support for multi-site clusters is available as a separate option to SUSE Linux Enterprise High Availability Extension. For an introduction and detailed information on how to set up multi-site clusters, refer to Chapter 13, *Multi-Site Clusters (Geo Clusters)* (page 215).

Some of the required steps need to be executed within the individual clusters (that are then combined to a multi-site cluster), while other steps take place on an “inter-cluster” layer. Therefore not all setup steps can alternatively be executed with Hawk. But Hawk offers the following features related to multi-site clusters:

- Viewing Tickets (page 144)
- Configuring Additional Cluster Resources and Constraints (page 146)
- Testing the Impact of Ticket Failover (page 148)

6.5.1 Viewing Tickets

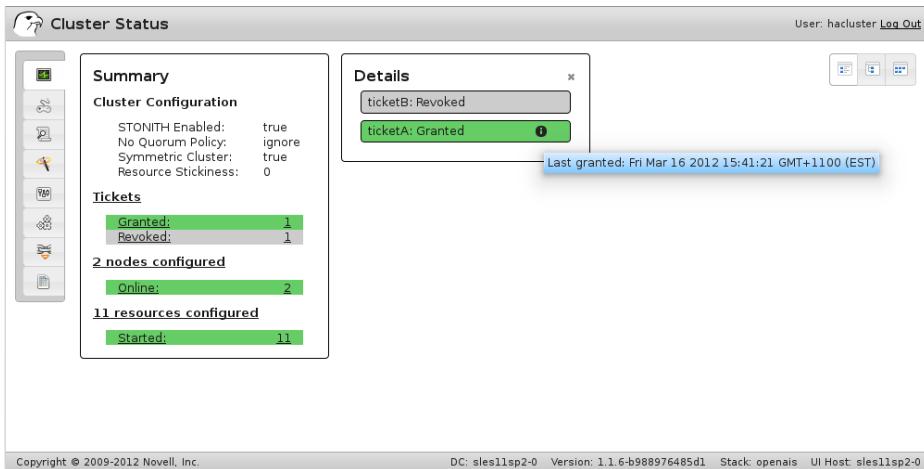
Procedure 6.26: Viewing Tickets with Hawk

Tickets are visible in Hawk if they have been granted or revoked at least once or if they are referenced in a ticket dependency—see Procedure 6.27, “Configuring Ticket Depen-

dencies” (page 146). In case a ticket is referenced in a ticket dependency, but has not been granted to any site yet, Hawk displays it as `revoked`.

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 In the left navigation bar, select *Cluster Status*.
- 3 If the *Summary View* is not already active, click the respective view icon on the upper right-hand side. Along with information about cluster nodes and resources, Hawk also displays a *Ticket* category.
- 4 For more details, either click the title of the *Ticket* category or the individual ticket entries that are marked as links. Hawk displays the ticket's name and, in a tooltip, the last time the ticket has been granted to the current site.

Figure 6.14: Hawk Cluster Status (*Summary View*)—Ticket Details



NOTE: Managing Tickets

To grant or revoke tickets, use the `booth client` command as described in Section 13.5, “Managing Multi-Site Clusters” (page 224). As managing tickets takes place on an “inter-cluster” layer, you cannot do so with Hawk.

6.5.2 Configuring Additional Cluster Resources and Constraints

Apart from the resources and constraints that you need to define for your specific cluster setup, multi-site clusters require additional resources and constraints as outlined in Section 13.4.1, “Configuring Cluster Resources and Constraints” (page 219). All of those can be configured with the CRM shell, but alternatively also with Hawk.

This section focuses on ticket dependencies only as they are specific to multi-site clusters. For general instructions on how to configure resource groups and order constraints with Hawk, refer to Section 6.3.10, “Configuring a Cluster Resource Group” (page 131) and Procedure 6.8, “Adding or Modifying Colocational or Order Constraints” (page 123), respectively.

Procedure 6.27: Configuring Ticket Dependencies

For multi-site clusters, you can specify which resources depend on a certain ticket. Together with this special type of constraint, you can set a `loss-policy` that defines what should happen to the respective resources if the ticket is revoked. The attribute `loss-policy` can have the following values:

- `fence`: Fence the nodes that are running the relevant resources.
- `stop`: Stop the relevant resources.
- `freeze`: Do nothing to the relevant resources.
- `demote`: Demote relevant resources that are running in master mode to slave mode.

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 In the left navigation bar, select *Constraints*. The *Constraints* screen shows categories for all types of constraints and lists all defined constraints.
- 3 To add a new ticket dependency, click the plus icon in the *Ticket* category.

To modify an existing constraint, click the wrench icon next to the constraint and select *Edit Constraint*.

- 4 Enter a unique *Constraint ID*. When modifying existing constraints, the ID is already defined.
- 5 Set a *Loss Policy*.
- 6 Enter the ID of the ticket that the resources should depend on.
- 7 Select a resource from the list *Add resource to constraint*. The list shows the IDs of all resources and all resource templates configured for the cluster.
- 8 To add the selected resource, click the plus icon next to the list. A new list appears beneath, showing the remaining resources. Add as many resources to the constraint as you would like to depend on the ticket.

Figure 6.15: Hawk—Example Ticket Dependency

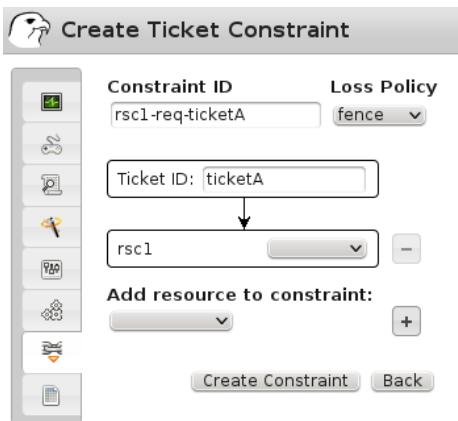


Figure 6.15, “Hawk—Example Ticket Dependency” (page 147) shows a constraint with the ID `rscl-req-ticketA`. It defines that the resource `rsc1` depends on `ticketA` and that the node running the resource should be fenced in case `ticketA` is revoked.

- 9 Click *Create Constraint* to finish the configuration. A message at the top of the screen shows if the constraint was successfully created.

6.5.3 Testing the Impact of Ticket Failover

Hawk's *Simulator* allows you to explore failure scenarios before they happen. To explore if your resources that depend on a certain ticket behave as expected, you can also test the impact of granting or revoking tickets.

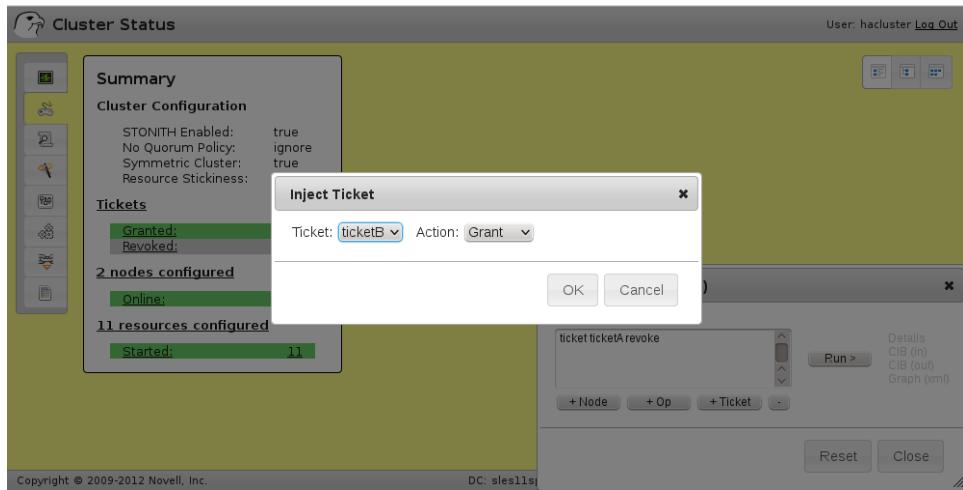
Procedure 6.28: *Simulating Granting and Revoking Tickets*

- 1 Start a Web browser and log in to the cluster as described in Section 6.1.1, “Starting Hawk and Logging In” (page 110).
- 2 In the left navigation bar, select *Simulator*.

Hawk's background changes color to indicate the simulator is active. A simulator dialog opens in the bottom right hand corner of the screen. Its title *Simulator (initial state)* indicates that *Cluster Status* screen still reflects the current state of the cluster.

- 3 To simulate status change of a ticket:
 - 3a Click *+Ticket* in the simulator control dialog.
 - 3b Select the *Action* you want to simulate.
 - 3c Confirm your changes to add them to the queue of events listed in the controller dialog below *Injected State*.
- 4 To start the simulation, click *Run* in the simulator control dialog. The *Cluster Status* screen displays the impact of the simulated events. The simulator control dialog changes to *Simulator (final state)*.
- 5 To exit the simulation mode, close the simulator control dialog. The *Cluster Status* screen switches back to its normal color and displays the current cluster state.

Figure 6.16: HawkSimulator—Tickets



For more information about Hawk's *Simulator* (and which other scenarios can be explored with it), refer to Section 6.5.3, "Testing the Impact of Ticket Failover" (page 148).

6.6 Troubleshooting

Hawk Log Files

Find the Hawk log files in `/srv/www/hawk/log`. Check these files in case you cannot access Hawk.

If you have trouble starting or stopping a resource with Hawk, check the Pacemaker log messages. By default, Pacemaker logs to `/var/log/messages`.

Authentication Fails

If you cannot log in to Hawk with a new user that is a member of the `haclient` group (or if you experience delays until Hawk accepts logins from this user), stop the `nscd` daemon with `rcnscd stop` and try again.

Replacing the Self-Signed Certificate

To avoid the warning about the self-signed certificate on first Hawk startup, 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 certificatefile > /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
```

Login to Hawk Fails After Using History Explorer/`hb_report`

Depending on the period of time you defined in the *History Explorer* or `hb_report` and the events that took place in the cluster during this time, Hawk might collect an extensive amount of information stored in log files in the `/tmp` directory. This might consume the remaining free disk space on your node. In case Hawk should not respond after using the *History Explorer* or `hb_report`, check the hard disk of your cluster node and remove the respective log files.

Configuring and Managing Cluster Resources (Command Line)

7

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 75).

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 fallback nodes, configuring resource monitoring, starting, cleaning up or removing resources, and migrating resources manually.

NOTE: User Privileges

Sufficient privileges are necessary to manage a cluster. The `crm` command 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 `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` so that `sudo` does not ask for a password.

7.1 `crm` Shell—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** Concatenate 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 `crm` Shell Script** Use `crm` and its commands in a script. 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  
statusnode 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`.

You can enter the level directly by typing `crm` and the respective subcommand(s) without any options and hit Enter.

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

In addition to previously explained methods, the `crm` shell also supports synchronous command execution. Use the `-w` option to activate it. If you have started `crm` without `-w`, you can enable it later with the user preference's `wait` set to `yes` (`options wait yes`). If this option is enabled, `crm` waits until the transition is finished. Whenever a transaction is started, dots are printed to indicate progress. Synchronous command execution is only applicable for commands like `resource start`.

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.

7.1.1 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 52)):

```
# 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
AoEtarger          AudibleAlarm           CTDB                ClusterMon
Delay              Dummy                 EvmsSCC            Evmssd
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 427).

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.

7.1.2 Using Configuration Templates

Configuration templates are ready-made cluster configurations for the `crm` shell. Do not confuse them with the *resource templates* (as described in Section 7.3.2, “Creating Resource Templates” (page 160)). Those are templates for the *cluster* and not for the `crm` shell.

Configuration templates 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` and start the `crm` tool:

```
# crm configure
```

- 2 Create a new configuration from a configuration template:

- 2a Switch to the `template` subcommand:

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

- 2b List the available configuration templates:

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

- 2c Decide which configuration 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
```

- 3 Define your parameters:

- 3a List the just created configuration:

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

- 3b** 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
```

- 3c** Invoke your preferred text editor and fill out all lines that have been displayed as errors in Step 3b (page 156):

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

- 4** Show the configuration and check whether it is valid (bold text depends on the configuration you have entered in Step 3c (page 156)):

```
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
```

- 5** Apply the configuration:

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

- 6** 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 configuration template. It does not apply nor commit it to the CIB.

7.1.3 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 Log in as `root` and start the `crm` tool:

```
# crm configure
```

- 2 Create a new shadow configuration:

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

- 3 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.

- 4 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)# commit
```

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

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

7.1.4 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
```

7.2 Configuring Global Cluster Options

Global cluster options control how the cluster behaves when confronted with certain situations. 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:

Procedure 7.1: Modifying Global Cluster Options With `crm`

- 1 Log in as `root` and start the `crm` tool:

```
# crm configure
```

- 2 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
```

- 3 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"
```

- 4 Commit your changes and exit:

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

7.3 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 54).

7.3.1 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 52)). To create a cluster resource use the `crm` tool. To add a new resource to the cluster, proceed as follows:

- 1 Log in as `root` and start the `crm` tool:

```
# crm configure
```

- 2 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.

3 Display and review the changes you have made:

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

4 Commit your changes to take effect:

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

7.3.2 Creating Resource Templates

If you want to create several resources with similar configurations, a resource template simplifies the task. See also Section 4.4.3, “Resource Templates and Constraints” (page 68) for some basic background information. Do not confuse them with the “normal” templates from Section 7.1.2, “Using Configuration Templates” (page 155). Use the `rsc_template` command to get familiar with the syntax:

```
# crm configure rsc_template
usage: rsc_template <name> [<class>:[<provider>:]]<type>
      [params <param>=<value> [<param>=<value>...]]
      [meta <attribute>=<value> [<attribute>=<value>...]]
      [utilization <attribute>=<value> [<attribute>=<value>...]]
      [operations id_spec
          [op op_type [<attribute>=<value>...] ...]]]
```

For example, the following command creates a new resource template with the name `BigVM` derived from the `ocf:heartbeat:Xen` resource and some default values and operations:

```
crm(live)configure# rsc_template BigVM ocf:heartbeat:Xen \
  params allow_mem_management="true" \
  op monitor timeout=60s interval=15s \
  op stop timeout=10m \
  op start timeout=10m
```

Once you defined the new resource template, you can use it in primitives or reference it in order, colocation, or `rsc_ticket` constraints. To reference the resource template, use the `@` sign:

```
crm(live)configure# primitive MyVM1 @BigVM \
  params xmfile="/etc/xen/shared-vm/MyVM1" name="MyVM1"
```

The new primitive MyVM1 is going to inherit everything from the BigVM resource templates. For example, the equivalent of the above two would be:

```
crm(live)configure# primitive MyVM1 ocf:heartbeat:Xen \
    params xmfile="/etc/xen/shared-vm/MyVM1" name="MyVM1"
    params allow_mem_management="true" \
    op monitor timeout=60s interval=15s \
    op stop timeout=10m \
    op start timeout=10m
```

If you want to overwrite some options or operations, add them to your (primitive) definition. For instance, the following new primitive MyVM2 doubles the timeout for monitor operations but leaves others untouched:

```
crm(live)configure# primitive MyVM2 @BigVM \
    params xmfile="/etc/xen/shared-vm/MyVM2" name="MyVM2" \
    op monitor timeout=120s interval=30s
```

A resource template may be referenced in constraints to stand for all primitives which are derived from that template. This helps to produce a more concise and clear cluster configuration. Resource template references are allowed in all constraints except location constraints. Colocation constraints may not contain more than one template reference.

7.3.3 Creating a STONITH Resource

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

- 1 Log in as `root` and start the `crm` tool:

```
# crm configure
```

- 2 Get a list of all STONITH types with the following command:

```
crm(live)# ra list stonith
apcmaster          apcmastersnmp          apcsmart
baytech            bladehpi              cyclades
drac3              external/drac5           external/dracmc-telnet
external/hetzner   external/hmchttp        external/ibmrss
external/ibmrss-telnet  external/ipmi        external/ippower9258
external/kdumpcheck external/libvirt        external/nut
external/rackpdu   external/riloe          external/sbd
external/vcenter   external/vmware         external/xeno
external/xen0-ha   fence_legacy          ibmhmc
ipmilan           meatware             nw_rpc100s
```

| | | |
|------------|---------|---------|
| rcd_serial | rps10 | suicide |
| wti_mpc | wti_nps | |

- 3 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.
...

```

- 4 Create the STONITH resource with the `stonith` class, the type you have chosen in Step 3, 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
```

7.3.4 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 66).

7.3.4.1 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
```

7.3.4.2 Colocational Constraints

The `colocation` 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 colocation 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 run 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.

7.3.4.3 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
```

7.3.4.4 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:start
```

7.3.5 Specifying Resource Failover Nodes

To determine a resource failover, use the meta attribute `migration-threshold`. In case `failcount` exceeds `migration-threshold` on all nodes, the resource will remain stopped. 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 \geq 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 4.4.4, “Failover Nodes” (page 69).

7.3.6 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 4.4.5, “Failback Nodes” (page 70).

7.3.7 Configuring Placement of Resources Based on Load Impact

Some resources may have specific capacity requirements such as minimum amount of memory. Otherwise, they may fail to start completely or run with degraded performance.

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 4.4.6, “Placing Resources Based on Their Load Impact” (page 71).

To configure the resource's requirements and the capacity a node provides, use utilization attributes as described in Procedure 5.10, “Adding Or Modifying Utilization Attributes” (page 92). 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 7.2: Adding Or Modifying Utilization Attributes With `crm`

- 1 Log in as `root` and start the `crm` tool:

```
# 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 thecrm 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.

7.3.8 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 65).

7.3.9 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 4.2.5.1, “Groups” (page 56).

7.3.10 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 4.2.5.2, “Clones” (page 57).

7.3.10.1 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** Log in as `root` and start the `crm` tool:

```
# crm configure
```

- 2** Configure the primitive, for example:

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

- 3** Clone the primitive:

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

7.3.10.2 Creating Stateful/Multi-State Clone Resources

To create an stateful clone resource, first create a primitive resource and then the master/slave resource. The master/slave resource must support at least promote and demote operations.

- 1 Log in as `root` and start the `crm` tool:

```
# crm configure
```

- 2 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
```

- 3 Create the master slave resource:

```
crm(live)configure# ms myRSC-clone myRSC
```

7.4 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.

7.4.1 Starting a New Cluster Resource

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

- 1 Log in as `root` and start the `crm` tool:

```
# crm configure
```

- 2 Switch to the resource level:

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

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

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

7.4.2 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 Remove the resource:

```
crm resource cleanup dlm-clvm
```

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

7.4.3 Removing a Cluster Resource

Proceed as follows to remove a cluster resource:

- 1 Log in as `root` and start the `crm` tool:

```
# crm configure
```

- 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
```

7.4.4 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 move a resource to another node in the cluster using either the Pacemaker GUI or the command line.

Use the `migrate` command for this task. For example, to migrate the resource `ipaddress1` to a cluster node named `node2`, use these commands:

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

7.5 Setting Passwords Independent of cib.xml

In case your cluster configuration contains sensitive information, such as passwords, it should be stored in local files. That way, these parameters will never be logged or leaked in support reports.

Before using `secret`, better run the `show` command first to get an overview of all your resources:

```
# crm configure show
primitive mydb ocf:heartbeat:mysql \
    params replication_user=admin ...
```

If you want to set a password for the above `mydb` resource, use the following commands:

```
#crm resource secret mydb set passwd linux  
INFO: syncing /var/lib/heartbeat/lrm/secrets/mydb/passwd to [your node list]
```

You can get the saved password back with:

```
#crm resource secret mydb show passwd  
linux
```

Note that the parameters need to be synchronized between nodes; the `crm resource secret` command will take care of that. We highly recommend to only use this command to manage secret parameters.

7.6 Retrieving History Information

Investigating the cluster history is a complex task. To simplify this task, the `crm` shell contains the `history` command with its subcommands. It is assumed SSH is configured correctly.

Each cluster moves states, migrates resources, or starts important processes. All these actions can be retrieved by subcommands of `history`. Alternatively, use Hawk as explained in Procedure 6.23, “Viewing Transitions with the History Explorer” (page 140).

By default, all `history` commands look at the events of the last hour. To change this time frame, use the `limit` subcommand. The syntax is:

```
#crm history  
crm(live)history# limit FROM_TIME [TO_TIME]
```

Some valid examples include:

```
limit 4:00pm , limit 16:00
```

Both commands mean the same, today at 4pm.

```
limit 2012/01/12 6pm
```

January 12th 2012 at 6pm

```
limit "Sun 5 20:46"
```

In the current year of the current month at Sunday the 5th at 8:46pm

Find more examples and how to create time frames at <http://labix.org/python-dateutil>.

The `info` subcommand shows all the parameters which are covered by the the `hb_report`:

```
crm(live)history# info
Source: live
Period: 2012-01-12 14:10:56 - end
Nodes: earth
Groups:
Resources:
```

To limit `hb_report` to certain parameters view the available options with the subcommand `help`.

To narrow down the level of detail, use the subcommand `detail` with a level:

```
crm(live)history# detail 2
```

The higher the number, the more detailed your report will be. Default is 0 (zero).

After you have set above parameters, use `log` to show the log messages.

To display the last transition, use the following command:

```
crm(live)history# transition -1
INFO: fetching new logs, please wait ...
```

This command fetches the logs and runs `dotty` (from the `graphviz` package) to show the transition graph. The shell opens the log file which you can browse with the ↓ and ↑ cursor keys.

If you do not want to open the transition graph, use the `nograph` option:

```
crm(live)history# transition -1 nograph
```

7.7 For More Information

- The `crm` man page.
- See Highly Available NFS Storage with DRBD and Pacemaker ([↑Highly Available NFS Storage with DRBD and Pacemaker](#)) for an exhaustive example.

Adding or Modifying Resource Agents

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 `crm ra list stonith`.

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 `cluster-glue` 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 52) for more information. Each resource agent must support 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 configuration

`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:configureprimitive 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 | N/A |

| OCF Re-turn Code | OCF Alias | Description | Recov-ery Type |
|-------------------------|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| | | on what is the expected resource status. If unexpected, then <code>soft</code> recovery. | |
| 8 | <code>OCF_RUNNING_MASTER</code> | The resource is running in Master mode. | <code>soft</code> |
| 9 | <code>OCF_FAILED_MASTER</code> | The resource is in Master mode but has failed. The resource will be demoted, stopped and then started (and possibly promoted) again. | <code>soft</code> |
| other | N/A | Custom error code. | <code>soft</code> |

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 except node1”, because an HA cluster must make sure that every resource may be started on only 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.

When the state of a 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.

Resource level fencing can be achieved by 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 it 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 simple if brutal way: reset or power off the node.

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 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 may even become standard in 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 while 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` 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 52)). 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 loading and unloading the STONITH device driver on a node. 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 logging it to the device (to verify that the device will work in case it is needed). When a STONITH resource fails over to another node it enables the current node to talk to the STONITH device by loading the respective driver.

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 83), Section 6.3.3, “Creating STONITH Resources” (page 119), or Section 7.3.3, “Creating a STONITH Resource” (page 161).

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 7, *Configuring and Managing Cluster Resources (Command Line)* (page 151).

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 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=PASSWORD
primitive st-ibmrsa-2 stonith:external/ibmrsa-telnet \
params nodename=node2 ipaddr=192.168.0.102 \
userid=USERID passwd=PASSWORD
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 for 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, 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 fencing device is similar to the examples above. The details are not covered here. All UPS devices employ the same mechanics for fencing. How the device is accessed varies. Old UPS devices only had 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 use a USB or Ethernet interface. The kind of connection you can use depends 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

As explained in Section 9.3.1, “Example STONITH Resource Configurations” (page 185), 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.

If clones are safe to use with your configuration and 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 for checking status.

NOTE: Monitoring STONITH Resources

Monitor STONITH resources regularly, yet sparingly. For most devices a monitoring interval of at least 1800 seconds (30 minutes) should suffice.

Fencing devices are an indispensable part of an HA cluster, but the less you need to use them, the better. Power management equipment is often affected by too much broadcast traffic. Some devices 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 status of fencing devices once every few hours should be enough in most cases. The probability that a fencing operation needs to be performed and the power switch fails is low.

For detailed information on how to configure monitor operations, refer to Procedure 5.3, “Adding or Modifying Meta and Instance Attributes” (page 82) for the GUI approach or to Section 7.3.8, “Configuring Resource Monitoring” (page 168) for the command line approach.

9.5 Special Fencing Devices

In addition to plug-ins which handle real STONITH devices, there are special purpose STONITH plug-ins.

WARNING: For Testing Only

Some of the STONITH plug-ins mentioned below are for demonstration and testing purposes only. Do not use any of the 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 checks if a kernel dump is in progress on a node. If so, it returns `true`, and acts as if the node has been fenced. The node cannot run any resources during the dump anyway. 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 stops the node from operating. Learn how to use this STONITH agent to implement storage-based fencing in Chapter 17, *Storage Protection* (page 269). See also http://www.linux-ha.org/wiki/SBD_Fencing for more details.

IMPORTANT: `external/sbd` and DRBD

The `external/sbd` fencing mechanism requires that the SBD partition is readable directly from each node. Thus, a DRBD* device must not be used for an SBD partition.

However, you can use the fencing mechanism for a DRBD cluster, provided the SBD partition is located on a shared disk that is not mirrored or replicated.

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 the user 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 issues a `meatclient(8)` command. This tells `meatware` to inform the cluster that the node should be considered dead. See `/usr/share/doc/packages/cluster-glue/README.meatware` for more information.

null

This is a fake device used in various testing scenarios. It always 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 “I do not shoot my host” rule.

9.6 Basic Recommendations

Check the following list of recommendations to avoid common mistakes:

- Do not configure several power switches in parallel.

- To test your STONITH devices and their configuration, pull the plug once from each node and verify that fencing the node does takes place.
- Test your resources under load and verify the timeout values are appropriate. Setting timeout values too low can trigger (unnecessary) fencing operations. For details, refer to Section 4.2.9, “Timeout Values” (page 64).
- Use appropriate fencing devices for your setup. For details, also refer to Section 9.5, “Special Fencing Devices” (page 189).
- Configure one ore more STONITH resources. By default, the global cluster option `stonith-enabled` is set to `true`. If no STONITH resources have been defined, the cluster will refuse to start any resources.
- Do not set the global cluster option `stonith-enabled` to `false` for the following reasons:
 - Clusters without STONITH enabled are not supported.
 - DLM/OCFS2 will block forever waiting for a fencing operation that will never happen.
- Do not set the global cluster option `startup-fencing` to `false`. By default, it is set to `true` for the following reason: If a node is in an unknown state during cluster startup, the node will be fenced once to clarify its status.

9.7 For More Information

`/usr/share/doc/packages/cluster-glue`

In your installed system, this directory contains 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/>”

- *Fencing and Stonith*: Information about fencing on the home page of the Pacemaker Project.
- *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.

10

Access Control Lists

The various tools for administrating clusters, like the `crm shell`, Hawk, or the Pacemaker GUI, can be used by `root` or any user in the group `haclient`. By default, these users have full read-write access. In some cases, you may want to limit access or assign more fine-grained access rights.

Optional *Access control lists* (ACLs) allow you to define rules for users in the `haclient` group to allow or deny access to any part of the cluster configuration. Typically, sets or rules are combined into roles. Then you can assign users to a role that fits their tasks.

10.1 Requirements and Prerequisites

Before you start using ACLs on your cluster, make sure the following conditions are fulfilled:

- The same users must be available on all nodes in your cluster. Use NIS to ensure this.
- All users must belong to the `haclient` group.
- All users have to run the `crm` shell by its absolute path `/usr/sbin/crm`.

Note the following points:

- ACLs are an optional feature. If the ACL feature is disabled, `root` and users belonging to the `haclient` group have full read/write access to the cluster configuration.

- If you want to enable the ACL feature, use this command:

```
crm configure property enable-acl=true
```

- If non-privileged users want to run the `crm` shell, they have to change the `PATH` variable and extend it with `/usr/sbin`.
- To use ACLs you need some knowledge about XPath. XPath is a language for selecting nodes in an XML document. Refer to <http://en.wikipedia.org/wiki/XPath> or look into the specification at <http://www.w3.org/TR/xpath/>.

10.2 The Basics of ACLs

An ACL role is a set of rules which describe access rights to CIB. Rules consist of:

- access rights to read, write, or deny, and
- a specification where to apply the rule. This specification can be a tag, an id reference, a combination of both, or an XPath expression.

In most cases, it is more convenient to bundle ACLs into roles and assign a role to a user. However, it is possible to give a user certain access rules without defining any roles.

There are two methods to manage ACL rules:

- **Via an XPath Expression** You need to know the structure of the underlying XML to create ACL rules.
- **Via a Tag and/or Ref Abbreviation** Create a shorthand syntax and ACL rules apply to the matched objects.

10.2.1 Setting ACL Rules via XPath

To manage ACL rules via XPath, you need to know the structure of the underlying XML. Retrieve the structure with the following command:

```
crm configure show xml
```

The XML structure can also be displayed in the Pacemaker GUI by selecting *Configuration > View XML*. Regardless of the tool, the output is your cluster configuration in XML (see Example 10.1, “Excerpt of a Cluster Configuration in XML” (page 195)).

Example 10.1: *Excerpt of a Cluster Configuration in XML*

```
<cib admin_epoch="0"
      cib-last-written="Wed Nov  2 16:42:51 2011"
      crm_feature_set="3.0.5"
      dc-uuid="stuttgart"
      epoch="13" have-quorum="1" num_updates="42"
      update-client="cibadmin"
      update-origin="nuernberg"
      update-user="root" validate-with="pacemaker-1.2">
<configuration>
  <crm_config>
    <cluster_property_set id="cib-bootstrap-options">
      <nvpair id="cib-bootstrap-options-stonith-enabled"
              name="stonith-enabled" value="true"/>
    </cluster_property_set>
  </crm_config>
  <nodes>
    <node id="stuttgart" type="normal" uname="stuttgart"/>
    <node id="nuernberg" type="normal" uname="nuernberg"/>
  </nodes>
  <resources> ... </resources>
  <constraints/>
  <rsc_defaults> ... </rsc_defaults>
  <op_defaults> ... </op_defaults>
<configuration>
</cib>
```

With the XPath language you can locate nodes in this XML document. For example, to select the root node (`cib`) use the XPath expression `/cib`. To locate the global cluster configurations, use the XPath `/cib/configuration/crm_config`.

The following table collects the access type and the XPath expression to create an “operator” role:

Table 10.1: Types and XPath Expression for an Operator Role

| Type | XPath/Explanation |
|-------|--------------------------------------------------------------------------------------------------|
| Write | //cm_config//nvpair[@name='maintenance-mode'] Turn maintenance mode on or off. |
| Write | //op_defaults//nvpair[@name='record-pending'] Choose whether pending operations are recorded. |
| Write | //nodes/node//nvpair[@name='standby'] Set node in online or standby mode. |
| Write | //resources//nvpair[@name='target-role'] Start, stop, promote or demote any resource. |
| Write | //resources//nvpair[@name='is-managed'] Select if a resource should be managed or not. |
| Write | //constraints/rsc_location Migrate/move resources from one node to another. |
| Read | /cib View the status of the cluster. |

10.2.2 Setting ACL Rules via Tag Abbreviations

For users who do not want to deal with the XML structure there is an easier method. It is a combination of a tag specifier and/or a reference.

For example, consider the following XPath:

```
/cib/resources/primitive[@id='rsc1']
```

`primitive` is a resource with the reference `rsc1`. The abbreviated syntax is:

```
tag: "primitive" ref:"rsc1"
```

This also works for constraints. Here is the verbose XPath:

```
/cib/constraint/rsc_location
```

The abbreviated syntax is written like this:

```
tag: "rsc_location"
```

The CIB daemon knows how to apply the ACL rules to the matched objects. The abbreviated syntax can be used in the `crm` Shell or the Pacemaker GUI.

10.3 Configuring ACLs with the Pacemaker GUI

Use the Pacemaker GUI to define your roles and users. The following procedure adds a “monitor” role which has only read access to the CIB. Proceed as follows:

Procedure 10.1: *Adding a Monitor Role and Assigning a User with the Pacemaker GUI*

- 1 Start the Pacemaker GUI and log in as described in Section 5.1.1, “Logging in to a Cluster” (page 76).
- 2 Click the *ACLs* entry in the *Configuration* tree.
- 3 Click *Add*. A dialog box appears. Choose between *ACL User* and *ACL Role*.
- 4 To define your ACL role(s):
 - 4a Choose *ACL Role*. A window opens in which you add your configuration options.
 - 4b Add a unique identifier in the *ID* textfield, for example `monitor`.

- 4c** Click *Add* and choose the rights (*Read*, *Write*, or *Deny*). In our example, select *Read* and proceed with *Ok*.
- 4d** Enter the XPath expression `/cib` in the *Xpath* textfield. Proceed with *Ok*.

Sidenote: If you have resources or constraints, you can also use the abbreviated syntax as explained in Section 10.2.2, “Setting ACL Rules via Tag Abbreviations” (page 196). In this case, enter your tag in the *Tag* textfield and the optional reference in the *Ref* textfield. In our example, there is no abbreviated form possible, so you can only use the XPath notation here.

- 4e** If you have other conditions, repeat the steps (Step 4c (page 198) and Step 4d (page 198)). In our example, this is not the case so your role is finished and you can close the window with *Ok*.

5 Assign your role to a user:

- 5a** Click the *Add* button. A dialog box appears to choose between *ACL User* and *ACL Role*.
- 5b** Choose *ACL User*. A window opens in which you add your configuration options.
- 5c** Enter the username in the *ID* textfield. Make sure this user belongs to the `haclient` group.
- 5d** Click *Add* and choose *Role Ref*.
- 5e** Use the role name specified in Step 4b (page 197).

10.4 Configuring ACLs with the `crm` Shell

The following procedure adds a “monitor” role as shown in Section 10.3, “Configuring ACLs with the Pacemaker GUI” (page 197) and assigns it to a user. Proceed as follows:

Procedure 10.2: Adding a Monitor Role and Assigning a User with the `crm` Shell

1 Log in as `root`.

2 Start the interactive mode of the `crm` shell:

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

3 Define your ACL role(s):

3a Use the `role` command to define your new role. To define a “monitor” role, use the following command:

```
role monitor read xpath:"/cib"
```

The previous command creates a new role with name `monitor`, sets the `read` rights and applies it to all elements in the CIB by using the XPath `/cib`. If necessary, you can add more access rights and XPath arguments.

Sidenote: If you have resources or constraints, you can also use the abbreviated syntax as explained in Section 10.2.2, “Setting ACL Rules via Tag Abbreviations” (page 196). If you have a primitive resource with the ID `rsc1`, use the following notation to set the access rights: `write tag:"primitive" ref:"rsc1"`. You can also refer to the ID with `write ref:"rsc1"`. This has the advantage that it can match a primitive resource and a local resource manager resource (LRM), which enables you to configure `rsc1` and also cleanup its status at the same time.

3b Add additional roles as needed.

4 Assign your roles to users. Make sure this user belongs to the `haclient` group.

```
crm(live)configure# user tux role:monitor
```

5 Check your changes:

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

6 Commit your changes:

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

10.5 For More Information

See <http://www.clusterlabs.org/doc/acls.html>.

Network Device Bonding

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.

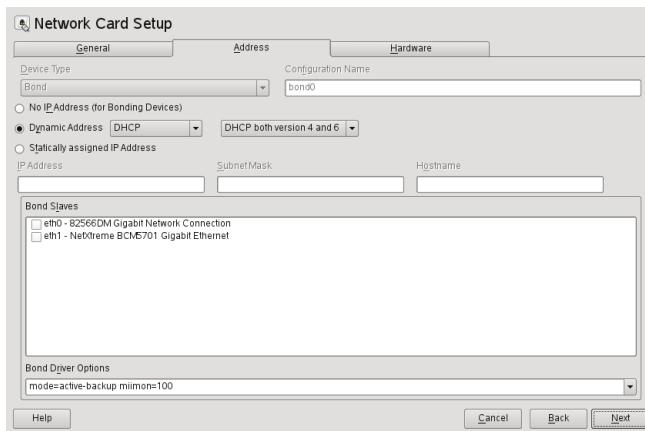
The configuration of the bonding device is done by means of bonding module options. The behavior is determined through the mode of the bonding device. By default, this is mode=active-backup, which means that a different slave device will become active if the active slave fails.

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*.



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 to add the parameter `miimon=100` 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`.

Load Balancing with Linux Virtual Server

12

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.

12.1 Conceptual Overview

The following sections give an overview of the main LVS components and concepts.

12.1.1 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.

12.1.2 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 `ldirectord` 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 list of available servers 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 list of available servers. In 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.

12.1.3 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.

12.1.4 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.

12.2 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 12.1, “Simple `ldirectord` Configuration” (page 212).

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.

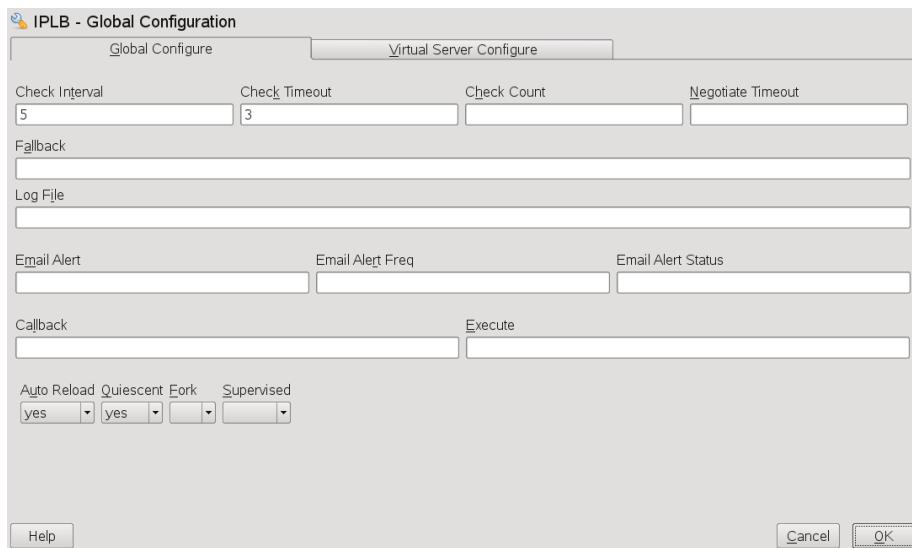
Procedure 12.1: Configuring Global Parameters

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 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 12.1: YaST IP Load Balancing—Global Parameters



Procedure 12.2: 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 (IPv4 or IPv6) and port under which the load balancers and the real servers are accessible as LVS. Instead of IP address and port number 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 addresses (IPv4, IPv6, or hostnames) of the servers, the ports (or service names) and the forwarding method. The forwarding method must either be `gate`, `ipip` or `masq`, see Section 12.1.3, “Packet Forwarding” (page 207).

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 211), 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 12.2: YaST IP Load Balancing—Virtual Services

The screenshot shows the 'IPLB - Virtual Servers Configuration' dialog. At the top, there's a 'Virtual Server' field containing '192.168.0.200:80'. Below it is a 'Real Servers' section with two entries: '192.168.0.110:80' (selected) and '192.168.0.120:80'. To the right of these are 'Add', 'Delete', and 'Edit' buttons. The main configuration area has several tabs: 'Check Type' (set to 'negotiate'), 'Service' (set to 'http'), 'Check Command' (empty), 'Check Port' (empty). Under 'Request' and 'Receive' fields, 'test.html' and 'still alive' are entered respectively. 'Http Method' and 'Virtual Host' dropdowns are shown. Other tabs include 'Login', 'Password', 'Database Name', 'Radius Secret', 'Persistent', 'Netmask', 'Scheduler Protocol' (set to 'wlc'), 'Check Timeout', 'Negotiate Timeout', 'Failure Count', 'Email Alert', 'Email Alert Freq', 'Email Alert Status' (dropdown set to 'fallback'), 'Fallback' (set to '127.0.0.1:80'), and 'Quiescent'. At the bottom are 'Help', 'Cancel', and 'OK' buttons.

Example 12.1: Simple ldirectord Configuration

The values shown in Figure 12.1, “YaST IP Load Balancing—Global Parameters” (page 210) and Figure 12.2, “YaST IP Load Balancing—Virtual Services” (page 212), would lead to the following configuration, defined in `/etc/ha.d/ldirectord.cf`:

```
autoreload = yes ①
checkinterval = 5 ②
checktimeout = 3 ③
quiescent = yes ④
virtual = 192.168.0.200:80 ⑤
checktype = negotiate ⑥
fallback = 127.0.0.1:80 ⑦
protocol = tcp ⑧
real = 192.168.0.110:80 gate ⑨
real = 192.168.0.120:80 gate ⑩
receive = "still alive" ⑪
request = "test.html" ⑫
scheduler = wlc ⑬
service = http ⑭
```

- 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.
- ⓫ URI to the object that is requested on each real server during the check intervals.
- ⓬ Selected scheduler to be used for load balancing.
- ⓭ 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 ❼. If it does not receive the expected `still alive` string ❼ 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.

12.3 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 3.5.5, “Transferring the Configuration to All Nodes” (page 39).

12.4 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.

Multi-Site Clusters (Geo Clusters)

Apart from local clusters and metro area clusters, SUSE® Linux Enterprise High Availability Extension 11 SP2 also supports multi-site clusters (geo clusters). That means you can have multiple, geographically dispersed sites with a local cluster each. Failover between these clusters is coordinated by a higher level entity, the so-called booth. Support for multi-site clusters is available as a separate option to SUSE Linux Enterprise High Availability Extension.

13.1 Challenges for Multi-Site Clusters

Typically, multi-site environments are too far apart to support synchronous communication between the sites and synchronous data replication. That leads to the following challenges:

- How to make sure that a cluster site is up and running?
- How to make sure that resources are only started once?
- How to make sure that quorum can be reached between the different sites and a split brain scenario can be avoided?
- How to manage failover between the sites?
- How to deal with high latency in case of resources that need to be stopped?

In the following sections, learn how to meet these challenges with SUSE Linux Enterprise High Availability Extension.

13.2 Conceptual Overview

Multi-site clusters based on SUSE Linux Enterprise High Availability Extension can be considered as “overlay” clusters where each cluster site corresponds to a cluster node in a traditional cluster. The overlay cluster is managed by the booth mechanism. It guarantees that the cluster resources will be highly available across different cluster sites. This is achieved by using so-called tickets that are treated as failover domain between cluster sites, in case a site should be down.

The following list explains the individual components and mechanisms that were introduced for multi-site clusters in more detail.

Components and Concepts

Ticket

A ticket grants the right to run certain resources on a specific cluster site. A ticket can only be owned by one site at a time. Initially, none of the sites has a ticket—each ticket must be granted once by the cluster administrator. After that, tickets are managed by the booth for automatic failover of resources. But administrators may also intervene and grant or revoke tickets manually.

Resources can be bound to a certain ticket by dependencies. Only if the defined ticket is available at a site, the respective resources are started. Vice versa, if the ticket is removed, the resources depending on that ticket are automatically stopped.

The presence or absence of tickets for a site is stored in the CIB as a cluster status. With regards to a certain ticket, there are only two states for a site: `true` (the site has the ticket) or `false` (the site does not have the ticket). The absence of a certain ticket (during the initial state of the multi-site cluster) is not treated differently from the situation after the ticket has been revoked: both are reflected by the value `false`.

A ticket within an overlay cluster is similar to a resource in a traditional cluster. But in contrast to traditional clusters, tickets are the only type of resource in an overlay cluster. They are primitive resources that do not need to be configured nor cloned.

Booth

The booth is the instance managing the ticket distribution and thus, the failover process between the sites of a multi-site cluster. Each of the participating clusters

and arbitrators runs a service, the `boothd`. It connects to the booth daemons running at the other sites and exchanges connectivity details. Once a ticket is granted to a site, the booth mechanism will manage the ticket automatically: If the site which holds the ticket is out of service, the booth daemons will vote which of the other sites will get the ticket. To protect against brief connection failures, sites that lose the vote (either explicitly or implicitly by being disconnected from the voting body) need to relinquish the ticket after a time-out. Thus, it is made sure that a ticket will only be re-distributed after it has been relinquished by the previous site. See also “Dead Man Dependency (`loss-policy="fence"`)” (page 217).

Arbitrator

Each site runs one booth instance that is responsible for communicating with the other sites. If you have a setup with an even number of sites, you need an additional instance to reach consensus about decisions such as failover of resources across sites. In this case, add one or more arbitrators running at additional sites. Arbitrators are single machines that run a booth instance in a special mode. As all booth instances communicate with each other, arbitrators help to make more reliable decisions about granting or revoking tickets.

An arbitrator is especially important for a two-site scenario: For example, if site A can no longer communicate with site B, there are two possible causes for that:

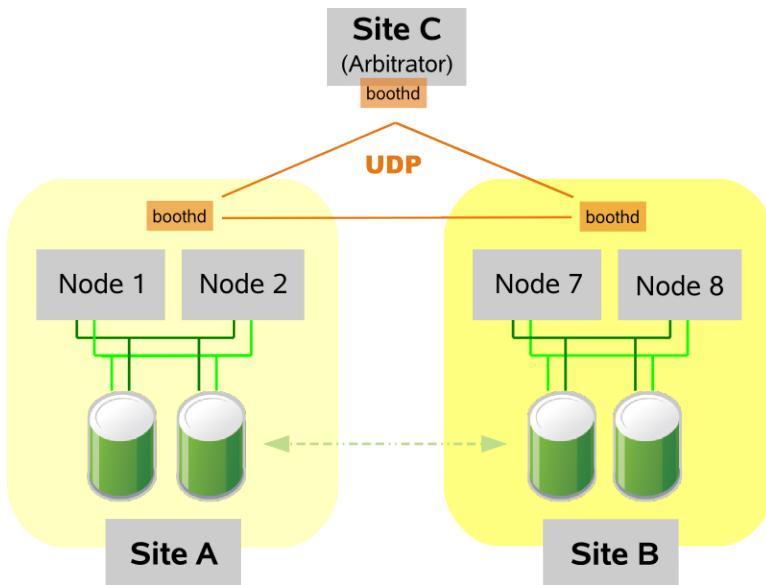
- A network failure between A and B.
- Site B is down.

However, if site C (the arbitrator) can still communicate with site B, site B must still be up and running.

Dead Man Dependency (`loss-policy="fence"`)

After a ticket is revoked, it can take a long time until all resources depending on that ticket are stopped, especially in case of cascaded resources. To cut that process short, the cluster administrator can configure a `loss-policy` (together with the ticket dependencies) for the case that a ticket gets revoked from a site. If the `loss-policy` is set to `fence`, the nodes that are hosting dependent resources are fenced. This considerably speeds up the recovery process of the cluster and makes sure that resources can be migrated more quickly.

Figure 13.1: Example Scenario: A Two-Site Cluster (4 Nodes + Arbitrator)



As usual, the CIB is synchronized within each cluster, but it is not synchronized across cluster sites of a multi-site cluster. You have to configure the resources that will be highly available across the multi-site cluster for every site accordingly.

13.3 Requirements

Software Requirements

- All clusters that will be part of the multi-site cluster must be based on SUSE Linux Enterprise High Availability Extension 11 SP2.
- SUSE® Linux Enterprise Server 11 SP2 must be installed on all arbitrators.
- The `booth` package must be installed on all cluster nodes *and* on all arbitrators that will be part of the multi-site cluster.

The most common scenario is probably a multi-site cluster with two sites and a single arbitrator on a third site. However, technically, there are no limitations with regards to the number of sites and the number of arbitrators involved.

Nodes belonging to the same cluster site should be synchronized via NTP. However, time synchronization is not required between the individual cluster sites.

13.4 Basic Setup

Configuring a multi-site cluster takes the following basic steps:

Configuring Cluster Resources and Constraints (page 219)

1. Configuring Ticket Dependencies (page 219)
2. Configuring a Resource Group for `boothd` (page 221)
3. Adding an Ordering Constraint (page 221)

Setting Up the Booth Services (page 222)

1. Editing The Booth Configuration File (page 222)
2. Starting the Booth Services (page 224)

13.4.1 Configuring Cluster Resources and Constraints

Apart from the resources and constraints that you need to define for your specific cluster setup, multi-site clusters require additional resources and constraints as described below. Instead of configuring them with the CRM shell, you can also do so with the HA Web Konsole. For details, refer to Section 6.5.2, “Configuring Additional Cluster Resources and Constraints” (page 146).

Procedure 13.1: Configuring Ticket Dependencies

The `crm configure rsc_ticket` command lets you specify the resources depending on a certain ticket. Together with the constraint, you can set a `loss-policy` that defines what should happen to the respective resources if the ticket is revoked. The attribute `loss-policy` can have the following values:

- `fence`: Fence the nodes that are running the relevant resources.

- `stop`: Stop the relevant resources.
- `freeze`: Do nothing to the relevant resources.
- `demote`: Demote relevant resources that are running in `master` mode to `slave` mode.

- 1** On one of the cluster nodes, start a shell and log in as `root` or equivalent.
- 2** Enter `crm configure` to switch to the interactive shell.
- 3** Configure a constraint that defines which resources depend on a certain ticket. For example:

```
crm(live)configure#
rsc_ticket rscl-req-ticketA ticketA: rscl loss-policy="fence"
```

This creates a constraint with the ID `rscl-req-ticketA`. It defines that the resource `rscl` depends on `ticketA` and that the node running the resource should be fenced in case `ticketA` is revoked.

If resource `rscl` was not a primitive, but a special clone resource that can run in `master` or `slave` mode, you may want to configure that only `rscl`'s master mode depends on `ticketA`. With the following configuration, `rscl` is automatically demoted to `slave` mode if `ticketA` is revoked:

```
crm(live)configure#
rsc_ticket rscl-req-ticketA ticketA: rscl:Master loss-policy="demote"
```

- 4** If you want other resources to depend on further tickets, create as many constraints as necessary with `rsc_ticket`.
- 5** Review your changes with `show`.
- 6** If everything is correct, submit your changes with `commit` and leave the `crm live` configuration with `exit`.

The constraints are saved to the CIB.

Procedure 13.2: Configuring a Resource Group for boothd

Each site needs to run one instance of `boothd` that communicates with the other booth daemons. The daemon can be started on any node, therefore it should be configured as primitive resource. To make the `boothd` resource stay on the same node, if possible, add resource stickiness to the configuration. As each daemon needs a persistent IP address, configure another primitive with a virtual IP address. Group booth primitives:

- 1** On one of the cluster nodes, start a shell and log in as `root` or equivalent.
- 2** Enter `crm configure` to switch to the interactive shell.
- 3** To create both primitive resources and to add them to one group, `g-booth`:

```
crm(live)configure#
primitive booth-ip ocf:heartbeat:IPAddr2 params ip="IP_ADDRESS"
primitive booth ocf:pacemaker:booth-site \
    meta resource-stickiness="INFINITY" \
    op monitor interval="10s" timeout="20s"
group g-booth booth-ip booth
```

- 4** Review your changes with `show`.
- 5** If everything is correct, submit your changes with `commit` and leave the `crm live` configuration with `exit`.
- 6** Repeat the resource group configuration on the other cluster sites, using a different IP address for each `boothd` resource group.

With this configuration, each booth daemon will be available at its individual IP address, independent of the node the daemon is running on.

Procedure 13.3: Adding an Ordering Constraint

If a ticket has been granted to a site but all nodes of that site should fail to host the `boothd` resource group for any reason, a “split-brain” situation among the geographically dispersed sites could occur. In that case, no `boothd` instance would be available to safely manage fail-over of the ticket to another site. To avoid a potential concurrency violation of the ticket (the ticket is granted to multiple sites simultaneously), add an ordering constraint:

- 1** On one of the cluster nodes, start a shell and log in as `root` or equivalent.

2 Enter `crm configure` to switch to the interactive shell.

3 Create an ordering constraint:

```
crm(live)configure#
order order-booth-rscl inf: g-booth rscl
```

This defines that `rscl` (that depends on `ticketA`) can only be started after the `g-booth` resource group.

In case `rscl` is not a primitive, but a special clone resource and configured as described in Step 3 (page 220), the ordering constraint should be configured as follows:

```
crm(live)configure#
order order-booth-rscl inf: g-booth rscl:promote
```

This defines that `rscl` can only be promoted to master mode after the `g-booth` resource group has started.

4 Review your changes with `show`.

5 For any other resources that depend on a certain ticket, define further ordering constraints.

6 If everything is correct, submit your changes with `commit` and leave the `crm live` configuration with `exit`.

13.4.2 Setting Up the Booth Services

After having configured the resource group for the `boothd` and the ticket dependencies, complete the booth setup:

Procedure 13.4: Editing The Booth Configuration File

1 Log in to a cluster node as `root` or equivalent.

2 Create `/etc/booth/booth.conf` and edit it according to the example below:

Example 13.1: Example Booth Configuration File

```
transport="UDP" ①
port="6666" ②
arbitrator="147.2.207.14" ③
```

```
site="147.4.215.19" ④  
site="147.18.2.1" ④  
ticket="ticketA;⑤1000⑥"  
ticket="ticketB;⑤1000⑥"
```

- ❶ Defines the transport protocol used for communication between the sites. For SP2, only UDP is supported, other transport layers will follow.
- ❷ Defines the port used for communication between the sites. Choose any port that is not already used for different services. Make sure to open the port in the nodes' and arbitrators' firewalls.
- ❸ Defines the IP address of the arbitrator. Insert an entry for each arbitrator you use in your setup.
- ❹ Defines the IP address used for the `boothd` on each site. Make sure to insert the correct virtual IP addresses (`IPAddr2`) for each site, otherwise the booth mechanism will not work correctly.
- ❺ Defines the ticket to be managed by the booth. For each ticket, add a `ticket` entry.
- ❻ Optional parameter. Defines the ticket's expiry time in seconds. A site that has been granted a ticket will renew the ticket regularly. If the booth does not receive any information about renewal of the ticket within the defined expiry time, the ticket will be revoked and granted to another site. If no expiry time is specified, the ticket will expire after 600 seconds by default.

An example booth configuration file is available at `/etc/booth/booth.conf.example`.

- ❸ Verify your changes and save the file.
- ❹ Copy `/etc/booth/booth.conf` to all sites and arbitrators. In case of any changes, make sure to update the file accordingly on all parties.

NOTE: Synchronize Booth Configuration to All Sites and Arbitrators

All cluster nodes and arbitrators within the multi-site cluster must use the same booth configuration. While you may need to copy the files manually to the arbitrators and to one cluster node per site, you can use Csync2 within each cluster site to synchronize the file to all nodes.

Procedure 13.5: Starting the Booth Services

- 1 Start the booth resource group on each other cluster site. It will start one instance of the booth service per site.
- 2 Log in to each arbitrator and start the booth service:

```
/etc/init.d/booth-arbitrator start
```

This starts the booth service in arbitrator mode. It can communicate with all other booth daemons but in contrast to the booth daemons running on the cluster sites, it cannot be granted a ticket.

After finishing the booth configuration and starting the booth services, you are now ready to start the ticket process.

13.5 Managing Multi-Site Clusters

Before the booth can manage a certain ticket within the multi-site cluster, you initially need to grant it to a site manually. Use the `booth client` command line tool to grant, list, or revoke tickets. The `booth client` commands work on any machine where the booth daemon is running.

WARNING: `crm_ticket` and `crm site ticket`

In case the booth service is not running for any reasons, you may also manage tickets manually with `crm_ticket` or `crm site ticket`. Both commands are only available on cluster nodes. In case of manual intervention, use them with great care as they *cannot* verify if the same ticket is already granted elsewhere. For basic information about the commands, refer to their man pages.

As long as booth is up and running, only use `booth client` for manual intervention.

Procedure 13.6: Managing Tickets Manually

- 1 To list all tickets on all sites:

```
booth client list
```

- 2** To grant a ticket to a site (for example, `ticketA` to the site `147.2.207.14`):

```
booth client grant -t ticketA -s 147.2.207.14
```

The command executes a sanity check and will warn you if the same ticket is already granted to another site.

- 3** To revoke a ticket from a site (for example, `ticketA` from the site `147.2.207.14`):

```
booth client revoke -t ticketA -s 147.2.207.14
```

After you have initially granted a ticket to a site, the booth mechanism will take over and manage the ticket automatically. If the site holding a ticket should be out of service, the ticket will automatically be revoked after the expiry time and granted to another site. The resources that depend on that ticket will fail over to the new site holding the ticket. The nodes that have run the resources before will be treated according to the `loss-policy` you set within the constraint.

Part III. Storage and Data Replication

14

OCFS2

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).

14.1 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.

14.2 OCFS2 Packages and Management Utilities

The OCFS2 kernel module (`ocfs2`) is installed automatically in the High Availability Extension on SUSE® Linux Enterprise Server 11 SP2. 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 14.1: 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. |

14.3 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 14.1: 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 extend the base group with the respective primitives as needed. As the base group has internal colocation and ordering, this facilitates the overall setup as you do not have to specify several 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 colocation and ordering, the `o2cb` service is only started on nodes where a copy of the `dlm` service is 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 14.2: *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 Create an SBD partition as described in Section 17.1.3.1, “Creating the SBD Partition” (page 272).
- 3 Run `crm configure`.
- 4 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"
```

- 5 Review your changes with `show`.
- 6 If everything is correct, submit your changes with `commit` and leave the `crm` live configuration with `exit`.

14.4 Creating OCFS2 Volumes

After you have configured DLM and O2CB as cluster resources as described in Section 14.3, “Configuring OCFS2 Services and a STONITH Resource” (page 231), 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 14.3, “Creating and Formatting an OCFS2 Volume” (page 236). The most important parameters for the command are listed in Table 14.2, “Important OCFS2 Parameters” (page 234). For more information and the command syntax, refer to the `mkfs.ocfs2` man page.

Table 14.2: *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 <code>tunefs.ocfs2</code> 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 <code>mkfs.ocfs2</code> 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, |

| OCFS2 Parameter | Description and Recommendation |
|---------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | <p>x86-64, and ia64) and big-endian architectures (such as ppc64 and s390x).</p> <p>Node-specific files are referred to as local files. A node slot number is appended to the local file. For example: <code>journal:0000</code> belongs to whatever node is assigned to slot number 0.</p> <p>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 <code>tunefs.ocfs2</code> utility to increase the number of node slots as needed. Note that the value cannot be decreased.</p> |
| Block Size (-b) | <p>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 <code>mkfs.ocfs2</code> man page.</p> |
| Specific Features On/Off (--fs-features) | <p>A comma separated list of feature flags can be provided, and <code>mkfs.ocfs2</code> 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 <code>no</code> to the name.</p> <p>For an overview of all available flags, refer to the <code>mkfs.ocfs2</code> man page.</p> |

| OCFS2 Parameter | Description and Recommendation |
|----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| 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 <code>mkfs.ocfs2</code> 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 14.3: 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
```

14.5 Mounting OCFS2 Volumes

You can either mount an OCFS2 volume manually or with the cluster manager, as described in Procedure 14.5, “Mounting an OCFS2 Volume with the Cluster Manager” (page 237).

Procedure 14.4: 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 14.5: 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 14.1, “Configuring DLM and O2CB Resources” (page 232):

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 colocation 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 431).
- 6 If everything is correct, submit your changes with `commit` and leave the `crm` live configuration with `exit`.

14.6 Using Quotas on OCFS2 File Systems

To use quotas on an OCFS2 file system, create and mount the files system with the appropriate quota features or mount options, respectively: `usrquota` (quota for individual users) or `grpquota` (quota for groups). These features can also be enabled later on an unmounted file system using `tunefs.ocfs2`.

When a file system has the appropriate quota feature enabled, it tracks in its metadata how much space and files each user (or group) uses. Since OCFS2 treats quota information as file system-internal metadata, you do not need to run the `quotacheck(8)` program. All functionality is built into `fsck.ocfs2` and the file system driver itself.

To enable enforcement of limits imposed on each user or group, run `quotaon(8)` like you would do for any other file system.

For performance reasons each cluster node performs quota accounting locally and synchronizes this information with a common central storage once per 10 seconds. This interval is tunable with `tunefs.ocfs2`, options `usrquota-sync-interval` and `grpquota-sync-interval`. Therefore quota information may not be exact at all times and as a consequence users or groups can slightly exceed their quota limit when operating on several cluster nodes in parallel.

14.7 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)

The *distributed replicated block device* (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.

15.1 Conceptual Overview

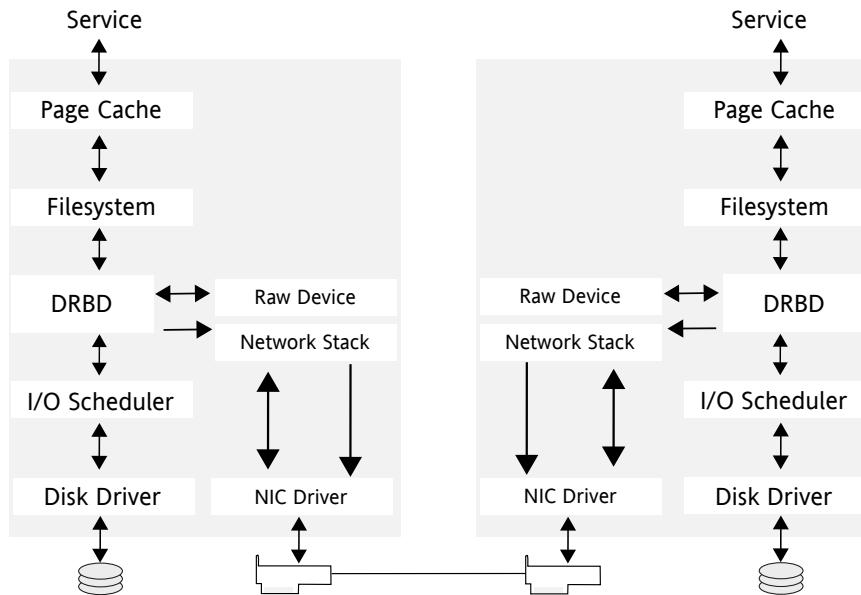
DRBD replicates data on the primary device to the secondary device in a way that ensures that both copies of the data remain identical. Think of it as a networked RAID 1. It mirrors data in real-time, so its replication occurs continuously. Applications do not need to know that in fact their data is 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 15.1, “Position of DRBD within Linux” (page 242). To communicate with DRBD, users use the high-level command `drbdadm`. For maximum flexibility DRBD comes with the low-level tool `drbdsetup`.

Figure 15.1: 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.

15.2 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, refer to Table 15.1, “DRBD RPM Packages” (page 243). It contains a list of all RPM packages for DRBD. Recently, the `drbd` package has been split into separate packages.

Table 15.1: DRBD RPM Packages

| Filename | Explanation |
|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <code>drbd</code> | Convenience package, split into other |
| <code>drbd-bash-completion</code> | Programmable bash completion support for <code>drbdadm</code> |
| <code>drbd-heartbeat</code> | Heartbeat resource agent for DRBD (only needed for Heartbeat) |
| <code>drbd-kmp-default</code> | Kernel module for DRBD (needed) |
| <code>drbd-kmp-xen</code> | Xen kernel module for DRBD |
| <code>drbd-udev</code> | udev integration scripts for DRBD, managing symlinks to DRBD devices in <code>/dev/drbd/by-res</code> and <code>/dev/drbd/by-disk</code> |

| Filename | Explanation |
|----------------|---------------------------------------------|
| drbd-utils | Management utilities for DRBD (needed) |
| 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.

15.3 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 15.1: 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/*.*";
```

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 { 1
    device /dev/drbd_r0 minor 0; 2
    disk /dev/sdal; 3
    meta-disk internal; 4
    on jupiter { 5
        address 192.168.1.10:7788; 6
    }
    on venus { 5 (page 245)
        address 192.168.1.11:7788; 6 (page 245)
    }
    syncer {
        rate 7M; 7
    }
}
```

- 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 bandwidth. It only limits the resynchronization, not the mirroring.

- ❽ Check the syntax of your configuration file(s). If the following command returns an error, verify your files:

```
drbdadm dump all
```

- ❾ If you have configured Csync2 (which should be the default), the DRBD configuration files are already included in the list of files which need to be synchronized. To syncronize them, use:

```
csync2 -xv
```

If you do not have Csync2 (or do not want to use it), copy the DRBD configuration files manually 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 already contains a file system that you do not need anymore, destroy the file system 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 file system 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/
```

To use YaST to configure DRBD, proceed as follows:

Procedure 15.2: Using YaST to Configure DRBD

- 1 Start YaST and select the configuration module *High Availability > DRBD*. If you have already a DRBD configuration, YaST warns you. YaST will change your configuration and will save your old DRBD configuration files as *.YaSTsave.
- 2 In *Start-up Configuration > Booting* select *On* to start DRBD always at boot time.
- 3 If you need to configure more than one replicated resource, select *Global Configuration*. The input field *Minor Count* selects how many different DRBD resources may be configured without restarting the computer.
- 4 The actual configuration of the resource is done in *Resource Configuration*. Press *Add* to create a new resource. The following parameters have to be set twice:

| | |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Resource Name</i> | The name of the resource, often called r0. This parameter is mandatory. |
| <i>Name</i> | The hostname of the relevant node |
| <i>Address:Port</i> | The IP address and port number (default 7788) of the respective node |
| <i>Device</i> | The device that holds the replicated data on the respective node. Use this device to create file systems and mount operations. |
| <i>Disk</i> | The device that is replicated between both nodes |
| <i>Meta-disk</i> | <p>The <i>Meta-disk</i> is either set to the value <code>internal</code> or specifies an explicit device extended by an index to hold the meta data needed by DRBD.</p> <p>When using <code>internal</code>, the last 128 MB of the replicated device are used to store the meta data.</p> |

A real device may also be used for multiple drbd resources. For example, if your *Meta-Disk* is `/dev/sda6[0]` for the first resource, you may use `/dev/sda6[1]` for the second resource. However, there must be at least 128 MB space for each resource available on this disk.

All of these options are explained in the examples in the `/usr/share/doc/packages/drbd-utils/drbd.conf` file and in the man page of `drbd.conf(5)`.

- 5 If you have configured Csync2 (which should be the default), the DRBD configuration files are already included in the list of files which need to be synchronized. To synchronize them, use:

```
csync2 -xv
```

If you do not have Csync2 (or do not want to use it), copy the DRBD configuration files manually to the other node (pretending to be another node with the name venus):

```
scp /etc/drbd.conf venus:/etc/
scp /etc/drbd.d/* venus:/etc/drbd.d/
```

- 6 Initialize and start the DRBD service on both systems by entering the following on each node:

```
drbdadm create-md r0
rcdrbd start
```

- 7 Configure node1 as the primary node by entering the following on node1:

```
drbdsetup /dev/drbd0 primary --overwrite-data-of-peer
```

- 8 Check the DRBD service status by entering the following on each node:

```
rcdrbd status
```

Before proceeding, wait until the block devices on both nodes are fully synchronized. Repeat the `rcdrbd status` command to follow the synchronization progress.

- 9** After the block devices on both nodes are fully synchronized, format the DRBD device on the primary with your preferred file system. Any Linux file system can be used.

IMPORTANT

Always use the `/dev/drbd<n>` name in the command, not the actual `/dev/disk` device name.

15.4 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 47).

15.5 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 only works if you use LVM.

3. Activate bmbv on Linux software RAID systems. The `use-bmbv` keyword enables DRBD to process IO requests in units not larger than 4kByte. However, this option should be adapted carefully. Use the following line in the common disk section of your DRBD configuration, usually in `/etc/drbd.d/global_common.conf`:

```
disk {  
    use-bmbv;  
}
```

15.6 Troubleshooting DRBD

The drbd setup involves many different components and problems may arise from different sources. The following sections cover several common scenarios and recommend various solutions.

15.6.1 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.

15.6.2 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`.

15.6.3 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.

15.6.4 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
```

15.7 For More Information

The following open source resources are available for DRBD:

- The project home page <http://www.drbd.org>.
- See Highly Available NFS Storage with DRBD and Pacemaker (↑Highly Available NFS Storage with DRBD and Pacemaker).
- 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)`, `drbdddisk(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`

16

Cluster Logical Volume Manager (cLVM)

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.

16.1 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.

16.2 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 16.2.2, “Scenario: cLVM With iSCSI on SANs” (page 261).
- **DRBD** This solution only provides RAID 0 (striping) and RAID 1 (mirroring). The last method is described in Section 16.2.3, “Scenario: cLVM With DRBD” (page 265).
- **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 16.2.1, “Creating the Cluster Resources” (page 258) and then your LVM volumes. Otherwise it is impossible to remove the volumes later.

16.2.1 Creating the Cluster Resources

Preparing the cluster for use of cLVM includes the following basic steps:

- Creating a DLM Resource (page 259)

- Creating LVM and cLVM Resources (page 259)

Procedure 16.1: *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 16.2, “Creating LVM and cLVM Resources” (page 259).

Otherwise, configure a DLM resource and a corresponding base group and base clone as described in Procedure 14.1, “Configuring DLM and O2CB Resources” (page 232).

- 5 Leave the `crm live` configuration with `exit`.

Procedure 16.2: *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"
```

- 4 Configure an LVM resource for the volume group as follows:

```
primitive vgl ocf:heartbeat:LVM \
    params volgrpname="cluster-vg" \
    op monitor interval="60" timeout="60"
```

- 5** 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 260):

```
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 14.1, “Configuring DLM and O2CB Resources” (page 232):

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

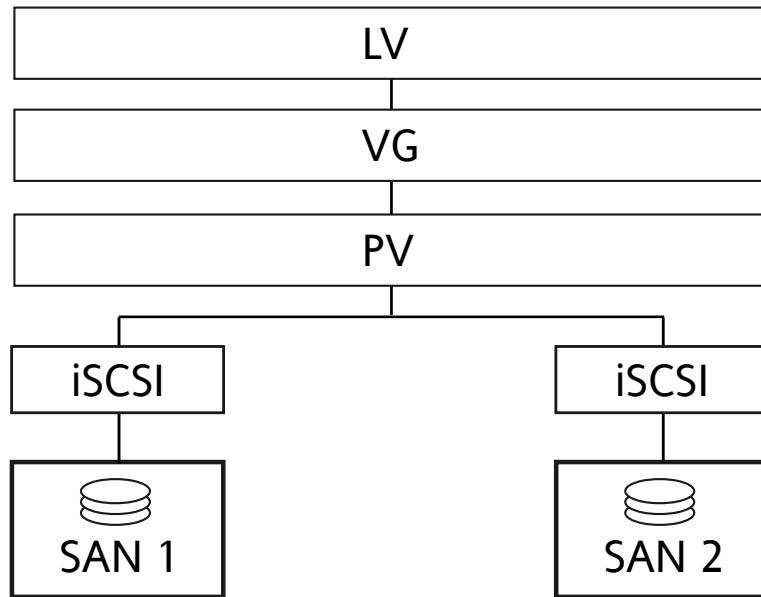
If your setup does not include OCFS2, omit the `ocfs2-1` primitive from the base group. The `o2cb` primitive can be configured and included in the group anyway, regardless of whether you use OCFS2 or not.

- 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 431).
- 8** If everything is correct, submit your changes with `commit` and leave the `crm live` configuration with `exit`.

16.2.2 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 16.1, “Setup of iSCSI with cLVM” (page 261).

Figure 16.1: 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 16.3: 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 for your different targets.
 - 5e** Click *Add*.
 - 5f** Enter the device name in *Path* and use a *Scsiid*.
 - 5g** Click *Next* twice.
 - 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 16.4: 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 16.3, “Configuring iSCSI Targets (SAN)” (page 261)). 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 16.5: Creating the LVM Volume Groups

- 1** Open a `root` shell on one of the nodes you have run the iSCSI initiator from Procedure 16.4, “Configuring iSCSI Initiators” (page 262).

- 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-AI58-1xB1-mWm2-xn51-agM2-0UuHFC
```

- 6** Check the volume group with `vgdisplay`:

```
--- Volume group ---  
VG Name          clustervg  
System ID        lvm2  
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 14, *OCFS2* (page 229)

16.2.3 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 16.6: *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 15.1, “Manually Configuring DRBD” (page 245). 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 16.1, “Creating a DLM Resource” (page 259) 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 logical 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.

16.3 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.

16.4 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

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.

17.1 Storage-based Fencing

You can reliably avoid split brain scenarios by using one or more STONITH Block Devices (SBD), watchdog support and the `external/sbd` STONITH agent.

17.1.1 Overview

In an environment where all nodes have access to shared storage, a small partition (1MB) of the device is formatted for 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.

17.1.2 Number of SBD Devices

SBD supports the use of 1-3 devices:

One Device

The most simple implementation. It is appropriate for clusters where all of your data is one the same shared storage.

Two Devices

This configuration is primarily useful for environments that use host-based mirroring but where no third storage device is available. SBD will not terminate itself if it loses access to one mirror leg, allowing the cluster to continue. However, since SBD does not have enough knowledge to detect an asymmetric split of the storage, it will not fence the other side while only one mirror leg is available. Thus, it cannot automatically tolerate a second failure while one of the storage arrays is down.

Three Devices

The most reliable configuration. It is resilient against outages of one device—be it due to failures or maintenance. SBD will only terminate itself if more than one device is lost. Fencing messages can be successfully be transmitted if at least two devices are still accessible.

This configuration is suitable for more complex scenarios where storage is not restricted to a single array. Host-based mirroring solutions can have one SBD per mirror leg (not mirrored itself), and an additional tie-breaker on iSCSI.

17.1.3 Setting Up Storage-based Protection

The following steps are necessary to set up storage-based protection:

- 1** Creating the SBD Partition (page 272)
- 2** Setting Up the Software Watchdog (page 273)
- 3** Starting the SBD Daemon (page 274)
- 4** Testing SBD (page 275)
- 5** Configuring the Fencing Resource (page 276)

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.
-

17.1.3.1 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 `sbd` 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.

If you want to use more than one device for SBD, provide the devices by specifying the `-d` option multiple times, for example:

```
sbd -d /dev/SBD1 -d /dev/SBD2 -d /dev/SBD3 create
```

- 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/SBD -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.

17.1.3.2 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 17.1.3.3, “Starting the SBD Daemon” (page 274), 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 -qf 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
```

17.1.3.3 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 you need to specify multiple devices in the first line, separate them by a semicolon (the order of the devices does not matter):

```
SBD_DEVICE="/dev/SBD1; /dev/SBD2; /dev/SBD3"
# 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. This is technically correct behavior, but depending on your administrative policies, most likely a nuisance. In such cases, better do not automatically start up OpenAIS on boot.

- 3** Copy `/etc/sysconfig/sbd` to all nodes (either manually or with Csync2, see also Section 3.5.5, “Transferring the Configuration to All Nodes” (page 39)).
- 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`.

17.1.3.4 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.

17.1.3.5 Configuring the Fencing Resource

- 1 To complete the SBD setup, 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
stonith_device="/dev/SBD"
crm(live)configure# commit
crm(live)configure# quit
```

The resource does not need to be cloned, as it would shut down the respective node anyway if there was a problem.

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 host list 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 successfully configured for shared-storage fencing and will utilize this method in case a node needs to be fenced.

17.1.3.6 For More Information

http://www.linux-ha.org/wiki/SBD_Fencing

17.2 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.

17.2.1 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.

17.2.2 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 17.1: 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 17.2: 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
```

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.

18.1 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 as a “cluster implementation of the TDB database used by Samba and other projects to store temporary data”.

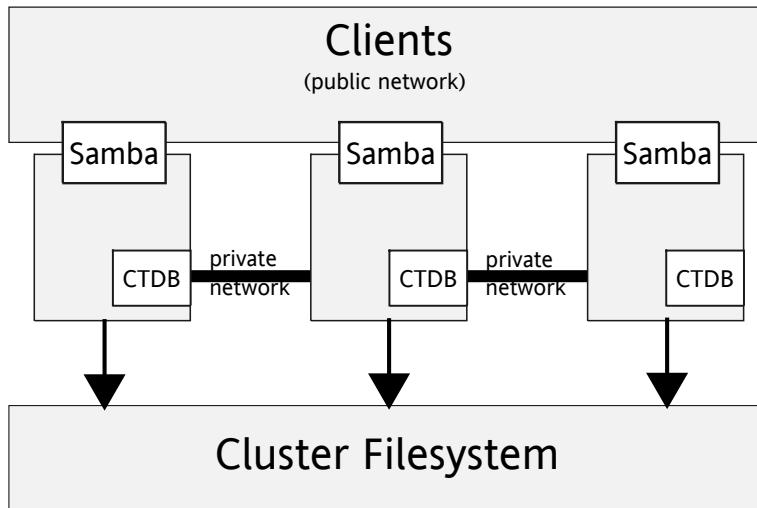
Each cluster node runs a local CTDB daemon. Samba communicates with its local CTDB daemon instead of writing directly to its TDB. The daemons exchange metadata over the network, but actual write and read operations are done on a local copy with fast storage. The concept of CTDB is displayed in Figure 18.1, “Structure of a CTDB Cluster” (page 280).

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.

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 18.1: 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.
- The 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.

18.2 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 that 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 47).
- 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=xattrDEVICE
```

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 Copy the configuration file to all of your nodes by using csync2:

```
csync2 -xv
```

For more information, see Procedure 3.9, “Synchronizing the Configuration Files with Csync2” (page 40).

5 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 interval="10" timeout="20" \  
    op start interval="0" timeout="90" \  
    op stop interval="0" timeout="100"  
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
```

6 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
```

7 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
```

- 8** 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
```

18.3 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. Detailed debug messages 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 gives some indication 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 18.1: 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 file `data.txt` 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 dramatic drop in 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 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 one after the other. 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, find more information in Chapter 14, *OCFS2* (page 229).

18.4 Joining Active Directory Domains

Active Directory (AD) is a directory service for Windows server systems.

To configure the CTDB, the general procedure is as follows:

- 1** Consult your Windows Server documentation for instructions on how to setup an Active Directory domain. In this example, we use the following parameters:

| | |
|--------------------------------|----------------------------|
| AD and DNS server | win2k3.2k3test.example.com |
| AD domain | 2k3test.example.com |
| Cluster AD member NETBIOS name | CTDB-SERVER |

- 2** Procedure 18.2, “Configuring CTDB” (page 285)
- 3** Procedure 18.3, “Joining Active Directory” (page 286)

The next step is to configure the CTDB:

Procedure 18.2: Configuring CTDB

- 1** Make sure you have configured your cluster as shown in Section 18.2, “Basic Configuration” (page 281).

2 Create a primitive with colocation and ordering constraints:

```
#crm configure
crm(live)configure# primitive ctdb ocf:heartbeat:CTDB \
    params ctdb_recovery_lock="/clusterfs/samba/ctdb.lock" \
    op monitor interval="10" timeout="20" \
    op start interval="0" timeout="90" \
    op stop interval="0" timeout="100"
crm(live)configure# clone ctdb-clone ctdb \
    meta globally-unique="false" interleave="true"
crm(live)configure# colocation ctdb-with-fs inf: ctdb-clone c-clusterfs
crm(live)configure# order start-ctdb-after-fs inf: c-clusterfs ctdb-clone
crm(live)configure# commit
```

3 Stop the CTDB resource on one node:

```
#crm resource stop ctdb-clone
```

4 Change the /etc/samba.conf configuration file:

```
[global]
workgroup = 2K3TEST
realm = 2k3test.example.com
security = ADS
netbios name = CTDB-SERVER
idmap config * : range = 1000000-2000000
```

5 Update on all nodes the file /etc/samba.conf:

```
csync2 -xv
```

6 Restart the CTDB resource:

```
#crm resource start ctdb-clone
```

Finally, join your cluster to the Active Directory server:

Procedure 18.3: Joining Active Directory

- 1 Edit the file /etc/resolv.conf and set the nameserver to your AD server. This addition needs to be added to all nodes.

You can use Csync2 for this file if—and only if—the following conditions are true:

- The content of /etc/resolv.conf is the same on all nodes.

- The file `/etc/resolv.conf` is manually edited.
- The file is not generated by the YaST network module.

In that case, add the file `/etc/resolv.conf` into the Csync2 configuration file `/etc/csync2/csync2.cfg`.

- 2** Synchronize the cluster node clocks with the AD server. This can be ensured either with Kerberos or the Network Time Protocol (NTP).
- 3** Run `crm configure edit` and search for the `ctdb` resource. Add the following line:

```
ctdb_manages_winbind="false"
```

- 4** Set the following lines in `/etc/nsswitch.conf`:

```
passwd: files winbind
group: files winbind
```

- 5** Restart the NSC daemon:

```
rcnscd restart
```

- 6** Create the Kerberos configuration file `/etc/krb5.conf`:

```
[libdefaults]
    default_realm = 2k3test.example.com

[realms]
    2k3test.example.com = {
        kdc = win2k3.2k3test.example.com
        admin_server = win2k3.2k3test.example.com
        default_domain = 2k3test.example.com
    }

[domain_realm]
    .2k3test.example.com = 2k3test.example.com
    2k3test.example.com = 2k3test.example.com
```

- 7** Run CTDB on all nodes:

```
# crm resource cleanup ctdb:0
# crm resource cleanup ctdb:1
```

8 Wait until the “unhealthy” status disappears. The status should look like this:

```
# ctdb status
Number of nodes:2
pnn:0 192.168.1.10  OK (THIS NODE)
pnn:1 192.168.1.20  OK
Generation:1046869196
Size:2
hash:0 lmast:0
hash:1 lmast:1
Recovery mode:NORMAL (0)
Recovery master:0
```

9 Join the realm 2k3test.example.com:

```
# net ads join -U Administrator
Enter Administrator's password: *****
Using short domain name -- 2K3TEST
Joined 'CTDB-SERVER' to realm '2k3test.example.com'
Not doing automatic DNS update in a clustered setup.
```

10 Change the `ctdb_manages_winbind` option:

10a Stop the `ctdb` resource:

```
crm resource stop ctdb-clone
```

10b Run `crm configure edit` and search for the `ctdb` resource as you did in Step 3 (page 287). Change the value from false to true:

```
ctdb_manages_winbind="true"
```

10c Restart the `ctdb` resource:

```
# crm resource start ctdb-clone
```

11 Run on all nodes to see the list of Active Directory users:

```
# wbinfo -u
2K3TEST\administrator
2K3TEST\guest
2K3TEST\support_388945a0
2K3TEST\krbtgt
```

18.5 For More Information

- [http://linux-ha.org/wiki/CTDB_\(resource_agent\)](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

Disaster Recovery with ReaR

ReaR (Relax and Recover) is an administrator tool-set for creating disaster recovery images. The disaster recovery information can either be stored via the network or locally on hard disks, USB devices, DVD/CD-R, tape or similar. The backup data is stored on a network file system (NFS).

Keep in mind, ReaR needs to be configured and tested *before* any disaster happens. ReaR will not save you, if a disaster has already taken place.

WARNING: Extensive Testing Required

It is essential, whenever you create a rescue CD, to *always* test the disaster recovery with an *identical* test machine. Only if this procedure works satisfactorily, your disaster recovery system is correctly and reliably set up.

19.1 Terminology

Disaster

Unexpected interruption of critical infrastructure induced by nature, humans, hardware failure, or software bugs.

Disaster Recovery

According to ReaR's home page, disaster recovery is “the process by which a business function is restored to the normal, steady state after a disaster.”

Disaster Recover Plan

A strategy to recover from a disaster with minimum impact on IT infrastructure.

19.2 Conceptual Overview

SUSE Linux Enterprise Server ships a disaster recovery system in two packages:

- `rear`
- `rear-SUSE`

The package `rear-SUSE` is another method of a disaster recovery which incorporates AutoYaST to recreate your basic system. You should try and test both methods for your systems. Every IT infrastructure is different, in one case `rear` is enough, in other situations `rear-SUSE` is a better fit. Regardless of the method, both are used in the following way:

1. **Preparation** Make a bootable CD and backup system and user data.
2. **Testing** Test the recovery process thoroughly and the backup on the same hardware as your main system before any disaster happens.
3. **Recovery** Boot from the rescue CD and restore your system from your backup.

To prepare the rescue media, the following steps are performed by ReaR:

Preparation of Rescue Media by ReaR

1. Gather system information.
2. Store disk layout (partitioning, filesystems, LVM, RAID, and boot loader).
3. Clone the system (Kernel drivers and modules, device driver configuration, network configuration, system software and tools).
4. Backup system and user data.
5. Create bootable rescue CD with system configuration.

When a disaster occurred, the recovery process takes these actions:

Recovery Process

1. Boot from the rescue media.
2. Restore the disk layout (partitions, RAID configurations, LVM, file systems).
3. Restore system and user data.
4. Restore boot loader.
5. Reboot system.

19.3 Preparing for the Worst Scenarios: Disaster Recovery Plans

Before the worst scenario happens, take actions to prepare with a *disaster recovery plan*. A disaster recovery plan is a document where all risks, infrastructure, and the budget is being collected. Maybe you have already some plan in place, but here is the general overview:

- **Risk Analysis** Conduct a solid risk analysis of your infrastructure. List all the possible threats and evaluate how serious they are. Determine how likely these threats are and prioritize them. It is recommended to use a simple categorization: probability and impact.
- **Budget Planing** The outcome of the analysis is an overview, which risks can be tolerated and which are critical for your business. Ask yourself, how can you minimize risks and how much will it cost. Depending on how big your company is, spend two to fifteen percent of the overall IT budget on disaster recovery.
- **Disaster Recovery Plan Development** Make checklists, test procedures, establish and assign priorities, and inventory your IT infrastructure. Define how to deal with a problem when some services in your infrastructure fail.
- **Test** After defining an elaborate plan, test it. Test it at least once a year. Use the same testing hardware as your main IT infrastructure.

19.4 Setting Up ReaR

ReaR supports some backup tools (Tivoli Storage Manager, QNetix Galaxy, Symantec NetBackup, HP DataProtector) and can output its rescue medium to a CD or PXE environment. The restore step is possible through NFS or CIFS and other network file systems. Find more information in the ReaR documentation at <http://rear.github.com/documentation/>.

To use ReaR you need at least two identical systems: the main machine where your productive environment is stored and the test machine. “Identical” in this context means, for example, you can replace a network card with another one using the same Kernel driver. If a hardware component does not use the same driver, it is not considered identical by ReaR.

ReaR can be used in different scenarios. The following example uses a NFS server as backup storage:

Procedure 19.1: Storing Your Backup on a NFS Server

- 1 Set up a NFS server with YaST as described in *Sharing File Systems with NFS* from http://www.suse.com/documentation/sles11/book_sle_admin/data/cha_nfs.html.
- 2 Adapt the configuration file(s). Depending on how many servers you want to recover, use `/etc/rear/site.conf` for site-wide settings and `/etc/rear/local.conf` for machine-local settings. The following example uses a `/etc/rear/local.conf` configuration file. Replace the `NETFS_URL` with your own values. Further options are listed in the *Various Settings* section of the documentation at <http://rear.github.com/documentation/>.

```
# Create ReaR rescue media as ISO image:  
OUTPUT=ISO  
# Store the backup file via NFS:  
BACKUP=NETFS  
# Only a NETFS_URL of the form 'nfs://host/path' is supported  
# so that 'mount -o nolock -t nfs host:/path' works.  
NETFS_URL=nfs://192.168.1.1/nfs/rear/  
# Keep an older copy of the backup in a HOSTNAME.old directory  
# provided there is no '.lockfile' in the HOSTNAME directory:  
NETFS_KEEP_OLD_BACKUP_COPY=yes
```

If your NFS host is not an IP address but a hostname, DNS must work when the backup is restored.

3 Prepare the backup by running:

```
rear mkbackup
```

To perform a disaster recovery on your test machine, proceed as follows:

Procedure 19.2: Perform Disaster Recovery on Test Machine

- 1** Locate the recovery ISO image stored as `/tmp/rear-HOSTNAME.iso` and burn it on CD.
- 2** Boot your test machine with the recovery CD.
- 3** Enter `rear` at the boot prompt.
- 4** Log in as `root` (no password needed).
- 5** Enter `rear recover` to start the recovery process. The recovery process installs and configures the machine and retrieves the backup data from your NFS server.

After this procedure, make sure the test machine is correctly set up and can serve as a replacement for your main machine. Test this procedure on a regular basis to ensure everything works as expected. Keep copies of the rescue CD iso, in case the media is damaged.

19.5 Setting Up `rear-SUSE` with AutoYaST

With the package `rear-SUSE`, the recovery process uses AutoYaST together with a ReaR backup, which is stored on a NFS server. Before you use it, check your disk space, you need at least 5 GB and up to 15 GB. The size is the sum of the following:

- The SUSE installation medium (size of 5 GB DVD)
- A “working directory” copied from the SUSE installation medium. The working copy is used by `rear-SUSE` for preparing the recovery ISO image.

- Recovery ISO image. From 500 MB up to 5 GB.

Depending on your situation, you can limit the overall size. Refer to the rear-SUSE documentation for details in `/usr/share/doc/packages/rear-SUSE/README`.

NOTE: Support of File Systems

AutoYaST can only recover file systems which are supported by YaST. Some file systems, like OCFS2, are currently not supported.

The following procedure creates a “full” recovery ISO image. This means, it downloads the SUSE installation medium, creates a full backup, and generates the recovery ISO image. Proceed as follows:

Procedure 19.3: Creating a Full Recovery Image

- 1 Get the IP or hostname of your NFS server where your ReaR backup will be stored. If you do not have one, set up as described in *Sharing File Systems with NFS*.
- 2 Proceed with the backup as described in Procedure 19.1, “Storing Your Backup on a NFS Server” (page 294), but use the command `rear mkbackuponly` to prepare the backup. You do not need to burn the CD.
- 3 Start the backup, and replace `BASE_DIR` with your working directory (for example, `/var/tmp/rear-SUSE/`), and `MEDIUM_URI` (for example, `http://server/path/medium.iso`):

```
RecoveryImage -c configure-all -d BASE_DIR \
  -l log-to-base-dir \
  -b make-rear-backup \
  -a clone-system \
  -i install-RPMs \
  -r restore-all \
  -m MEDIUM_URI
```

This will start the backup process and store your data on your NFS server. After the backup process, the recovery ISO image is created.

- 4 Burn the recovery ISO image on a DVD. Find the image in your `BASE_DIR` path with the name `RecoverImage.DATE.iso`.

After the DVD has been burnt, test your disaster recovery with another server using the same hardware. For example, identical hardware is a hard disk with the same disk geometry, or a network card which uses the same Kernel driver.

Procedure 19.4: Recovering with the Recovery Image

- 1** Boot your test machine with the recovery image from Procedure 19.3, “Creating a Full Recovery Image” (page 296) of Step 4.
- 2** Enter `autorecover` at the boot prompt. The recovery image boots and starts the AutoYaST installation process.
- 3** Follow the instructions on the screen. Any error messages regarding the packages `rear` or `rear-SUSE` can be ignored.

After the recovery, make sure the test machine functions properly and all data has been restored correctly from your NFS backup.

19.6 For More Information

- <http://rear.github.com/>
- Manpage for `rear`
- `/usr/share/doc/packages/rear/README`
- `/usr/share/doc/packages/rear-SUSE/README`

Part IV. Troubleshooting and Reference

20

Troubleshooting

Strange problems may occur that are not easy to understand, especially when starting to experiment with High Availability. However, there are several utilities that allow you to take a closer look at the High Availability internal processes. This chapter recommends various solutions.

20.1 Installation and First Steps

Troubleshooting difficulties when installing the packages or 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 SP2 on each of the cluster nodes and if the *High Availability* pattern is installed on each of the machines as described in Section 3.3, “Installation as Add-on” (page 27).

Is the initial configuration the same for all cluster nodes?

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 3.5, “Manual Cluster Setup (YaST)” (page 31).

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 , 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`.

20.2 Logging

I enabled monitoring but there is no trace of monitoring operations in the logs?

The `lrmrd` daemon does not log recurring monitor operations unless an error occurred. Logging all recurring operations would produce too much noise. Therefore recurring monitor operations are logged only once an hour.

I only get a failed message. Is it possible to get more information?

Add the `--verbose` parameter to your commands. If you do that multiple times, the debug output becomes quite verbose. See `/var/log/messages` for useful hints.

How can I get an overview of all my nodes and resources?

Use the `crm_mon` command. The following displays the resource operation history (option `-o`) and inactive resources (`-r`):

```
crm_mon -o -r
```

The display is refreshed when the status changes (to cancel this press Ctrl + C.) An example may look like:

Example 20.1: 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, then check your resources and operations.

The *Configuration Explained* PDF covers three different recovery types in the *How Does the Cluster Interpret the OCF Return Codes?* section. It is available at “<http://www.clusterlabs.org/doc/>”.

20.3 Resources

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 7.4.2, “Cleaning Up Resources” (page 171).

How can I list my currently known resources?

Use the command `crm resource list` to display your current resources.

I configured a resource, but it always fails. Why?

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.

Why do resources not fail over and why are there no errors?

If your cluster is a two node cluster, killing one node will leave the remaining node without quorum. Unless you set the `no-quorum-policy` property to `ignore`, nothing happens. For two-node clusters you need:

```
property no-quorum-policy="ignore"
```

Another possibility is that the killed node is considered unclean. Then it is necessary to fence it. If the stonith resource is not operational or does not exist, the remaining node will wait for the fencing to happen. The fencing timeouts are typically high, so it may take quite a while to see any obvious sign of problems (if ever).

Yet another possible explanation is that a resource is simply not allowed to run on this node. That may be due to a failure which happened in the past and which was not “cleaned”. Or it may be due to an earlier administrative action, i.e. a location

constraint with a negative score. Such a location constraint is for instance inserted by the `crm resource migrate` command.

Why can I never tell where my resource will run?

If there are no location constraints for a resource, its placement is subject to an (almost) random node choice. You are well advised to always express a preferred node for resources. That does not mean that you need to specify location preferences for *all* resources. One preference suffices for a set of related (colocated) resources. A node preference looks like this:

```
location rsc-prefers-node1 rsc 100: node1
```

20.4 STONITH and Fencing

Why does my STONITH resource not start?

Start (or enable) operation includes checking the status of the device. If the device is not ready, the STONITH resource will fail to start.

At the same time the STONITH plugin will be asked to produce a host list. If this list is empty, there is no point in running a STONITH resource which cannot shoot anything. The name of the host on which STONITH is running is filtered from the list, since the node cannot shoot itself.

If you want to use single-host management devices such as lights-out devices, make sure that the `stonith` resource is *not* allowed to run on the node which it is supposed to fence. Use an infinitely negative location node preference (constraint). The cluster will move the `stonith` resource to another place where it can start, but not before informing you.

Why does fencing not happen, although I have the STONITH resource?

Each STONITH resource must provide a host list. This list may be inserted by hand in the STONITH resource configuration or retrieved from the device itself, for instance from outlet names. That depends on the nature of the STONITH plugin.

`stonithd` uses the list to find out which STONITH resource can fence the target node. Only if the node appears in the list can the STONITH resource shoot (fence) the node.

If `stonithd` does not find the node in any of the host lists provided by running STONITH resources, it will ask `stonithd` instances on other nodes. If the target

node does not show up in the host lists of other `stonithd` instances, the fencing request ends in a timeout at the originating node.

Why does my STONITH resource fail occasionally?

Power management devices may give up if there is too much broadcast traffic.

Space out the monitor operations. Given that fencing is necessary only once in a while (and hopefully never), checking the device status once a few hours is more than enough.

Also, some of these devices may refuse to talk to more than one party at the same time. This may be a problem if you keep a terminal or browser session open while the cluster tries to test the status.

20.5 Miscellaneous

How can I run commands on all cluster nodes?

Use the command `pssh` for this task. If necessary, install `pssh`. Create a file (for example `hosts.txt`) where you collect all your IP addresses or hostnames you want to visit. Make sure you can log in with `ssh` to each host listed in your `hosts.txt` file. If everything is correctly prepared, execute `pssh` and use the `hosts.txt` file (option `-h`) and the interactive mode (option `-i`) as shown in this example:

```
pssh -i -h hosts.txt "ls -l /corosync/*.conf"
[1] 08:28:32 [SUCCESS] root@venus.example.com
-rw-r--r-- 1 root root 1480 Nov 14 13:37 /etc/corosync/corosync.conf
[2] 08:28:32 [SUCCESS] root@192.168.2.102
-rw-r--r-- 1 root root 1480 Nov 14 13:37 /etc/corosync/corosync.conf
```

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.

Why can several nodes of my cluster not see each other?

There could be several reasons:

- Look first in the configuration file `/etc/corosync/corosync.conf` and check if the multicast or unicast 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 or unicast 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.

Why can an OCFS2 device not be mounted?

Check `/var/log/messages` for 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.

How can I create a report with an analysis of all my cluster nodes?

Use the tool `hb_report` to create a report. The tool is used to compile:

- Cluster-wide log files,
- Package states,
- DLM/OCFS2 states,
- System information,
- CIB history,
- Parsing of core dump reports, if a `debuginfo` package is installed.

Usually run `hb_report` with the following command:

```
hb_report -f 0:00 -n earth -n venus
```

The command extracts all information since 0am on the hosts earth and venus and creates a `.tar.bz2` archive named `hb_report-DATE.tar.bz2` in the current directory, for example, `hb_report-Wed-03-Mar-2012`. If you are only interested in a specific time frame, add the end time with the `-t` option.

WARNING: Remove Sensitive Information

The `hb_report` tool tries to remove any sensitive information from the CIB and the `peinput` files, however, it can not do everything. If you have more sensitive information, supply additional patterns. The logs and the `crm_mon`, `ccm_tool`, and `crm_verify` output are *not* sanitized.

Before sharing your data in any way, check the archive and remove all information you do not want to expose.

Customize the command execution with further options. For example, if you have an OpenAIS cluster, you certainly want to add the option `-A`. In case you have another user who has permissions to the cluster, use the `-u` option and specify this user (in addition to `root` and `hacluster`). Further options can be found in the manpage of `hb_report`.

After `hb_report` analyzed all the relevant log files and created the directory (or archive), check the logs for an uppercase `ERROR` string. The most important files in the top level directory of the report are:

`analysis.txt`

Compares files that should be identical on all nodes.

`crm_mon.txt`

Contains the output of the `crm_mon` command.

`corosync.txt`

Contains a copy of the Corosync configuration file.

`description.txt`

Contains all cluster package versions on your nodes. There is also the `sysinfo.txt` file which is node specific. It is linked to the top directory.

Node-specific files are stored in a subdirectory named by the node's name.

20.6 Fore More Information

For additional information about high availability on Linux, including configuring cluster resources and managing and customizing a High Availability cluster, see <http://clusterlabs.org/wiki/Documentation>.

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_workdir=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_workdir=Full path name of the work directory

The path from where the binfile will be executed.

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. This 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 parameter 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_testconfigfile=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:conntrackd (7)

ocf:conntrackd — This resource agent manages conntrackd

Synopsis

```
[OCF_RESKEY_conntrackd=string] [OCF_RESKEY_config=string]
conntrackd [start | promote | demote | notify | stop | monitor | monitor | meta-data |
validate-all]
```

Description

Master/Slave OCF Resource Agent for conntrackd

Supported Parameters

OCF_RESKEY_conntrackd=Name of the conntrackd executable

Name of the conntrackd executable. If conntrackd is installed and available in the default PATH, it is sufficient to configure the name of the binary For example "my-conntrackd-binary-version-0.9.14" If conntrackd is installed somewhere else, you may also give a full path For example "/packages/conntrackd-0.9.14/sbin/conntrackd"

OCF_RESKEY_config=Path to conntrackd.conf

Full path to the conntrackd.conf file. For example "/packages/conntrackd-0.9.14/etc/conntrackd/conntrackd.conf"

ocf:CTDB (7)

ocf:CTDB — CTDB Resource Agent

Synopsis

```
OCF_RESKEY_ctdb_recovery_lock=string
[OCF_RESKEY_ctdb_manages_samba=boolean]
[OCF_RESKEY_ctdb_manages_winbind=boolean]
[OCF_RESKEY_ctdb_service_smb=string]
[OCF_RESKEY_ctdb_service_nmb=string]
[OCF_RESKEY_ctdb_service_winbind=string]
[OCF_RESKEY_ctdb_samba_skip_share_check=boolean]
[OCF_RESKEY_ctdb_monitor_free_memory=integer]
[OCF_RESKEY_ctdb_start_as_disabled=boolean]
[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]
OCF_RESKEY_smb_private_dir=string
[OCF_RESKEY_smb_passdb_backend=string]
[OCF_RESKEY_smb_idmap_backend=string]
[OCF_RESKEY_smb_fileid_algorithm=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. Create /etc/ctdb/nodes containing a list of private IP addresses of each node in the cluster, then configure this RA as a clone. To have CTDB manage Samba, set ctdb_manages_samba="yes". Note that this option will be deprecated in future, in favour of configuring a separate Samba resource. For more information see [http://linux-ha.org/wiki/CTDB_\(resource_agent\)](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_ctdb_manages_samba=Should CTDB manage Samba?

Should CTDB manage starting/stopping the Samba service for you? This will be deprecated in future, in favor of configuring a separate Samba resource.

OCF_RESKEY_ctdb_manages_winbind=Should CTDB manage Winbind?

Should CTDB manage starting/stopping the Winbind service for you? This will be deprecated in future, in favor of configuring a separate Winbind resource.

OCF_RESKEY_ctdb_service_smb=Name of smb init script

Name of smb init script. Only necessary if CTDB is managing Samba directly.
Will usually be auto-detected.

OCF_RESKEY_ctdb_service_nmb=Name of nmb init script

Name of nmb init script. Only necessary if CTDB is managing Samba directly.
Will usually be auto-detected.

OCF_RESKEY_ctdb_service_winbind=Name of winbind init script

Name of winbind init script. Only necessary if CTDB is managing Winbind directly.
Will usually be auto-detected.

OCF_RESKEY_ctdb_samba_skip_share_check=Skip share check during monitor?

If there are very many shares it may not be feasible to check that all of them are available during each monitoring interval. In that case this check can be disabled.

OCF_RESKEY_ctdb_monitor_free_memory=Minimum amount of free memory (MB)

If the amount of free memory drops below this value the node will become unhealthy and ctdb and all managed services will be shutdown. Once this occurs, the administrator needs to find the reason for the OOM situation, rectify it and restart ctdb with "service ctdb start".

OCF_RESKEY_ctdb_start_as_disabled=Start CTDB disabled?

When set to yes, the CTDB node will start in DISABLED mode and not host any public ip addresses.

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. Only necessary if CTDB is managing Samba.

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_smb_passdb_backend=Samba passdb backend

Which backend to use for storing user and possibly group information. Only necessary if CTDB is managing Samba.

OCF_RESKEY_smb_idmap_backend=Samba idmap backend

Which backend to use for SID/uid/gid mapping. Only necessary if CTDB is managing Samba.

OCF_RESKEY_smb_fileid_algorithm=Samba VFS fileid algorithm

Which fileid:algorithm to use with vfs_fileid. The correct value depends on which clustered filesystem is in use, e.g.: for OCFS2, this should be set to "fsid". Only necessary if CTDB is managing Samba.

ocf:db2 (7)

ocf:db2 — Resource Agent that manages an IBM DB2 LUW databases in Standard role as primitive or in HADR roles as master/slave configuration. Multiple partitions are supported.

Synopsis

```
OCF_RESKEY_instance=string [OCF_RESKEY_dblist=string]  
[OCF_RESKEY_admin=string] [OCF_RESKEY_dbpartitionnum=string] db2  
[start | stop | promote | demote | notify | monitor | validate-all | meta-data]
```

Description

Resource Agent that manages an IBM DB2 LUW databases in Standard role as primitive or in HADR roles in master/slave configuration. Multiple partitions are supported.

Standard mode: An instance including all or selected databases is made highly available.

Configure each partition as a separate primitive resource. HADR mode: A single database in HADR configuration is made highly available by automating takeover operations.

Configure a master / slave resource with notifications enabled and an additional monitoring operation with role "Master". In case of HADR be very deliberate in specifying intervals/timeouts. The detection of a failure including promote must complete within HADR_PEER_WINDOW. In addition to honoring requirements for crash recovery etc. for your specific database use the following relations as guidance: "monitor interval" < HADR_PEER_WINDOW - (appr 30 sec) "promote timeout" <

HADR_PEER_WINDOW + (appr 20 sec) For further information and examples consult [http://www.linux-ha.org/wiki/db2_\(resource_agent\)](http://www.linux-ha.org/wiki/db2_(resource_agent))

Supported Parameters

OCF_RESKEY_instance=instance

The instance of the database(s).

OCF_RESKEY_dblist=List of databases to be managed

List of databases to be managed, e.g "db1 db2". Defaults to all databases in the instance. Specify one db for HADR mode.

OCF_RESKEY_admin=DEPRECATED: admin

DEPRECATED: The admin user of the instance.

OCF_RESKEY_dbpartitionnum=database partition number (DBPARTITIONNUM)

The number of the partition (DBPARTITIONNUM) to be managed.

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 | 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 [OCF_RESKEY_fake=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_RESKEY_fake=Fake attribute that can be changed to cause a reload

Fake attribute that can be changed to cause a reload

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 of 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:ethmonitor (7)

ocf:ethmonitor — Monitors network interfaces

Synopsis

```
OCF_RESKEY_interface=string OCF_RESKEY_name=string
[OCF_RESKEY_multiplier=integer] [OCF_RESKEY_repeat_count=integer]
[OCF_RESKEY_repeat_interval=integer]
[OCF_RESKEY_pktnr_timeout=integer]
[OCF_RESKEY_arping_count=integer]
[OCF_RESKEY_arping_timeout=integer]
[OCF_RESKEY_arping_cache_entries=integer] ethmonitor [start | stop |
status | monitor | meta-data | validate-all]
```

Description

Monitor the vitality of a local network interface. You may setup this RA as a clone resource to monitor the network interfaces on different nodes, with the same interface name. This is not related to the IP address or the network on which a interface is configured. You may use this RA to move resources away from a node, which has a faulty interface or prevent moving resources to such a node. This gives you independent control of the resources, without involving cluster intercommunication. But it requires your nodes to have more than one network interface. The resource configuration requires a monitor operation, because the monitor does the main part of the work. In addition to the resource configuration, you need to configure some location constraints, based on a CIB attribute value. The name of the attribute value is configured in the 'name' option of this RA. Example constraint configuration: location loc_connected_node my_resource_grp rule ="rule_loc_connected_node" -INF: ethmonitor eq 0 The ethmonitor works in 3 different modes to test the interface vitality. 1. call ip to see if the link status is up (if link is down -> error) 2. call ip and watch the RX counter (if packages come around in a certain time -> success) 3. call arping to check whether any of the IPs found in the local ARP cache answers an ARP REQUEST (one answer -> success) 4. return error

Supported Parameters

OCF_RESKEY_interface=Network interface name

The name of the network interface which should be monitored (e.g. eth0).

OCF_RESKEY_name=Attribute name

The name of the CIB attribute to set. This is the name to be used in the constraints.

Defaults to "ethmonitor-'interface_name'".

OCF_RESKEY_multiplier=Multiplier for result variable

Multiplier for the value of the CIB attribute specified in parameter name.

OCF_RESKEY_repeat_count=Monitor repeat count

Specify how often the interface will be monitored, before the status is set to failed.

You need to set the timeout of the monitoring operation to at least repeat_count * repeat_interval

OCF_RESKEY_repeat_interval=Monitor repeat interval in seconds

Specify how long to wait in seconds between the repeat_counts.

OCF_RESKEY_pktnr_timeout=packet counter timeout

Timeout for the RX packet counter. Stop listening for packet counter changes after the given number of seconds.

OCF_RESKEY_arping_count=Number of arpings per IP

Number of ARP REQUEST packets to send for every IP. Usually one ARP REQUEST (arping) is send

OCF_RESKEY_arping_timeout=Timeout for arping per IP

Time in seconds to wait for ARP REQUESTs (all packets of arping_count). This is to limit the time for arp requests, to be able to send requests to more than one node, without running in the monitor operation timeout.

OCF_RESKEY_arping_cache_entries=Number of ARP cache entries to try

Maximum number of IPs from ARP cache list to check for ARP REQUEST (arping) answers. Newest entries are tried first.

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:exports (7)

ocf:exports — Manages NFS exports

Synopsis

```
[OCF_RESKEY_clientspec=string] [OCF_RESKEY_options=string]  
[OCF_RESKEY_directory=string] OCF_RESKEY_fsid=string  
[OCF_RESKEY_unlock_on_stop=boolean]  
[OCF_RESKEY_wait_for_leasetime_on_stop=boolean]  
[OCF_RESKEY_rmtab_backup=string] exportfs [start | stop | monitor | meta-  
data | validate-all]
```

Description

Exports 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 can be a unique positive integer, a UUID, or the special string "root" which is functionally identical to numeric fsid of 0. 0 (root) identifies the export as the root of an NFSv4 pseudofilesystem -- avoid this

setting unless you understand its special status. This value will override any fsid provided via the options parameter.

OCF_RESKEY_unlock_on_stop= Unlock filesystem on stop?

Relinquish NFS locks associated with this filesystem when the resource stops.

Enabling this parameter is highly recommended unless the path exported by this exportfs resource is also exported by a different resource.

OCF_RESKEY_wait_for_leasetime_on_stop= Ride out the NFSv4 lease time on resource stop?

When stopping (unexporting), wait out the NFSv4 lease time. Only after all leases have expired does the NFS kernel server relinquish all server-side handles on the exported filesystem. If this exportfs resource manages an export that resides on a mount point designed to fail over along with the NFS export itself, then enabling this parameter will ensure such failover is working properly. Note that when this parameter is set, your stop timeout MUST accommodate for the wait period. This parameter is safe to disable if none of your NFS clients are using NFS version 4 or later.

OCF_RESKEY_rmtab_backup= Location of the rmtab backup, relative to directory.

Back up those entries from the NFS rmtab that apply to the exported directory, to the specified backup file. The filename is interpreted as relative to the exported directory. This backup is required if clients are connecting to the export via NFSv3 over TCP. Note that a configured monitor operation is required for this functionality. To disable rmtab backups, set this parameter to the special string "none".

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]  
[OCF_RESKEY_fast_stop=boolean] 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 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_RESKEY_fast_stop=fast stop

Normally, we expect no users of the filesystem and the stop operation to finish quickly. If you cannot control the filesystem users easily and want to prevent the stop action from failing, then set this parameter to "no" and add an appropriate timeout for the stop operation.

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]
[OCF_RESKEY_flush_routes=boolean] 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_RESKEY_flush_routes=Flush kernel routing table on stop

Flush the routing table on stop. This is for applications which use the cluster IP address and which run on the same physical host that the IP address lives on. The Linux kernel may force that application to take a shortcut to the local loopback interface, instead of the interface the address is really bound to. Under those circumstances, an application may, somewhat unexpectedly, continue to use connections for some time even after the IP address is deconfigured. Set this parameter in order to immediately disable said shortcut when the IP address goes away.

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_RESKEYImplementation=string] [OCF_RESKEYtarget_iqn=string]  
[OCF_RESKEYlun=integer] [OCF_RESKEYpath=string]  
OCF_RESKEYscsi_id=string OCF_RESKEYscsi_sn=string  
[OCF_RESKEYvendor_id=string] [OCF_RESKEYproduct_id=string]  
[OCF_RESKEYadditional_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_RESKEYImplementation=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_RESKEYtarget_iqn=iSCSI target IQN

The iSCSI Qualified Name (IQN) that this Logical Unit belongs to.

OCF_RESKEYlun=Logical Unit number (LUN)

The Logical Unit number (LUN) exposed to initiators.

OCF_RESKEYpath=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_RESKEYImplementation=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. This is a unique parameter, as it is not allowed to re-use a single username across multiple target instances.

`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_iscsadm=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_iscsadm=iscsadm 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:jboss (7)

ocf:jboss — Manages a JBoss application server instance

Synopsis

```
OCF_RESKEY_resource_name=string OCF_RESKEY_console=string  
[OCF_RESKEY_shutdown_timeout=integer]  
[OCF_RESKEY_kill_timeout=integer] [OCF_RESKEY_user=string]  
[OCF_RESKEY_statusurl=string] [OCF_RESKEY_java_home=string]  
OCF_RESKEY_jboss_home=string [OCF_RESKEY_pstring=string]  
[OCF_RESKEY_run_OPTS=string] [OCF_RESKEY_shutdown_OPTS=string]  
jboss [start | stop | status | monitor | meta-data | validate-all]
```

Description

Resource script for Jboss. It manages a Jboss instance as an HA resource.

Supported Parameters

OCF_RESKEY_resource_name=The name of the resource

The name of the resource. Defaults to the name of the resource instance.

OCF_RESKEY_console=jboss log path

A destination of the log of jboss run and shutdown script.

OCF_RESKEY_shutdown_timeout=shutdown timeout

Timeout for jboss bin/shutdown.sh. We wait for this timeout to expire, then send the TERM and QUIT signals. Finally, the KILL signal is used to terminate the jboss process. You should set the timeout for the stop operation to a value bigger than the sum of the timeout parameters. See also kill_timeout.

OCF_RESKEY_kill_timeout=stop by signal timeout

If bin/shutdown.sh doesn't stop the jboss process, then we send it TERM and QUIT signals, intermittently and once a second. After this timeout expires, if the process is still live, we use the KILL signal. See also shutdown_timeout.

OCF_RESKEY_user=A user name to start a resource.

A user name to start a JBoss.

OCF_RESKEY_statusurl=URL to test in the monitor operation.

URL to test in the monitor operation.

OCF_RESKEY_java_home=Home directory of Java.

Home directory of Java. Defaults to the environment variable JAVA_HOME. If it is not set, then define this parameter.

OCF_RESKEY_jboss_home=Home directory of Jboss.

Home directory of Jboss.

OCF_RESKEY_pstring=pkill/pgrep search string

With this string heartbeat matches for the right process to kill.

OCF_RESKEY_run_opts=options for jboss run.sh

Start options to start Jboss with, defaults are from the Jboss-Doku.

OCF_RESKEY_shutdown_opts=options for jboss shutdown.sh

Stop options to stop Jboss with.

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 makes 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]  
[OCF_RESKEY_partial_activation=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_RESKEY_partial_activation=Activate VG even with partial PV only
If set, the volume group will be activated even only partial of the physical volumes available. It helps to set to true, when you are using mirroring logical volumes. Set to true by default in SLE11SP1 HAE.

ocf:lxc (7)

ocf:lxc — Manages LXC containers

Synopsis

```
OCF_RESKEY_container=string [OCF_RESKEY_config=string]
[OCF_RESKEY_log=string] [OCF_RESKEY_use_screen=boolean] lxc [start |
stop | monitor | validate-all | meta-data]
```

Description

Allows LXC containers to be managed by the cluster. If the container is running "init" it will also perform an orderly shutdown. It is 'assumed' that the 'init' system will do an orderly shutdown if presented with a 'kill -PWR' signal. On a 'sysvinit' this would require the container to have an inittab file containing "p0::powerfail:/sbin/init 0" I have absolutely no idea how this is done with 'upstart' or 'systemd', YMMV if your container is using one of them.

Supported Parameters

OCF_RESKEY_container=Container Name

The unique name for this 'Container Instance' e.g. 'test1'.

OCF_RESKEY_config=The LXC config file.

Absolute path to the file holding the specific configuration for this container e.g. '/etc/lxc/test1/config'.

OCF_RESKEY_log=Container log file

Absolute path to the container log file

OCF_RESKEY_use_screen=Use 'screen' for container 'root console' output

Provides the option of capturing the 'root console' from the container and showing it on a separate screen. To see the screen output run 'screen -r {container name}'

The default value is set to 'false', change to 'true' to activate this option

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 | migrate_to  
| migrate_from | validate-all | meta-data]
```

Description

This OCF compliant 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)

`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:named (7)

ocf:named — Manages a named server

Synopsis

```
[OCF_RESKEY_named=string] [OCF_RESKEY_rndc=string]  
[OCF_RESKEY_host=string] [OCF_RESKEY_named_user=string]  
OCF_RESKEY_named_config=string OCF_RESKEY_named_pidfile=string  
OCF_RESKEY_named_rootdir=string [OCF_RESKEY_named_options=string]  
[OCF_RESKEY_named_keytab_file=string]  
[OCF_RESKEY_monitor_request=string]  
[OCF_RESKEY_monitor_response=string]  
[OCF_RESKEY_monitor_ip=string] named [start | stop | reload | status | monitor  
| meta-data | validate-all | methods]
```

Description

Resource script for named (Bind) server. It manages named as an HA resource.

Supported Parameters

OCF_RESKEY_named=named
Path to the named command.

OCF_RESKEY_rndc=rndc
Path to the rndc command.

OCF_RESKEY_host=host
Path to the host command.

OCF_RESKEY_named_user=named_user
User that should own named process.

OCF_RESKEY_named_config=named_config

Configuration file for named.

OCF_RESKEY_named_pidfile=named_pidfile

PIDFILE file for named.

OCF_RESKEY_named_rootdir=named_rootdir

Directory that named should use for chroot if any.

OCF_RESKEY_named_options=named_options

Options for named process if any.

OCF_RESKEY_named_keytab_file=named_keytab_file

named service keytab file (for GSS-TSIG).

OCF_RESKEY_monitor_request=monitor_request

Request that shall be sent to named for monitoring. Usually an A record in DNS.

OCF_RESKEY_monitor_response=monitor_response

Expected response from named server.

OCF_RESKEY_monitor_ip=monitor_ip

IP Address where named listens.

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:nginx (7)

ocf:nginx — Manages an Nginx web/proxy server instance

Synopsis

```
OCF_RESKEY_configfile=string [OCF_RESKEY_httpd=string]
[OCF_RESKEY_port=integer] [OCF_RESKEY_status10url=string]
[OCF_RESKEY_status10regex=string] [OCF_RESKEY_testclient=string]
[OCF_RESKEY_testurl=string] [OCF_RESKEY_test20regex=string]
[OCF_RESKEY_testconffile=string] [OCF_RESKEY_test20name=string]
[OCF_RESKEY_external_monitor30_cmd=string]
[OCF_RESKEY_options=string] nginx [start | stop | reload | status | monitor |
monitor | monitor | monitor | meta-data | validate-all]
```

Description

This is the resource agent for the Nginx web/proxy server. This resource agent does not monitor POP or IMAP servers, as we don't know how to determine meaningful status for them. 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 nginx resource will end with an error status. The default monitor operation will verify that nginx is running. The level 10 monitor operation by default will try and fetch the /nginx_status page - which is commented out in sample nginx configurations. Make sure that the /nginx_status page works and that the access is restricted to localhost (address 127.0.0.1) plus whatever places _outside the cluster_ you want to monitor the server from. See the status10url and status10regex attributes for more details. The level 20 monitor operation will perform a more complex set of tests from a configuration file. The level 30 monitor operation will run an external command to perform an arbitrary monitoring operation.

Supported Parameters

`OCF_RESKEY_configfile=configuration file path`

The full pathname of the Nginx 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_status10url=url name`

The URL to monitor (the nginx server status page by default) when given a level 10 monitor operation. If left unspecified, it will be inferred from the nginx configuration file, or defaulted to /nginx_status. If you set this, make sure that it succeeds *only* from the localhost (127.0.0.1) and no other cluster nodes. Otherwise, the cluster software may complain about it being active on multiple nodes.

`OCF_RESKEY_status10regex=monitor regular expression`

Regular expression to match in the output of status10url. Case insensitive.

`OCF_RESKEY_testclient=http client`

Client to use to query to Nginx for level 10 and level 20 tests. If not specified, the RA will try to find one on the system. Currently, wget and curl are supported, with curl being preferred. For example, you can set this parameter to "wget" if you prefer that to curl.

`OCF_RESKEY_testurl=Level 10 monitor url`

URL to test. If it does not start with "http", then it's considered to be relative to the document root address.

`OCF_RESKEY_test20regex=Level 20 monitor regular expression`

Regular expression to match in the output of testurl. Case insensitive.

OCF_RESKEY_testconf=Level 20 test configuration file

A file which contains a more complex 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_test20name=Level 20 test name

Name of the test within the test configuration file.

OCF_RESKEY_external_monitor30_cmd=Level 30 test string

Command string to run which implements level 30 monitoring.

OCF_RESKEY_options=nginx start options

Extra options to apply when starting nginx.

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 explicitly.

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_monitor_user=string]
[OCF_RESKEY_monitor_password=string]
[OCF_RESKEY_monitor_sql=string] [OCF_RESKEY_config=string]
[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_monitor_user=monitor_user

PostgreSQL user that pgsql RA will use for monitor operations. If it's not set pgdba user will be used.

OCF_RESKEY_monitor_password=monitor_password

Password for monitor user.

OCF_RESKEY_monitor_sql=monitor_sql

SQL script that will be used for monitor operations.

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 PostgreSQL

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 | val-  
idate-all | meta-data]
```

Description

This script manages Postfix as an OCF resource in a high-availability setup.

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 absolute path to the curl binary

The absolute 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. Alternatively, set to "auto" to manage all devices specified in raidconf.

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 accidentally 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:rsyslog (7)

ocf:rsyslog — rsyslog resource agent

Synopsis

```
OCF_RESKEY_configfile=string [OCF_RESKEY_rsyslog_binary=string]
[OCF_RESKEY_start_opts=string] rsyslog [start | stop | status | monitor | meta-
data | validate-all]
```

Description

This script manages a rsyslog instance as an HA resource.

Supported Parameters

OCF_RESKEY_configfile=Configuration file

This parameter specifies a configuration file for a rsyslog instance managed by this RA.

OCF_RESKEY_rsyslog_binary=rsyslog executable

This parameter specifies rsyslog's executable file.

OCF_RESKEY_start_opts=Start options

This parameter specifies startup options for a rsyslog 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: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 automatically 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 automatically read from the bootstrap.properties file in Java engine 6.40 and 7.00. For Java engine 7.10 and higher 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: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 thruh 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 RA can be used _instead_ of the IPAddr2 or IPAddr RA to send gratuitous ARP for an IP address on a given interface, without adding the address to that interface. For example, if for some reason you wanted to send gratuitous ARP for addresses managed by IPAddr2 or IPAddr on an additional interface.

Supported Parameters

OCF_RESKEY_ip=IP address

The IP address for sending ARP packet.

OCF_RESKEY_nic=NIC

The NIC for sending ARP packet.

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 access 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 |  
validate-all]
```

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 100 seconds. The lock_timeout is calculated by the following formula. lock_timeout = monitor_interval + "The expiration time of

the lock" We suggest 90 seconds as a default value of the "The expiration time of the lock", but you should change it in consideration of access delay to the shared disk and the switch time of the multipath driver. The lock timeout have an impact on start action timeout because start action timeout value is calculated by the following formula. $\text{start timeout} = \text{collision_timeout} + \text{lock_timeout} + \text{"safety margin"}$ The "safety margin" is decided within the range of about 10-20 seconds(It depends on your system requirement).

ocf:slapd (7)

ocf:slapd — Manages a Stand-alone LDAP Daemon (slapd) instance

Synopsis

```
[OCF_RESKEY_slapd=string] [OCF_RESKEY_ldapsearch=string]  
OCF_RESKEY_config=string [OCF_RESKEY_pidfile=string]  
[OCF_RESKEY_user=string] [OCF_RESKEY_group=string]  
OCF_RESKEY_services=string [OCF_RESKEY_watch_suffix=string]  
[OCF_RESKEY_ignore_suffix=string] [OCF_RESKEY_bind_dn=string]  
[OCF_RESKEY_password=string] [OCF_RESKEY_parameters=string]  
[OCF_RESKEY_stop_escalate=integer] slapd [start | stop | monitor | validate-all | meta-data]
```

Description

Resource script for Stand-alone LDAP Daemon (slapd). It manages a slapd instance as an OCF resource.

Supported Parameters

OCF_RESKEY_slapd=Full path to slapd binary
Full path to the slapd binary. For example, "/usr/sbin/slapd".

OCF_RESKEY_ldapsearch=Full path to ldapsearch binary
Full path to the ldapsearch binary. For example, "/usr/bin/ldapsearch".

OCF_RESKEY_config=Full path to configuration directory or file
Full path to a slapd configuration directory or a slapd configuration file. For example, "/etc/ldap/slapd.d" or "/etc/ldap/slapd.conf".

OCF_RESKEY_pidfile=File to read PID from
File to read the PID from; read from olcPidFile/pidfile in config if not set.

OCF_RESKEY_user=User name or id slapd will run with

User name or id slapd will run with. The group id is also changed to this user's gid, unless the group parameter is used to override.

OCF_RESKEY_group=Group name or id slapd will run with

Group name or id slapd will run with.

OCF_RESKEY_services=LDAP (and other scheme) URLs to serve

LDAP (and other scheme) URLs slapd will serve. For example, "ldap://127.0.0.1:389 ldaps:/// ldapi:///"

OCF_RESKEY_watch_suffix=Suffix that will be monitored for availability.

Suffix (database backend) that will be monitored for availability. Multiple suffixes can be specified by providing a space separated list. By providing one or more suffixes here, the ignore_suffix parameter is discarded. All suffixes will be monitored if left blank.

OCF_RESKEY_ignore_suffix=Suffix that will not be monitored for availability.

Suffix (database backend) that will not be monitored for availability. Multiple suffixes can be specified by providing a space separated list. No suffix will be excluded if left blank.

OCF_RESKEY_bind_dn=Distinguished Name used to bind to the LDAP directory for testing.

Distinguished Name used to bind to the LDAP directory for testing. Leave blank to bind to the LDAP directory anonymously.

OCF_RESKEY_password=Password used to bind to the LDAP directory for testing.

Password used to bind to the LDAP directory for testing.

OCF_RESKEY_parameters=Any additional parameters to slapd.

slapd may be called with additional parameters. Specify any of them here.

OCF_RESKEY_stop_escalate=Seconds before stop escalation to KILL

Number of seconds to wait for shutdown (using SIGTERM) before resorting to SIGKILL

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 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 specify 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 required, 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 implements two states

Supported Parameters

```
OCF_RESKEY_state=State file  
Location to store the resource state in
```

ocf:symlink (7)

ocf:symlink — Manages a symbolic link

Synopsis

```
[OCF_RESKEY_link=string] [OCF_RESKEY_target=string]  
[OCF_RESKEY_backup_suffix=string] symlink [start | stop | monitor | meta-  
data | validate-all]
```

Description

This resource agent that manages a symbolic link (symlink). It is primarily intended to manage configuration files which should be enabled or disabled based on where the resource is running, such as cron job definitions and the like.

Supported Parameters

OCF_RESKEY_link=Full path of the symlink

Full path of the symbolic link to be managed. This must obviously be in a filesystem that supports symbolic links.

OCF_RESKEY_target=Full path to the link target

Full path to the link target (the file or directory which the symlink points to).

OCF_RESKEY_backup_suffix=Suffix to append to backup files

A suffix to append to any files that the resource agent moves out of the way because they clash with "link". If this is unset (the default), then the resource agent will simply refuse to create a symlink if it clashes with an existing file.

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_java_opts=string]
OCF_RESKEY_catalina_home=string OCF_RESKEY_catalina_base=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. DEPRECATED

Time-out for stop operation. DEPRECATED

OCF_RESKEY_tomcat_suspend_trialcount=Max retry count for stop operation. DEPRECATED

Maximum number of times to retry stop operation before suspending and killing Tomcat. DEPRECATED. Does not retry.

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_java_opts=Java options parsed to JVM, used on start and stop.

Java JVM options used on start and stop.

OCF_RESKEY_catalina_home=Home directory of Tomcat

Home directory of Tomcat.

OCF_RESKEY_catalina_base=Instance directory of Tomcat, defaults to catalina_home

Instance 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:VIPArp (7)

ocf:VIPArp — 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]
VIPArp [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 information 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_migration_network_suffix=string]
[OCF_RESKEY_monitor_scripts=string]
[OCF_RESKEY_dynamic_utilization=boolean]
[OCF_RESKEY_set_utilization_cpu=boolean]
[OCF_RESKEY_set_utilization_memory=boolean] 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_migration_network_suffix`=Migration network host name suffix

Use a dedicated migration network. The migration URI is composed by adding this parameters value to the end of the node name. If the node name happens to be an FQDN (as opposed to an unqualified host name), insert the suffix immediately prior to the first period (.) in the FQDN. At the moment Qemu/KVM and Xen migration via a dedicated network is supported. Note: Be sure this composed host name is locally resolvable and the associated IP is reachable through the favored network.

`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_RESKEY_dynamic_utilization`=Set utilization of resource when agent monitor

If set, the utilization parameter of resource will be reset if there are difference between resource parameters and system parameters when agent monitor. Otherwise, the resource parameters will be set once when agent start.

`OCF_RESKEY_set_utilization_cpu`=Enable setting cpu of utilization
Enable setting cpu of utilization.

`OCF_RESKEY_set_utilization_memory`=Enable setting memory of utilization
Enable setting memory of utilization.

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=configuration 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=string]  
[OCF_RESKEY_allow_mem_management=boolean]  
[OCF_RESKEY_node_ip_attribute=string]  
[OCF_RESKEY_reserved_Dom0_memory=string]  
[OCF_RESKEY_monitor_scripts=string] Xen [start | stop | migrate_from | mi-  
grate_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_node_ip_attribute=Node attribute containing target IP address`

In case of a live migration, the system will default to using the IP address associated with the hostname via DNS or /etc/hosts. This parameter allows you to specify a node attribute that will be queried instead for the target node, overriding the IP address. This allows you to use a dedicated network for live migration traffic to a specific node. Warning: make very sure the IP address does point to the right node. Or else the live migration will end up somewhere else, greatly confusing the cluster and causing havoc.

`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

A

Example of Setting Up a Simple Testing Resource

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* (page 23) 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 75)
- *Configuring and Managing Cluster Resources (Command Line)* (page 151)

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, “Logging in to a Cluster” (page 76).

- 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 21, *HA OCF Agents* (page 311).

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# resourceprimitive 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 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
```

B

Example Configuration for OCFS2 and cLVM

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"
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 vgl ocf:heartbeat:LVM \
    params volgrpname="cluster-vg" \
    op monitor interval="60" timeout="60"
group base-group dlm o2cb clvm vgl 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 431) can be configured and be running in the cluster regardless of your setup.

C

Cluster Management Tools

High Availability Extension ships with a comprehensive set of tools to assist 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 20, *Troubleshooting* (page 301).

NOTE: Use the `crm` Shell

These tools are for experts only. In most cases the `crm` shell is the recommended way of managing your cluster.

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 the `crm_mon` man page 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 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 the `cibadmin` man page 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`. See the `crm_diff` man page 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 the `crm_attribute` man page 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 the High Availability Extension 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 the `crm_verify` man page 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 the `crm_resource` man page 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 the `crm_failcount` man page 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 the `crm_standby` man page for a detailed introduction to this tool's usage and command syntax.

D

Upgrading Your Cluster to the Latest Product Version

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.

D.1 Upgrading from SLES 10 to SLE HA 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 438)
2. Updating to SUSE Linux Enterprise 11 (page 439)
3. Testing the Conversion (page 440)
4. Converting the Data (page 440)

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.

D.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 D.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 D.1.3, “Data Conversion” (page 439)), 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`.

D.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 D.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 3.3, “Installation as Add-on” (page 27).

D.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 set-ups 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 D.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 D.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 3.5.7, “Bringing the Cluster Online” (page 43).

After the upgrade process, reverting back to SUSE Linux Enterprise Server 10 is not supported.

D.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.

D.2 Upgrading from SLE HA 11 to SLE HA 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 D.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.

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.3 Upgrading from SLE HA 11 SP1 to SLE HA 11 SP2

Migrating an existing cluster from SUSE Linux Enterprise High Availability Extension 11 SP1 to 11 SP2 is done via a `rolling upgrade`, similar to the upgrade procedure from version 11 to 11 SP1.

Proceed as described in Procedure D.5, “Performing a Rolling Upgrade” (page 442) with the following two deviations:

- In Step 3 (page 442), upgrade from SUSE Linux Enterprise Server 11 SP1 to SUSE Linux Enterprise Server 11 SP2 and from SUSE Linux Enterprise High Availability Extension 11 SP1 to SUSE Linux Enterprise High Availability Extension 11 SP2.

As Xen Hypervisor is discontinued for 32-bit architectures, you might need to solve dependencies for the package `drbd-xen` manually. Note that cross-platform clusters are not supported.

- Because of the kernel update shipped with SP2, reboot the node between Step 3 (page 442) and Step 4 (page 442).

IMPORTANT: Time Limit for Rolling Upgrade

The new features shipped with SUSE Linux Enterprise High Availability Extension 11 SP2 will only be available after *all* cluster nodes have been upgraded to the latest product version. Mixed SP1/SP2 clusters are only supported for a short time frame during the rolling upgrade. Finish the rolling upgrade within one week.

E

What's New?

The most important software modifications from version to version are outlined 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.

For more details and the most recent information, refer to the release notes of the respective product version. They are available in the installed system at `/usr/share/doc/release-notes`.

E.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.

E.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).

E.1.2 Changed Features and Functions

Naming Conventions for Resource and Cluster 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 446).

Arguments for Command Line Tools

The arguments for command-line tools have been made consistent. See also “Naming Conventions for Resource and Cluster Options” (page 447).

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.

E.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 the `crm_resource` man page.

Setting Global Defaults

Resource and operation defaults are no longer read from `crm_config`.

E.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 in-

formation 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 (Hawk), 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.3 Version 11 SP1 to Version 11 SP2

Multi-Site Clusters (Geo Clusters) (page 215)

Apart from local clusters and metro area clusters, SUSE® Linux Enterprise High Availability Extension 11 SP2 also supports multi-site clusters. That means you can have multiple, geographically dispersed sites with a local cluster each. Failover between these clusters is coordinated by a higher level entity, the so-called `booth`. Support for multi-site clusters is available as a separate option to SUSE Linux Enterprise High Availability Extension.

Access Control Lists (page 193)

For defining fine-grained access rights to any part of the cluster configuration ACLs are supported. If this feature is enabled in the CRM, the available functions in the cluster management tools depend on the role and access rights assigned to a user.

Automatic Cluster Setup (`sleha-bootstrap`) (page 28)

For quick and easy cluster setup, use the bootstrap scripts `sleha-init` and `sleha-join` to get a one-node cluster up and running in a few minutes and to make other nodes join, respectively. Any options set during the bootstrap process can be modified later with the YaST cluster module.

Corosync Unicast Mode

While multicast is still default, using unicast for the communication between nodes is now also supported. For more information, refer to Section 3.5.2, “Defining the Communication Channels” (page 32).

HA Web Konsole (Hawk)

Hawk's functionality has been considerably extended. Now you can configure global cluster properties, basic and advanced types of resources, constraints and resource monitoring. For detailed analysis of the cluster status, Hawk generates a cluster report (`hb_report`). View the cluster history or explore potential failure scenarios with the simulator. For details, refer to Chapter 6, *Configuring and Managing Cluster Resources (Web Interface)* (page 109).

Resource Templates (page 55)

To ease configuration of similar resources, all cluster management tools now let you define resource templates that can be referenced in primitives or certain types of constraints.

Virtualization and Cloud Integration

For placing resources based on load impact, the High Availability Extension now offers automatic detection of both the capacity of a node and the capacities a resource requires. The minimal requirements of a virtual machine (for example, the memory assigned to a Xen or KVM guest or the number of CPU cores) can be detected by a resource agent. Utilization attributes (used to define the requirements or capacity) will automatically be added to the CIB. For more information, refer to Section 4.4.6, “Placing Resources Based on Their Load Impact” (page 71).

To protect a node's network connection from being overloaded by a large number of parallel Xen or KVM live migrations, a new global cluster property has been

introduced: `migration-limit`. It allows you to limit the number of migration jobs that the TE may execute in parallel on a node. By default, it is set to `-1`, which means the number of parallel migrations is unlimited.

conntrack Tools

To synchronize the connection status between cluster nodes, the High Availability Extension uses the `conntrack-tools`. They allow interaction with the in-kernel Connection Tracking System for enabling *stateful* packet inspection for iptables. For more information, refer to Section 3.5.6, “Synchronizing Connection Status Between Cluster Nodes” (page 42).

Parallel SSH (`pssh`)

To execute commands on all cluster nodes without having to log in to each node, use `pssh`. For more information, refer to Section 20.5, “Miscellaneous” (page 306).

```
crm resource secret
```

To set passwords for STONITH or other resources independent of `cib.xml`, use `crm resource secret`. For more information, refer to Section 7.5, “Setting Passwords Independent of `cib.xml`” (page 172).

Samba Clustering

The CTDB functionality to join Active Directory Domains has been improved. For more information, refer to Section 18.4, “Joining Active Directory Domains” (page 285).

Disaster Recovery with ReaR (page 291)

ReaR (Relax and Recover) is an administrator tool-set for creating disaster recovery images. The disaster recovery information can either be stored via the network or locally on hard disks, USB devices, DVD/CD-R, tape or similar. The backup data is stored on a network file system (NFS).

Quotas on OCFS2

To use quotas on OCFS2 file systems, create and mount the files system with the appropriate quota features or mount options, respectively: `usrquota` (quota for individual users) or `grpquota` (quota for groups).

Terminology

active/active, active/passive

A concept of 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 node waits for the active node to fail. Active-active means that each node is active and passive at the same time.

arbitrator

Additional instance in a multi-site cluster that helps to reach consensus about decisions such as failover of resources across sites. Arbitrators are single machines that run a booth instance in a special mode.

AutoYaST

AutoYaST is a system for installing one or more SUSE Linux Enterprise systems automatically and without user intervention.

booth

The instance that manages the failover process between the sites of a multi-site cluster. It guarantees that the cluster resources will be highly available across different cluster sites. This is achieved by using so-called tickets that are treated as failover domain between cluster sites, in case a site should be down.

booth daemon (boothd)

Each of the participating clusters and arbitrators in a multi-site cluster runs a service, the boothd. It connects to the booth daemons running at the other sites and exchanges connectivity details.

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 (cluster options, nodes, resources, constraints and the relationship to each other). It is written in XML and resides in memory. A master CIB is kept and maintained on the designated coordinator (DC) and replicated to the other nodes. Normal read and write operations on the CIB are serialized through the master CIB.

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 are split in partitions but still active. They can only communicate with nodes in the same partition and are unaware of the separated nodes. As the loss of the nodes on the other partition cannot be confirmed, a split brain scenario develops (see also split brain (page 457)).

cluster resource manager (CRM)

The main management entity responsible for coordinating all non-local interactions. The High Availability Extension uses Pacemaker as CRM. Each node of the cluster has its own CRM instance, 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.

cluster resource manager daemon (crmd)

The CRM is implemented as daemon, crmd. It has an instance on each cluster node. All cluster decision-making is centralized by electing one of the crmd instances to act as a master. If the elected crmd process fails (or the node it is on), a new one is established.

concurrency violation

A resource that should be running on only one node in the cluster is running on several nodes.

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.

Csync2

A synchronization tool that can be used to replicate configuration files across all nodes in the cluster.

designated coordinator (DC)

One CRM in the cluster is elected as the Designated Coordinator (DC). 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. The DC is also the node where the master copy of the CIB is kept. All other nodes get their configura-

tion 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.

failover domain

A named subset of cluster nodes that are eligible to run a cluster service in the event of a node failure.

fencing

Describes the concept of preventing access to a shared resource by isolated or failing cluster members. Should a cluster node fail, it will be shut down or reset to prevent it from causing trouble. This way, resources are locked out of a node whose status is uncertain.

geographically dispersed cluster (geo cluster)

See multi-site cluster (page 456).

local cluster

A single cluster in one location (for example, all nodes are located in one data center). Network latency can be neglected. Storage is typically accessed synchronously by all nodes.

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 these operations. The LRM is “dumb” in that it does not know of any policy. It needs the DC to tell it what to do.

metro cluster

A single cluster that can stretch over multiple buildings or data centers, with all sites connected by fibre channel. Network latency is usually low (<5 ms for distances of approximately 20 miles). Storage is frequently replicated (mirroring or synchronous replication).

multicast

A technology used for a one-to-many communication within a network that can be used for cluster communication. Corosync supports both multicast and unicast.

multi-site cluster

Consists of multiple, geographically dispersed sites with a local cluster each. The sites communicate via IP. Failover across the sites is coordinated by a higher-level entity, the booth. Multi-site clusters have to cope with limited network bandwidth and high latency. Storage is replicated asynchronously.

node

Any computer (real or virtual) that is a member of a cluster and invisible to the user.

policy engine (PE)

The policy engine computes the actions that need to be taken to implement policy changes in the CIB. The PE also produces a transition graph containing a list of (resource) actions and dependencies to achieve the next cluster state. The PE always runs on the DC.

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 Pacemaker. 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 (for example, to start, stop or monitor a resource). The High Availability Extension supports

three different kinds of resource agents: OCF (Open Cluster Framework) resource agents, LSB (Linux Standards Base) resource agents (Standard LSB init scripts), and Heartbeat resource agents (Heartbeat v1 resources). For more information, refer to Section 4.2.2, “Supported Resource Agent Classes” (page 52).

shared disk file exclusiveness (SFEX)

SFEX provides storage protection over SAN.

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 that the two nodes contain different data.

STONITH

The acronym for “Shoot the other node in the head”, which refers to the fencing mechanism that shuts down a misbehaving node to prevent it from causing trouble in a cluster.

STONITH block device (SBD)

In an environment where all nodes have access to shared storage, a small partition is used for disk-based fencing.

ticket

A component used in multi-site clusters. A ticket grants the right to run certain resources on a specific cluster site. A ticket can only be owned by one site at a time. Resources can be bound to a certain ticket by dependencies. Only if the defined ticket is available at a site, the respective resources are started. Vice versa, if the ticket is removed, the resources depending on that ticket are automatically stopped.

Totem redundant ring protocol (RRP)

Allows the use of multiple redundant local-area networks for resilience against partial or total network faults. This way, cluster communication can still be kept up as long as a single network is operational. A logical token-passing ring is imposed

on all participating nodes to deliver messages in a reliable and sorted manner. A node is allowed to broadcast a message only if it holds the token.

unicast

A technology for sending messages to a single network destination. Corosync supports both multicast and unicast. In Corosync, unicast is implemented as UDP-unicast (UDPU).

F

GNU Licenses

This appendix contains the GNU General Public License version 2 and the GNU Free Documentation License version 1.2.

GNU Free Documentation License

Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. 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. Secondly, 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.

1. 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.

2. 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.

3. 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.

4. 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.

5. 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".

6. 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.

7. 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.

8. 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.

9. 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.

10. 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

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.

