



Administration Guide

SUSE Linux Enterprise High Availability
Extension 12



Administration Guide

SUSE Linux Enterprise High Availability Extension 12

by Tanja Roth and Thomas Schraitle

Publication date: Sep 25 2014

SUSE Linux Products GmbH

Maxfeldstr. 5


90409 Nürnberg

GERMANY

<https://www.suse.com/documentation> 

Copyright © 2006–2017 SUSE LLC 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 SUSE or 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 SUSE or 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 SUSE LLC, its affiliates, the authors nor the translators shall be held liable for possible errors or the consequences thereof.

Contents

About This Guide **xii**

I INSTALLATION AND SETUP **1**

1 Product Overview **2**

1.1 Availability as Add-On/Extension **2**

1.2 Key Features **3**

Wide Range of Clustering Scenarios **3** • Flexibility **3** • Storage and Data Replication **4** • Support for Virtualized Environments **4** • Support of Local, Metro, and GEO Clusters **4** • Resource Agents **5** • User-friendly Administration Tools **5**

1.3 Benefits **6**

1.4 Cluster Configurations: Storage **10**

1.5 Architecture **12**

Architecture Layers **12** • Process Flow **15**

2 System Requirements and Recommendations **17**

2.1 Hardware Requirements **17**

2.2 Software Requirements **18**

2.3 Storage Requirements **18**

2.4 Other Requirements and Recommendations **19**

3 Installation and Basic Setup **22**

3.1 Definition of Terms **22**

3.2 Overview **24**

- 3.3 Installation as Add-on 25
- 3.4 Automatic Cluster Setup (ha-cluster-bootstrap) 26
- 3.5 Manual Cluster Setup (YaST) 31
 - YaST Cluster Module 31 • Defining the Communication Channels 33 • Defining Authentication Settings 37 • Transferring the Configuration to All Nodes 38 • Synchronizing Connection Status Between Cluster Nodes 42 • Configuring Services 44 • Bringing the Cluster Online 45
- 3.6 Mass Deployment with AutoYaST 46

II CONFIGURATION AND ADMINISTRATION 49

4 Configuration and Administration Basics 50

- 4.1 Global Cluster Options 50
 - Overview 50 • Option no-quorum-policy 51 • Option stonith-enabled 52
- 4.2 Cluster Resources 52
 - Resource Management 52 • Supported Resource Agent Classes 53 • Types of Resources 55 • Resource Templates 56 • Advanced Resource Types 56 • Resource Options (Meta Attributes) 59 • Instance Attributes (Parameters) 62 • Resource Operations 65 • Timeout Values 67
- 4.3 Resource Monitoring 68
- 4.4 Resource Constraints 69
 - Types of Constraints 70 • Scores and Infinity 73 • Resource Templates and Constraints 74 • Failover Nodes 74 • Failback Nodes 75 • Placing Resources Based on Their Load Impact 76 • Grouping Resources by Using Tags 79
- 4.5 Managing Services on Remote Hosts 80
 - Monitoring Services on Remote Hosts with Monitoring Plug-ins 80 • Managing Services on Remote Nodes with `pacemaker_remote` 82
- 4.6 Monitoring System Health 83

4.7 Maintenance Mode 84

4.8 For More Information 86

5 Configuring and Managing Cluster Resources (Web Interface) 87

5.1 Hawk—Overview 87

Starting Hawk and Logging In 87 • Main Screen: Cluster Status 89

5.2 Configuring Global Cluster Options 94

5.3 Configuring Cluster Resources 95

Configuring Resources with the Setup Wizard 96 • Creating Simple Cluster Resources 100 • Creating STONITH Resources 102 • Using Resource Templates 103 • Configuring Resource Constraints 105 • Specifying Resource Failover Nodes 110 • Specifying Resource Failback Nodes (Resource Stickiness) 111 • Configuring Placement of Resources Based on Load Impact 111 • Configuring Resource Monitoring 113 • Configuring a Cluster Resource Group 115 • Configuring a Clone Resource 116

5.4 Managing Cluster Resources 117

Starting Resources 118 • Cleaning Up Resources 119 • Removing Cluster Resources 120 • Migrating Cluster Resources 120 • Using Maintenance Mode 122 • Viewing the Cluster History 124 • Exploring Potential Failure Scenarios 128 • Generating a Cluster Report 131

5.5 Monitoring Multiple Clusters 131

5.6 Hawk for GEO Clusters 134

5.7 Troubleshooting 134

6 Configuring and Managing Cluster Resources (Command Line) 136

6.1 crmsh—Overview 136

Getting Help 137 • Executing crmsh's Subcommands 138 • Displaying Information about OCF Resource Agents 140 • Using Configuration Tem-

	plates 141 • Testing with Shadow Configuration 144 • Debugging Your Configuration Changes 145 • Cluster Diagram 145
6.2	Managing Corosync Configuration 145
6.3	Configuring Global Cluster Options 147
6.4	Configuring Cluster Resources 148
	Creating Cluster Resources 148 • Creating Resource Templates 149 • Creating a STONITH Resource 150 • Configuring Resource Constraints 152 • Specifying Resource Failover Nodes 155 • Specifying Resource Failback Nodes (Resource Stickiness) 155 • Configuring Placement of Resources Based on Load Impact 155 • Configuring Resource Monitoring 158 • Configuring a Cluster Resource Group 159 • Configuring a Clone Resource 160
6.5	Managing Cluster Resources 161
	Starting a New Cluster Resource 161 • Cleaning Up Resources 161 • Removing a Cluster Resource 162 • Migrating a Cluster Resource 163 • Grouping/Tagging Resources 163 • Using Maintenance Mode 163 • Getting Health Status 165
6.6	Setting Passwords Independent of <code>cib.xml</code> 165
6.7	Retrieving History Information 166
6.8	For More Information 168
7	Adding or Modifying Resource Agents 169
7.1	STONITH Agents 169
7.2	Writing OCF Resource Agents 169
7.3	OCF Return Codes and Failure Recovery 171
8	Fencing and STONITH 173
8.1	Classes of Fencing 173
8.2	Node Level Fencing 174
	STONITH Devices 174 • STONITH Implementation 175

8.3	STONITH Configuration	176
	Example STONITH Resource Configurations	176 • Constraints Versus Clones 180
8.4	Monitoring Fencing Devices	180
8.5	Special Fencing Devices	181
8.6	Basic Recommendations	182
8.7	For More Information	183
9	Access Control Lists	184
9.1	Requirements and Prerequisites	184
9.2	Enabling Use of ACLs In Your Cluster	185
9.3	The Basics of ACLs	185
	Setting ACL Rules via XPath Expressions	186 • Setting ACL Rules via Tag Abbreviations 188
9.4	Configuring ACLs with Hawk	189
9.5	Configuring ACLs with crmsh	190
9.6	For More Information	192
10	Network Device Bonding	193
10.1	Configuring Bonding Devices with YaST	193
10.2	Hotplugging of Bonding Slaves	196
10.3	For More Information	198
11	Load Balancing	199
11.1	Conceptual Overview	199
11.2	Configuring Load Balancing with Linux Virtual Server	201
	Director	201 • User Space Controller and Daemons 201 • Packet Forwarding 202 • Scheduling Algorithms 202 • Setting Up IP Load Balancing with YaST 203 • Further Setup 208

11.3	Configuring Load Balancing with HAProxy	208
11.4	For More Information	212
12	GEO Clusters (Multi-Site Clusters)	213
III	STORAGE AND DATA REPLICATION	214
13	OCFS2	215
13.1	Features and Benefits	215
13.2	OCFS2 Packages and Management Utilities	216
13.3	Configuring OCFS2 Services and a STONITH Resource	217
13.4	Creating OCFS2 Volumes	219
13.5	Mounting OCFS2 Volumes	221
13.6	Configuring OCFS2 Resources With Hawk	223
13.7	Using Quotas on OCFS2 File Systems	225
13.8	For More Information	225
14	GFS2	226
14.1	GFS2 Packages and Management Utilities	226
14.2	Configuring GFS2 Services and a STONITH Resource	227
14.3	Creating GFS2 Volumes	228
14.4	Mounting GFS2 Volumes	230
15	DRBD	232
15.1	Conceptual Overview	232
15.2	Installing DRBD Services	234
15.3	Configuring the DRBD Service	234
15.4	Testing the DRBD Service	241

- 15.5 Tuning DRBD 243
- 15.6 Troubleshooting DRBD 243
 - Configuration 243 • Hostnames 244 • TCP Port 7788 244 • DRBD Devices Broken after Reboot 245
- 15.7 For More Information 245
- 16 Cluster Logical Volume Manager (cLVM) 246**
- 16.1 Conceptual Overview 246
- 16.2 Configuration of cLVM 246
 - Configuring Cmirrord 247 • Creating the Cluster Resources 249 • Scenario: cLVM With iSCSI on SANs 251 • Scenario: cLVM With DRBD 256
- 16.3 Configuring Eligible LVM2 Devices Explicitly 258
- 16.4 For More Information 259
- 17 Storage Protection 260**
- 17.1 Storage-based Fencing 260
 - Overview 260 • Number of SBD Devices 261 • Setting Up Storage-based Protection 262
- 17.2 Ensuring Exclusive Storage Activation 270
 - Overview 270 • Setup 270
- 17.3 For More Information 272
- 18 Samba Clustering 273**
- 18.1 Conceptual Overview 273
- 18.2 Basic Configuration 274
- 18.3 Joining an Active Directory Domain 278
- 18.4 Debugging and Testing Clustered Samba 280
- 18.5 For More Information 282

19 Disaster Recovery with Rear (Relax-and-Recover) 283

- 19.1 Conceptual Overview 283
- 19.2 Preparing for the Worst Scenarios: Disaster Recovery Plans 285
- 19.3 Setting Up Rear 285
- 19.4 Storing your Backup on a NFS Server 286
- 19.5 Using the YaST Rear Module 288
- 19.6 For More Information 289

IV APPENDIX 290

A Troubleshooting 291

- A.1 Installation and First Steps 291
- A.2 Logging 292
- A.3 Resources 294
- A.4 STONITH and Fencing 295
- A.5 Miscellaneous 296
- A.6 For More Information 299

B Naming Conventions 300

C Cluster Management Tools 301

D Upgrading Your Cluster and Updating Software Packages 303

- D.1 Terminology 303
- D.2 Upgrading your Cluster to the Latest Product Version 304
 - Upgrading from SLE HA 11 SP3 to SLE HA 12 304
- D.3 Updating Software Packages on Cluster Nodes 306

D.4 For More Information 307

E Documentation Updates 308

E.1 October, 2014 (Initial Release of SUSE Linux Enterprise High Availability Extension 12) 308

Terminology 314

F GNU Licenses 321

F.1 GNU Free Documentation License 321

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 several YaST modules as graphical user interface (GUI), the Web interface Hawk, and crmsh (the crm shell) as command line interface (CLI).

This guide is intended for administrators who need to set up, configure, and maintain High Availability (HA) clusters. Both approaches (graphical Web interface 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 Web interface (HA Web Konsole), or the crmsh command line interface. 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. 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 the cluster-aware file systems OCFS2 and GFS2 and the clustered Logical Volume Manager (cLVM). For replication of your data, use DRBD* 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.

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/doc/> .

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 SUSE 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/doc/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
- , : a key to press or a key combination; keys are shown in uppercase as on a keyboard
- *File*, *File* > *Save As*: menu items, buttons
-  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.

For an overview of naming conventions with regards to cluster nodes and names, resources, and constraints, see [Appendix B, Naming Conventions](#).

I Installation and Setup

- 1 Product Overview 2
- 2 System Requirements and Recommendations 17
- 3 Installation and Basic Setup 22

1 Product Overview

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

It ships with essential monitoring, messaging, and cluster resource management functionality (supporting failover, failback, and migration (load balancing) of individually managed cluster resources).

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*.

1.1 Availability as Add-On/Extension

The High Availability Extension is available as add-on to SUSE Linux Enterprise Server 12. Support for geographically dispersed clusters (GEO clusters) is available as a separate extension to the High Availability Extension, called GEO Clustering for SUSE Linux Enterprise High Availability Extension.

1.2 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:

1.2.1 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)
- GEO 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.

1.2.2 Flexibility

The High Availability Extension ships with Corosync 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.

1.2.3 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.

1.2.4 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 12 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.

1.2.5 Support of Local, Metro, and GEO Clusters

SUSE Linux Enterprise High Availability Extension has been extended to support different geographical scenarios. Support for geographically dispersed clusters (GEO clusters) is available as a separate extension to High Availability Extension, called GEO Clustering for 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).

GEO Clusters (Multi-Site Clusters)

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. GEO 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.

1.2.6 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 an overview of Open Cluster Framework (OCF) resource agents included with your product, use the `crm ra` command as described in *Section 6.1.3, "Displaying Information about OCF Resource Agents"*.

1.2.7 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"*. 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 \(YaST\)”](#).
- DRBD: Configuration of a Distributed Replicated Block Device.
- IP Load Balancing: Configuration of load balancing with Linux Virtual Server or HAProxy. For details, refer to [Chapter 11, Load Balancing](#).

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 5, Configuring and Managing Cluster Resources \(Web Interface\)](#).

crm Shell

A powerful unified command line interface to configure resources and execute all monitoring or administration tasks. For details, refer to [Chapter 6, Configuring and Managing Cluster Resources \(Command Line\)](#).

1.3 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.

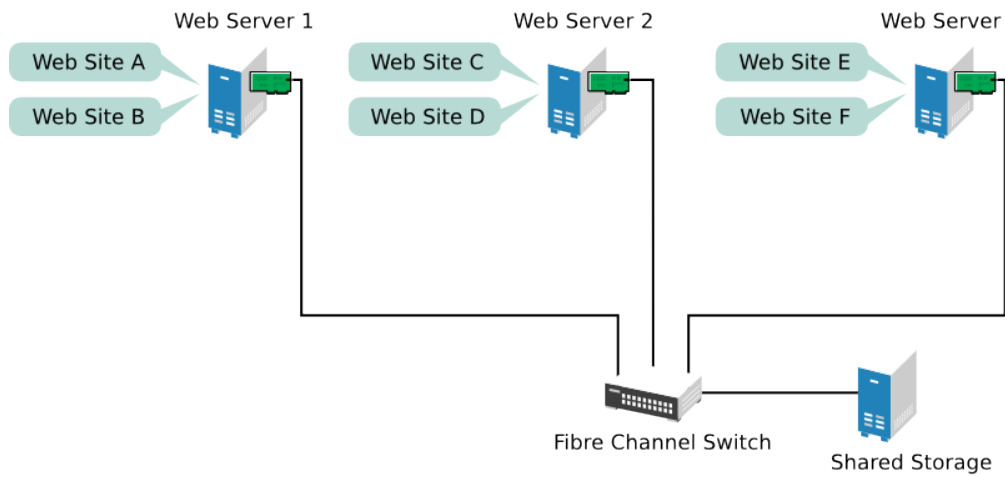


FIGURE 1.1: THREE-SERVER CLUSTER

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.

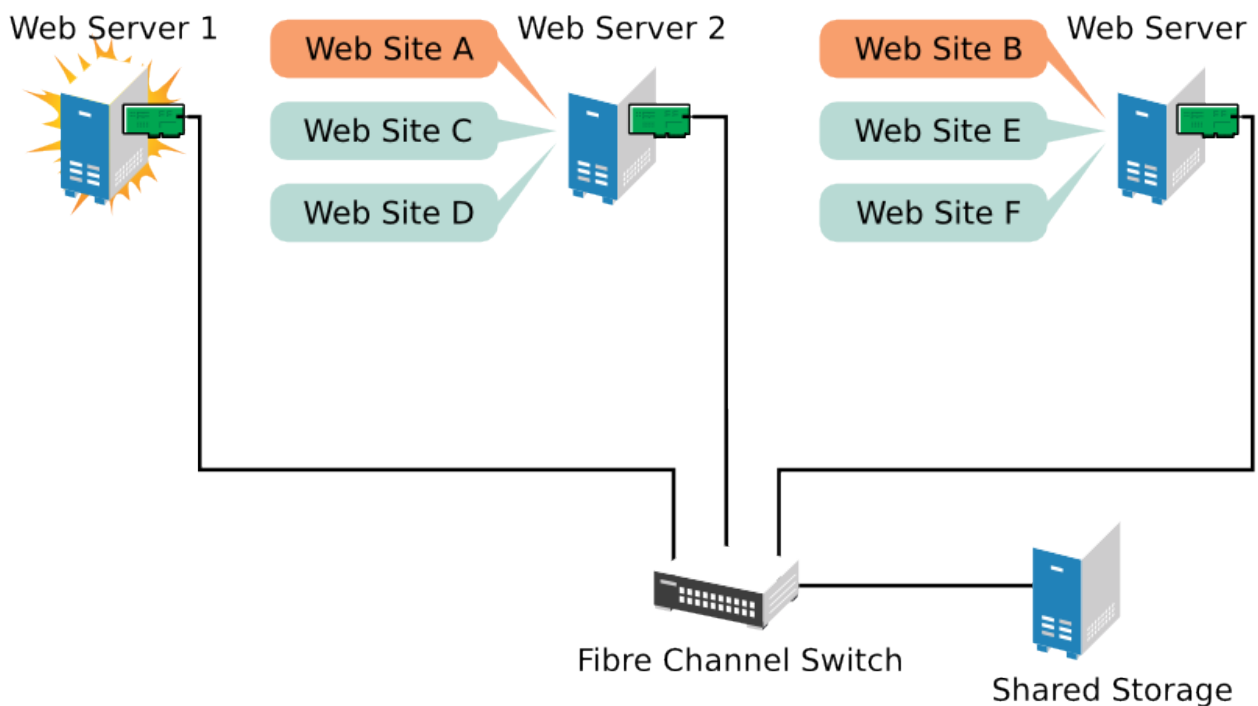


FIGURE 1.2: THREE-SERVER CLUSTER AFTER ONE SERVER FAILS

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.

1.4 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.

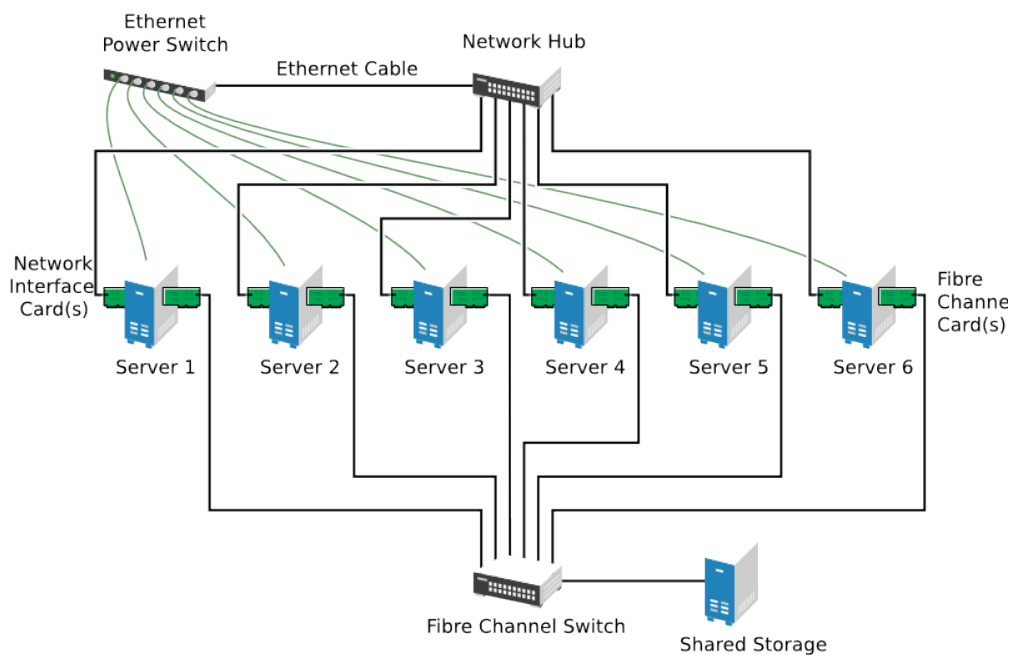


FIGURE 1.3: TYPICAL FIBRE CHANNEL CLUSTER CONFIGURATION

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.

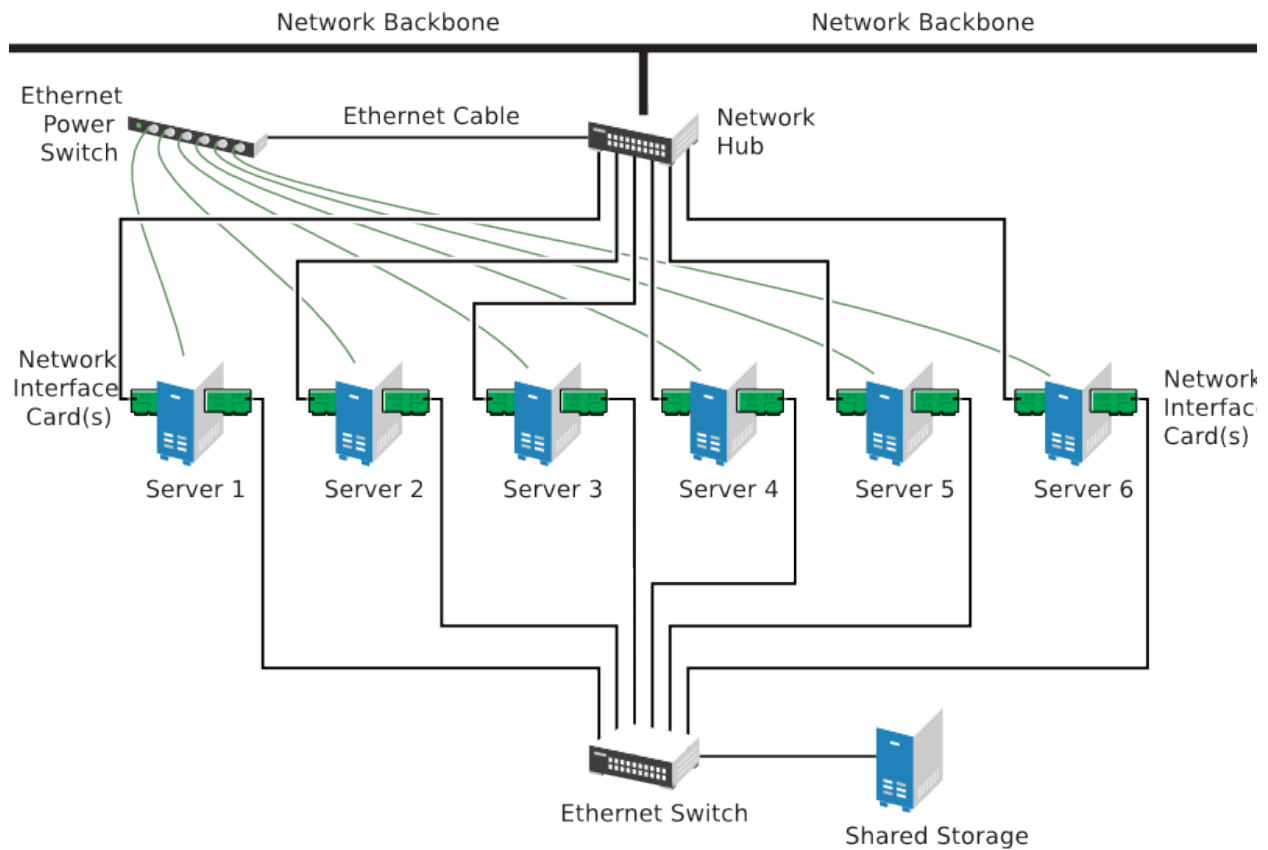


FIGURE 1.4: TYPICAL iSCSI CLUSTER CONFIGURATION

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.

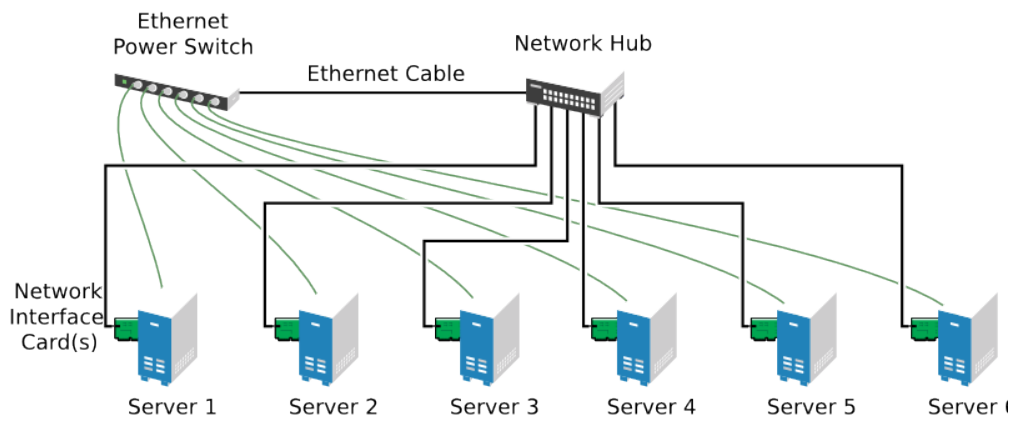


FIGURE 1.5: TYPICAL CLUSTER CONFIGURATION WITHOUT SHARED STORAGE

1.5 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.

1.5.1 Architecture Layers

The High Availability Extension has a layered architecture. *Figure 1.6, "Architecture"* illustrates the different layers and their associated components.

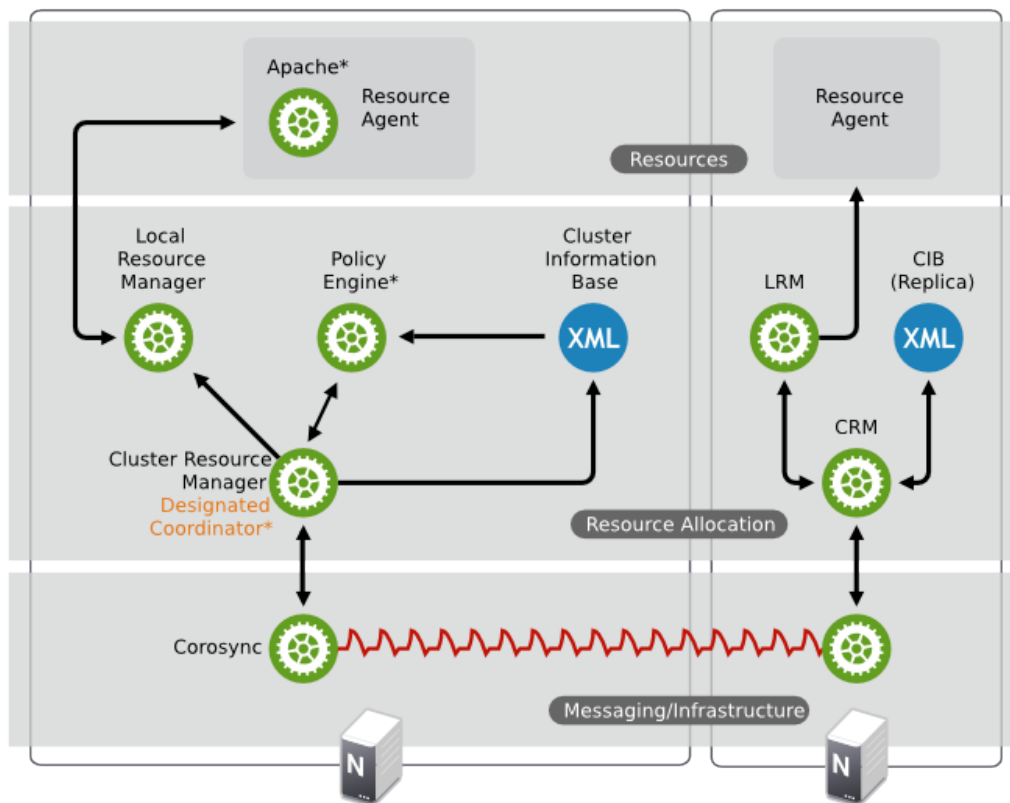


FIGURE 1.6: ARCHITECTURE

1.5.1.1 Messaging and Infrastructure Layer

The primary or first layer is the messaging/infrastructure layer, also known as the Corosync 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.

1.5.1.2 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 1.5.1.3, "Resource Layer"*) 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). The LRM is the authoritative source for all resource-related information on its local node.

1.5.1.3 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.

1.5.2 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 Web interface 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.

2 System Requirements and Recommendations

The following section informs you about system requirements, and some prerequisites for SUSE® Linux Enterprise High Availability Extension. It also includes recommendations for cluster setup.

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”](#). 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 per cluster node. 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. See [Section 2.3, “Storage Requirements”](#).
- A *STONITH* mechanism. 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. This is the only reliable way to ensure that no data corruption is performed by nodes that hang and only appear to be dead.

2.2 Software Requirements

Ensure that the following software requirements are met:

- SUSE® Linux Enterprise Server 12 (with all available online updates) is installed on all nodes that will be part of the cluster.
- SUSE Linux Enterprise High Availability Extension 12 (with all available online updates) is installed on all nodes that will be part of the cluster.
- If you want to use GEO clusters, make sure that GEO Clustering for SUSE Linux Enterprise High Availability Extension 12 (with all available online updates) is installed on all nodes that will be part of the cluster.

2.3 Storage Requirements

To make data highly available, a shared disk system (Storage Area Network, or SAN) is recommended for your cluster. 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 and Recommendations

For a supported and useful High Availability setup, consider the following recommendations:

Number of Cluster Nodes

Each cluster must consist of at least two cluster nodes.

Important: Odd Number of Cluster Nodes

It is strongly recommended to use an *odd* number of cluster nodes with a *minimum* of three nodes.

A cluster needs *quorum* to keep services running. Therefore a three-node cluster can tolerate only failure of one node at a time, whereas a five-node cluster can tolerate failures of two nodes etc.

STONITH

Important: No Support Without STONITH

A cluster without STONITH is not supported.

For a supported High Availability setup, ensure the following:

- Each node in the High Availability cluster must have at least one STONITH device (usually a piece of hardware). We strongly recommend multiple STONITH devices per node, unless SBD is used. SBD provides a way to enable STONITH and fencing in clusters without external power switches, but it requires shared storage.
- The global cluster options `stonith-enabled` and `startup-fencing` must be set to `true`. As soon as you change them, you will lose support.

Redundant Communication Paths

For a supported High Availability setup, it is required to set up cluster communication via two or more redundant paths. This can be done via:

- *Network Device Bonding*.
- A second communication channel in Corosync. For details, see *Procedure 3.6, "Defining a Redundant Communication Channel"*.

If possible, choose network device bonding.

Time Synchronization

Cluster nodes should synchronize to an NTP server outside the cluster. For more information, see the *Administration Guide*, for SUSE Linux Enterprise Server 12, available at <http://www.suse.com/doc/>. Refer to the chapter *Time Synchronization with NTP*.

If nodes are not synchronized, log files and 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 *Administration Guide* for SUSE Linux Enterprise Server 12, available at <http://www.suse.com/doc/>. Refer to chapter *Basic Networking > Configuring Hostname and DNS*.

Storage Requirements

Some services may require shared storage. For requirements, see *Section 2.3, "Storage Requirements"*. You can also use an external NFS share or DRBD. If using an external NFS share, it must be reliably accessible from all cluster nodes via redundant communication paths. See *Redundant Communication Paths*.

When using SBD as STONITH device, additional requirements apply for the shared storage. For details, see http://linux-ha.org/wiki/SBD_Fencing, section *Requirements*.

SSH

All cluster nodes must be able to access each other via SSH. Tools like `hb_report` or `crm_report` (for troubleshooting) and the history explorer require passwordless SSH access between the nodes, otherwise they can only collect data from the current node. In case you use a non-standard SSH port, use the `-X` option (see man page). For example, if your SSH port is `3479`, invoke a `hb_report` with:

```
root # hb_report -X "-p 3479" [...]
```



Note: Regulatory Requirements

If passwordless SSH access does not comply with regulatory requirements, you can use the following work-around for crm_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 crm_report from command line with the -u option to specify the user's name. For more information, see the crm_report man page.

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

3 Installation and Basic Setup

This chapter describes how to install and set up SUSE® Linux Enterprise High Availability Extension 12 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 and Updating Software Packages](#) 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 that 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. Corosync supports both multicast and unicast. 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 be used for multicasting by the Corosync executive. The IP address can either be IPv4 or IPv6. If IPv6 networking is used, node IDs must be specified. You can use any multicast address in your private network.

Multicast Port (mcastport)

The port to use for cluster communication. Corosync uses two ports: the specified mcastport for receiving multicast, and mcastport -1 for sending multicast.

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).

Bind Network Address (`bindnetaddr`)

The network address the Corosync executive should bind to. To ease sharing configuration files across the cluster, Corosync uses network interface netmask to mask only the address bits that are used for routing the network. For example, if the local interface is `192.168.5.92` with netmask `255.255.255.0`, set `bindnetaddr` to `192.168.5.0`. If the local interface is `192.168.5.92` with netmask `255.255.255.192`, set `bindnetaddr` to `192.168.5.64`.



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)

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. Corosync supports the Totem Redundant Ring Protocol. 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. 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 set to `none`, RRP is disabled.

Csync2

A synchronization tool that can be used to replicate configuration files across all nodes in the cluster, and even across GEO clusters. 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

Allow interaction with the in-kernel connection tracking system for enabling *stateful* packet inspection for iptables. Used by the High Availability Extension to synchronize the connection status between cluster nodes. 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 12 Deployment Guide*, available at <http://www.suse.com/doc> ↗. 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:*

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

```
root # zypper in -t pattern ha_sles
```

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.

- a. *Defining the Communication Channels*
- b. Optional: *Defining Authentication Settings*
- c. *Transferring the Configuration to All Nodes*. Whereas the configuration of Csync2 is done on one node only, the services Csync2 and xinetd need to be started on all nodes.
- d. Optional: *Synchronizing Connection Status Between Cluster Nodes*
- e. *Configuring Services*
- f. *Bringing the Cluster Online*. The 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 (ha-cluster-bootstrap)”*.
- For a manual setup (or for adjusting any options after the automatic setup), refer to *Section 3.5, “Manual Cluster Setup (YaST)”*.

You can also use a combination of both setup methods, for example: set up one node with YaST cluster and then use **ha-cluster-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”*.

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 12 Deployment Guide*, available at <http://www.suse.com/doc>⁷. 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 module 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 12 and SUSE Linux Enterprise High Availability Extension 12 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”](#).

3.4 Automatic Cluster Setup (ha-cluster-bootstrap)

The ha-cluster-bootstrap package provides everything you need to get a one-node cluster up and running, to make other nodes join, and to remove nodes from an existing cluster:

Automatically Setting Up the First Node

With ha-cluster-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

With ha-cluster-join, add more nodes to your cluster.

Removing Nodes From An Existing Cluster

With ha-cluster-remove, remove nodes from your cluster.

All commands execute bootstrap scripts that require only a minimum of time and manual intervention. The bootstrap scripts for initialization and joining automatically open the ports in the firewall that are needed for cluster communication. The configuration is written to `/etc/sysconfig/SuSEfirewall2.d/services/cluster`. 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”* and *Section 2.4, “Other Requirements and Recommendations”* are fulfilled.
- The `ha-cluster-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 10, Network Device Bonding*.
- 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. In addition, you will need a suitable hardware watchdog device. For more information, refer to *Chapter 17, Storage Protection*.
- 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 `ha-cluster-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.

Additionally, you can run the script with the `-t` option to let it perform additional cluster configuration based on templates. For example, `ha-cluster-init -t ocfs2` will partition some shared storage into two pieces: 1 MB for SBD, the remainder for OCFS2. For details about the script's range of functions, its options, and an overview of the files it can create and modify, refer to the `ha-cluster-init` man page.

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

```
root # ha-cluster-init
```

If NTP has not been configured to start at boot time, a message appears. The script also checks for a hardware watchdog device (which is important in case you want to configure SBD) and warns you if none is present.

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):
 - a. 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`.
 - b. Enter a multicast address. The script proposes a random address that you can use as default.
 - c. 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 Pacemaker 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/ha-cluster-bootstrap.log`.

You now have a running one-node cluster. Check the cluster status with `crm status`:

```
root # crm status
Last updated: Thu Jul  3 11:04:10 2014
Last change: Thu Jul  3 10:58:43 2014
Current DC: alice (175704363) - partition with quorum
1 Nodes configured
0 Resources configured

Online: [ alice ]
```

! Important: Secure Password

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

```
root # 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 `ha-cluster-join` bootstrap script. The script only needs access to an existing cluster node and will complete the basic setup on the current machine automatically. Follow the steps below. For details, refer to the `ha-cluster-join` man page.

If you have configured the existing cluster nodes with the YaST cluster module, make sure the following prerequisites are fulfilled before you run `ha-cluster-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"*.

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:

```
root # ha-cluster-join
```

If NTP has not been configured to start at boot time, a message appears. The script also checks for a hardware watchdog device (which is important in case you want to configure SBD) and warns you if none is present.

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/ha-cluster-bootstrap.log`.

Check the cluster status with `crm status`. If you have successfully added a second node, the output will be similar to the following:

```
root # crm status
Last updated: Thu Jul  3 11:07:10 2014
Last change: Thu Jul  3 10:58:43 2014
Current DC: alice (175704363) - partition with quorum
2 Nodes configured
0 Resources configured

Online: [ alice bob ]
```

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 4.1.2, "Option no-quorum-policy"](#).

PROCEDURE 3.4: REMOVING NODES FROM AN EXISTING CLUSTER

If you have a cluster up and running (with at least two nodes), you can remove single nodes from the cluster with the `ha-cluster-remove` bootstrap script. You need to know the IP address or hostname of the node you want to remove from the cluster. Follow the steps below. For details, refer to the `ha-cluster-remove` man page.

1. Log in as `root` to one of the cluster nodes.
2. Start the bootstrap script by executing:

```
root # ha-cluster-remove -c IP_ADDR_OR_HOSTNAME
```

The script enables the `sshd`, stops the Pacemaker service on the specified node, and propagates the files to synchronize with Csync2 across the remaining nodes.

If you specified a hostname and the node to remove cannot be contacted (or the hostname cannot be resolved), the script will inform you and ask if the node should be removed anyway. If you specified an IP address and the node cannot be contacted, you will be asked to enter the hostname and to confirm whether to remove the node anyway.

3. To remove more nodes, repeat the step above.
4. For details of the process, check `/var/log/ha-cluster-bootstrap.log`.

If you need to re-add the removed node at a later point in time, add it with `ha-cluster-join`. For details, refer to *Procedure 3.3, "Adding Nodes to an Existing Cluster"*.

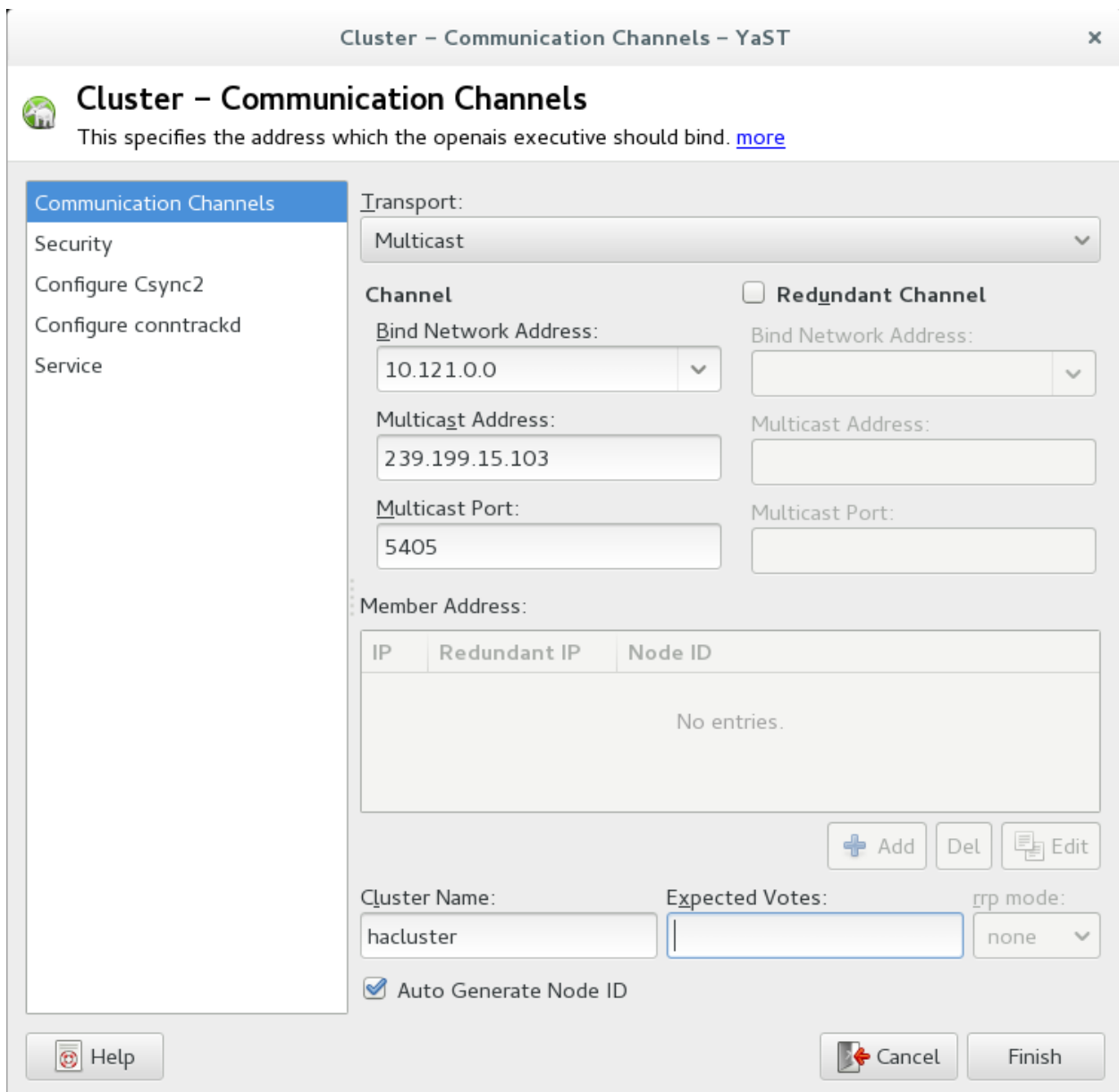
3.5 Manual Cluster Setup (YaST)

See *Section 3.2, "Overview"* 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.



Cluster - Communication Channels
 This specifies the address which the openais executive should bind. [more](#)

Communication Channels

- Security
- Configure Csync2
- Configure contrackd
- Service

Transport:
 Multicast

Channel

Bind Network Address:
 10.121.0.0

Multicast Address:
 239.199.15.103

Multicast Port:
 5405

☐ **Redundant Channel**

Bind Network Address:

Multicast Address:

Multicast Port:

Member Address:

IP	Redundant IP	Node ID
No entries.		

[+](#) Add [Del](#) [Edit](#)

Cluster Name:
 hacluster

Expected Votes:

grp mode:
 none

☒ Auto Generate Node ID

[Help](#) [Cancel](#) [Finish](#)

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

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

- *Network Device Bonding.*
- A second communication channel in Corosync. For details, see *Procedure 3.6, "Defining a Redundant Communication Channel"*.

If possible, choose network device bonding.

PROCEDURE 3.5: DEFINING THE FIRST COMMUNICATION CHANNEL

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

1. In the YaST cluster module, switch to the *Communication Channels* category.
2. To use multicast:
 - a. Set the *Transport* protocol to Multicast.
 - b. Define the *Bind Network Address*. Set the value to the subnet you will use for cluster multicast.
 - c. Define the *Multicast Address*.
 - d. 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.

Cluster – Communication Channels – YaST

Cluster – Communication Channels

This specifies the address which the openais executive should bind. [more](#)

Communication Channels

Security

Configure Csync2

Configure contrackd

Service

Transport:

Multicast

☐ Redundant Channel

Channel

Bind Network Address:

10.121.0.0

Multicast Address:

239.199.15.103

Multicast Port:

5405

Member Address:

IP	Redundant IP	Node ID
No entries.		

+ Add Del Edit

Cluster Name:

NUE1

Expected Votes:

3

rrp mode:

none

☒ Auto Generate Node ID

Help Cancel Finish

FIGURE 3.2: YAST CLUSTER—MULTICAST CONFIGURATION

3. To use unicast:

- Set the *Transport* protocol to Unicast.
- Define the *Bind Network Address*. Set the value to the subnet you will use for cluster unicast.
- Define the *Multicast Port*.
- 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 the following details:

- *IP Address*
- *Redundant IP Address* (only required if you use a second communication channel in Corosync)
- *Node ID* (only required if the option *Auto Generate Node ID* is disabled)

To modify or remove any addresses of cluster members, use the *Edit* or *Del* buttons.

Cluster - Communication Channels - YaST

Cluster - Communication Channels
This specifies the address which the openais executive should bind. [more](#)

Communication Channels

Security
Configure Csync2
Configure contrackd
Service

Transport: Unicast

Channel

Bind Network Address: 10.121.0.0

Multicast Address: 239.199.15.103

Multicast Port: 5405

☐ **Redundant Channel**

Bind Network Address:

Multicast Address:

Multicast Port:

Member Address:

IP	Redundant IP	Node ID
192.168.2.100		
192.168.2.101		
192.168.2.103		

+ Add Del Edit

Cluster Name: NUE1 Expected Votes: 3 rrp mode: none

☒ Auto Generate Node ID

Help Cancel Finish

FIGURE 3.3: YAST CLUSTER—UNICAST CONFIGURATION

4. The option *Auto Generate Node ID* is enabled by default. If you are using IPv4 addresses, node IDs are optional but they are required when using IPv6 addresses. To automatically generate a unique ID for every cluster node (which is less error-prone than specifying IDs manually for each node), keep this option enabled.
5. Define a *Cluster Name*.
6. Enter the number of *Expected Votes*. This is important for Corosync to calculate *quorum* in case of a partitioned cluster. By default, each node each node has 1 vote. The number of *Expected Votes* must match the number of nodes in your cluster.
7. 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.
8. If needed, define a second communication channel as described below. Or click *Next* and proceed with *Procedure 3.7, "Enabling Secure Authentication"*.

PROCEDURE 3.6: 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)* 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"*.

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.example.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 3.7: 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`.

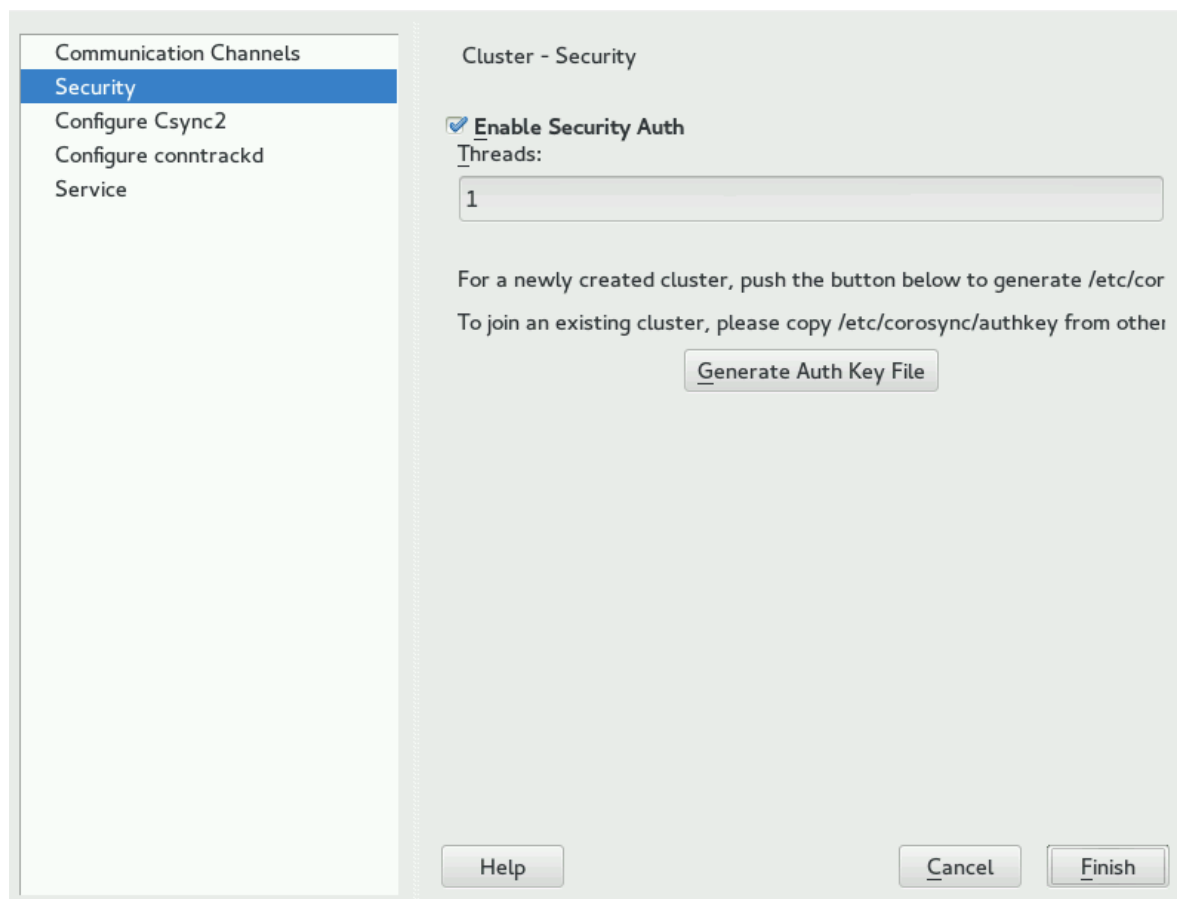


FIGURE 3.4: YAST CLUSTER—SECURITY

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, click *Next* and proceed with [Section 3.5.4, “Transferring the Configuration to All Nodes”](#).

3.5.4 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.*
2. *Synchronizing the Configuration Files with Csync2.*

Csync2 helps you to keep track of configuration changes and to keep files synchronized across the cluster nodes:

- You can define a list of files that are important for operation.
- You can show changes of these files (against the other cluster nodes).
- You can synchronize the configured files with a single command.
- With a simple shell script in `~/ .bash_logout`, you can be reminded about unsynchronized changes before logging out from the system.

Find detailed information about Csync2 at <http://oss.linbit.com/csync2/> and <http://oss.linbit.com/csync2/paper.pdf>.

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.



Tip: Hostname Resolution

In case hostname resolution should not work properly in your network for some reason, you can also specify a combination of hostname and IP address for each cluster node. To do so, use the string `HOSTNAME@IP` such as `alice@192.168.2.100`, for example. Csync2 will then use the IP addresses when connecting.

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*.

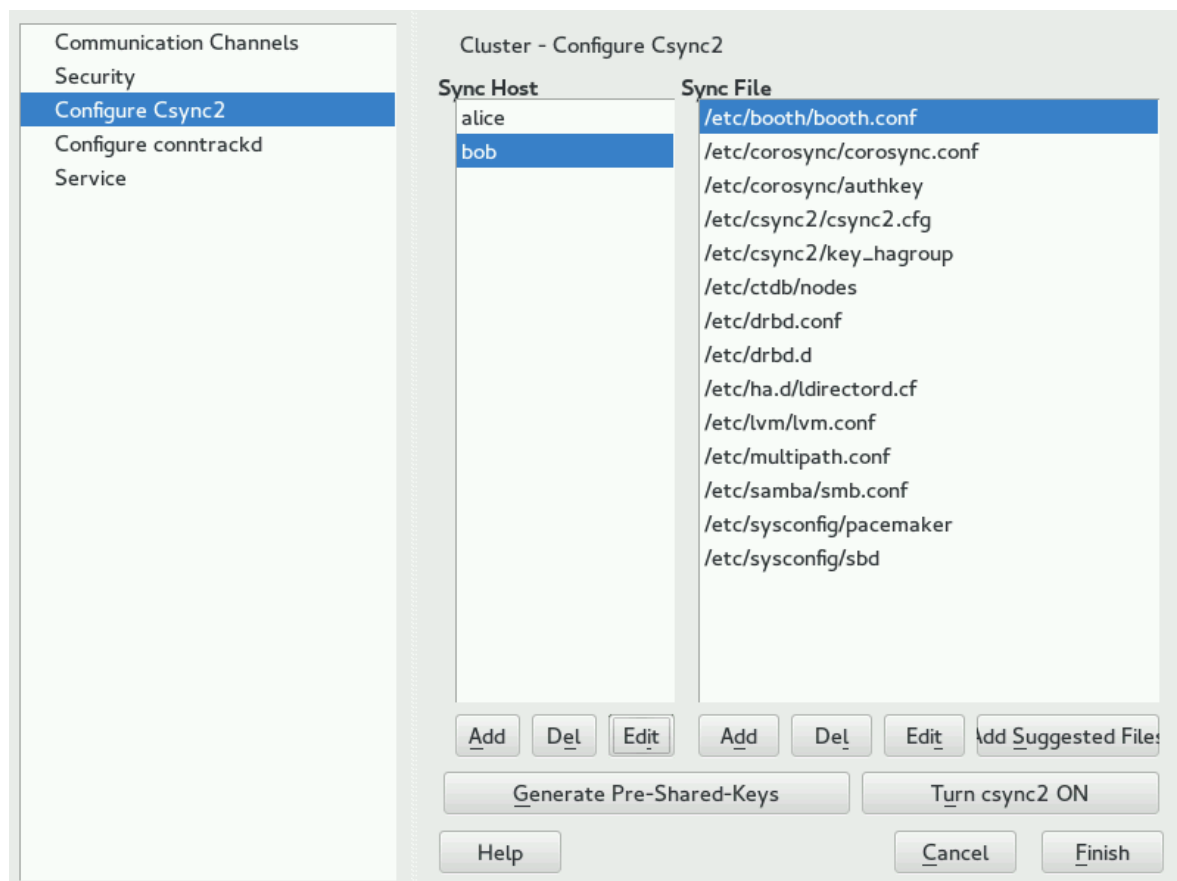


FIGURE 3.5: YAST CLUSTER—CSYNC2

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 the following command to start Csync2 automatically at boot time:

```
root # systemctl enable csync2.socket
```

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”](#).
8. For further cluster configuration, click *Next* and proceed with [Section 3.5.5, “Synchronizing Connection Status Between Cluster Nodes”](#).

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"*. 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* 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:

```
root # systemctl enable csync2.socket
root # systemctl enable xinetd.service
root # systemctl start xinetd.service
```

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

```
root # 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:

```
root # 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.

3.5.5 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.*
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 Linux Virtual Server, see [Load Balancing](#).

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. If you modified any options for an existing cluster, confirm your changes and close the cluster module.
7. For further cluster configuration, click *Next* and proceed with [Section 3.5.6, “Configuring Services”](#).

Cluster - Configure conntrackd

Communication Channels
Security
Configure Csync2
Configure conntrackd
Service

Conntrackd is a daemon which helps to duplicate firewall status between cluster nodes.
YaST can help to configure some basic aspects of conntrackd.
You need to start it with the `ocf:heartbeat:conntrackd`.

Dedicated Interface:
eth0 IP: 10.121.9.43

Multicast Address:
239.180.93.156

Group Number:
1

Generate /etc/conntrackd/conntrackd.conf

Help Cancel Finish

FIGURE 3.6: YAST CLUSTER—conntrackd

3.5.6 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, Pacemaker must be running as a service.

PROCEDURE 3.11: ENABLING PACEMAKER

1. In the YaST cluster module, switch to the *Service* category.
2. To start Pacemaker 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 Pacemaker manually each time this node is booted. To start Pacemaker manually, use the command:

```
root # systemctl start pacemaker.service
```

3. To start or stop Pacemaker immediately, click the respective button.
4. 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.

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”](#).

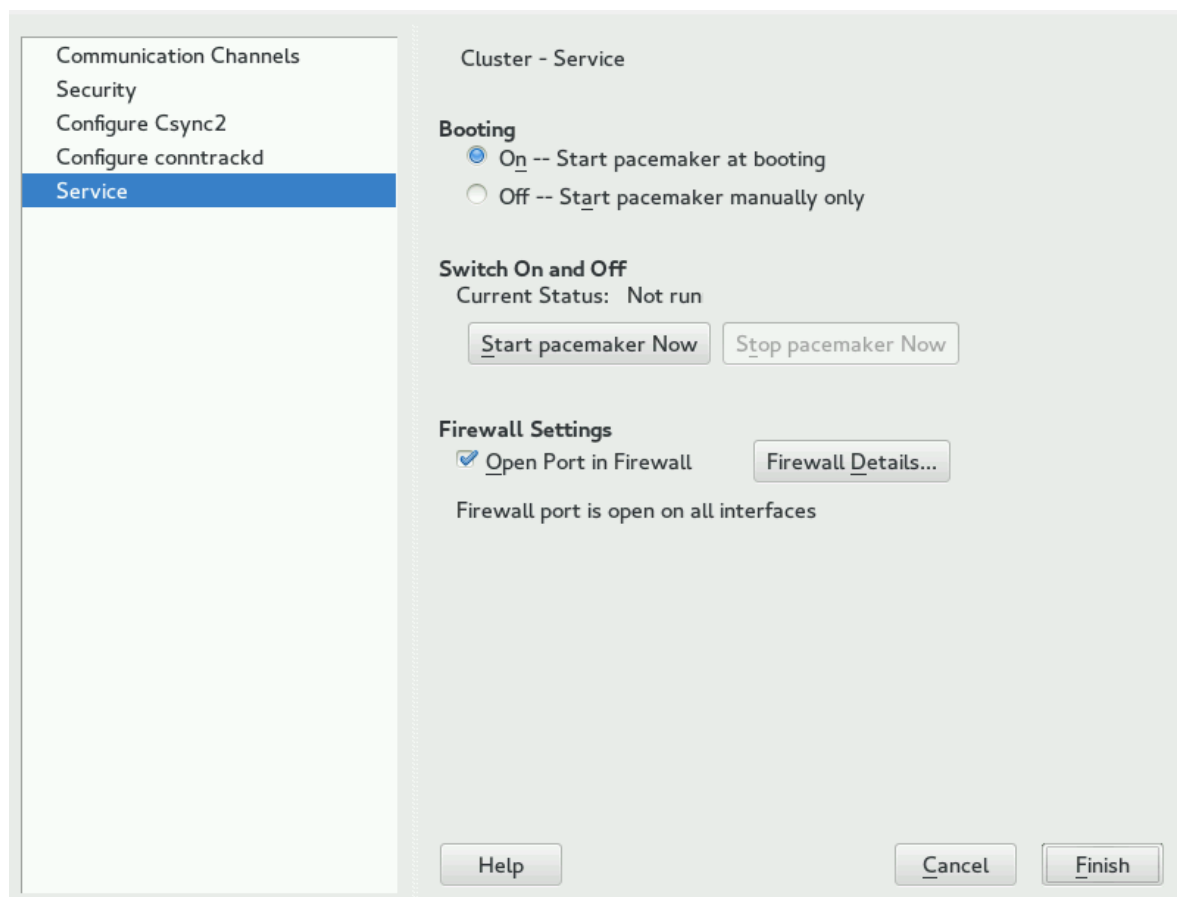


FIGURE 3.7: YAST CLUSTER—SERVICES

3.5.7 Bringing the Cluster Online

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

PROCEDURE 3.12: STARTING PACEMAKER AND CHECKING THE STATUS

1. Log in to an existing node.
2. Check if the service is already running:

```
root # systemctl status pacemaker.service
```

If not, start Pacemaker now:

```
root # systemctl start pacemaker.service
```

3. Repeat the steps above for each of the cluster nodes.
4. On one of the nodes, check the cluster status with the `crm status` command. If all nodes are online, the output should be similar to the following:

```
root # crm status
Last updated: Thu Jul  3 11:07:10 2014
Last change: Thu Jul  3 10:58:43 2014
Current DC: alice (175704363) - partition with quorum
2 Nodes configured
0 Resources configured

Online: [ alice bob ]
```

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. Use one of the cluster management tools like the `crm shell` (`crmsh`) or the HA Web Konsole. For more information, see [Chapter 6, Configuring and Managing Cluster Resources \(Command Line\)](#) or [Chapter 5, Configuring and Managing Cluster Resources \(Web Interface\)](#).

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 3.13: CLONING A CLUSTER NODE WITH AUTOYAST



Important: Identical Hardware

This scenario assumes you are rolling out SUSE Linux Enterprise High Availability Extension 12 to a set of machines with exactly the same hardware configuration.

If you need to deploy cluster nodes on non-identical hardware, refer to chapter *Automated Installation*, section *Rule-Based Autoinstallation* in the *SUSE Linux Enterprise 12 Deployment Guide*, available at <http://www.suse.com/doc> .

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"*, and *Section 3.4, "Automatic Cluster Setup (ha-cluster-bootstrap)"* or *Section 3.5, "Manual Cluster Setup (YaST)"*, respectively.
2. Follow the description outlined in the *SUSE Linux Enterprise 12 Deployment Guide* for simple mass installation. This includes the following basic steps:
 - a. 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.
If you have configured DRBD, you can select and clone this module in the AutoYaST GUI, too.
 - b. Determining the source of the AutoYaST profile and the parameter to pass to the installation routines for the other nodes.
 - c. Determining the source of the SUSE Linux Enterprise Server and SUSE Linux Enterprise High Availability Extension installation data.
 - d. Determining and setting up the boot scenario for autoinstallation.
 - e. Passing the command line to the installation routines, either by adding the parameters manually or by creating an `info` file.
 - f. 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 3.14: 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 3.5.4, "Transferring the Configuration to All Nodes"*.
2. To bring the node online, start the Pacemaker service on the cloned node as described in *Section 3.5.7, "Bringing the Cluster Online"*.

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

II Configuration and Administration

- 4 Configuration and Administration Basics **50**
- 5 Configuring and Managing Cluster Resources (Web Interface) **87**
- 6 Configuring and Managing Cluster Resources (Command Line) **136**
- 7 Adding or Modifying Resource Agents **169**
- 8 Fencing and STONITH **173**
- 9 Access Control Lists **184**
- 10 Network Device Bonding **193**
- 11 Load Balancing **199**
- 12 GEO Clusters (Multi-Site Clusters) **213**

4 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 Hawk and the `crm` shell.

4.1.1 Overview

For an overview of all global cluster options and their default values, see *Pacemaker Explained* (*Pacemaker 1.1 for Corosync 2.x and crmsh*), available from <http://www.clusterlabs.org/doc/>⁷. Refer to section *Available Cluster Options*.

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*
- *Option stonith-enabled*

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

- Hawk: *Procedure 5.3, “Modifying Global Cluster Options”*
- crmsh: *Section 6.3, “Configuring Global Cluster Options”*

4.1.2 Option no-quorum-policy

This global option defines what to do when a cluster partition 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 partition 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

If quorum is lost, all nodes in the affected cluster partition are fenced.

4.1.3 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`, but be aware that this has impact on the support status for your product. Furthermore, with `stonith-enabled="false"`, resources like the Distributed Lock Manager (DLM) and all services depending on DLM (such as cLVM2, GFS2, and OCFS2) will fail to start.



Important: No Support Without STONITH

A cluster without STONITH is not supported.

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.

If you need to execute any testing or maintenance tasks after the services are already running under cluster control, make sure to put the respective resources, nodes (or the whole cluster) into maintenance mode before you touch any of them manually. For details, see [Section 4.7, “Maintenance Mode”](#).

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 5, Configuring and Managing Cluster Resources \(Web Interface\)](#)
- crmsh: [Chapter 6, Configuring and Managing Cluster Resources \(Command Line\)](#)

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:

Open Cluster Framework (OCF) Resource Agents

OCF RA agents are best suited for use with High Availability, especially when you need multi-state 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 specifi-

cations have strict definitions of which exit codes must be returned by actions, see [Section 7.3, “OCF Return Codes and Failure Recovery”](#). The cluster follows these specifications exactly.

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).

Alternatively, use the `crmsh` to view information on OCF resource agents. For details, see [Section 6.1.3, “Displaying Information about OCF Resource Agents”](#).

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/inisrptact.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`.

Systemd

Starting with SUSE Linux Enterprise 12, `systemd` is a replacement for the popular System V init daemon. Pacemaker can manage `systemd` services if they are present. Instead of init scripts, `systemd` has unit files. Generally the services (or unit files) are provided by the operating system. In case you want to convert existing init scripts, find more information at: <http://0pointer.de/blog/projects/systemd-for-admins-3.html>.

Service

There are currently many “common” types of system services that exist in parallel: `LSB` (belonging to System V init), `systemd`, and (in some distributions) `upstart`. Therefore, Pacemaker supports a special alias which intelligently figures out which one applies to a

given cluster node. This is particularly useful when the cluster contains a mix of systemd, upstart, and LSB services. Pacemaker will try to find the named service in the following order: as LSB (SYS-V) init script, a Systemd unit file, or an Upstart job.

Nagios

Monitoring plug-ins (formerly called Nagios plug-ins) allow to monitor services on remote hosts. Pacemaker can do remote monitoring with the monitoring plug-ins if they are present. For detailed information, see [Section 4.5.1, “Monitoring Services on Remote Hosts with Monitoring Plug-ins”](#).

STONITH (Fencing) Resource Agents

This class is used exclusively for fencing related resources. For more information, see [Chapter 8, Fencing and STONITH](#).

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 5.5, “Adding Primitive Resources”](#)
- crmsh: [Section 6.4.1, “Creating Cluster Resources”](#)

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”](#).

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”](#).

Multi-state Resources (formerly known as Master/Slave Resources)

Multi-state resources are a special type of clone resources, they can have multiple modes. For more information, refer to [Section 4.2.5.3, “Multi-state Resources”](#).

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”](#).

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 5.3.4, “Using Resource Templates”](#)
- crmsh: [Section 6.4.2, “Creating Resource Templates”](#)

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 multi-state resources.

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.

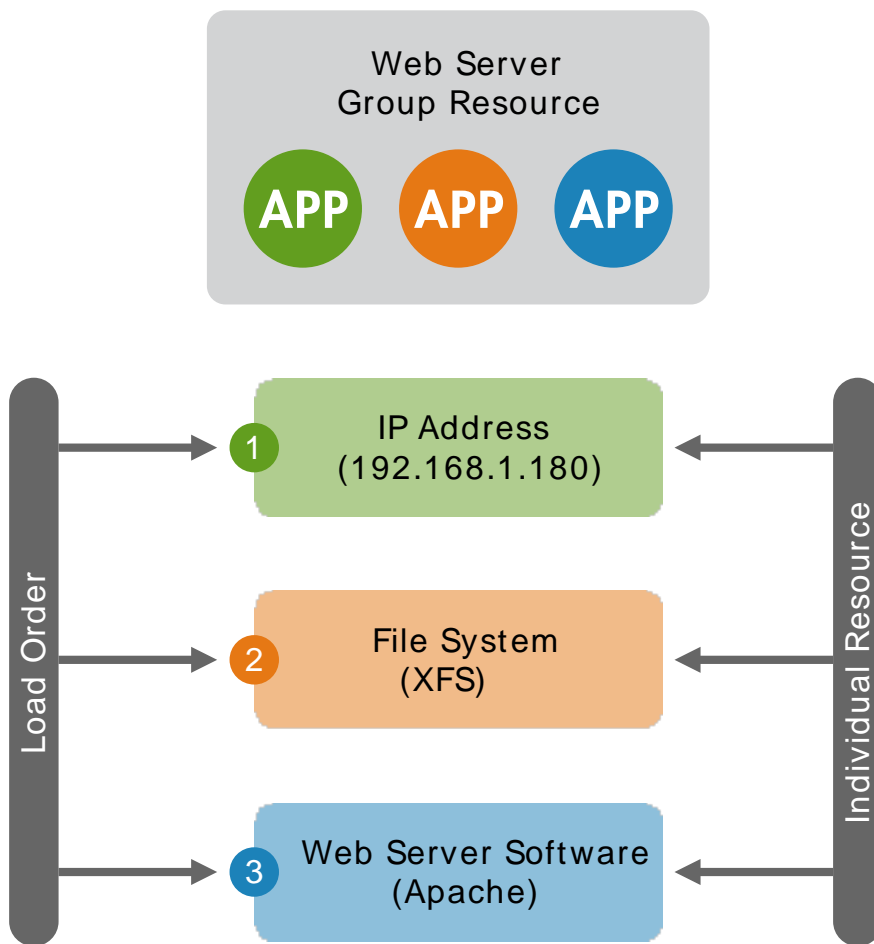


FIGURE 4.1: **GROUP RESOURCE**

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 5.16, "Adding a Resource Group"*
- crmsh: *Section 6.4.9, "Configuring a Cluster Resource Group"*

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 (Multi-state Resources)

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, "Multi-state Resources"*.

Clones must contain exactly one group or one regular resource.

When configuring resource monitoring or constraints, clones have different requirements than simple resources. For details, see *Pacemaker Explained (Pacemaker 1.1 for Corosync 2.x and crmsh)*, available from <http://www.clusterlabs.org/doc/>. Refer to section *Clones - Resources That Get Active on Multiple Hosts*.

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

- Hawk: *Procedure 5.17, "Adding or Modifying Clones"*
- crmsh: *Section 6.4.10, "Configuring a Clone Resource"*.

4.2.5.3 Multi-state Resources

Multi-state resources are a specialization of clones that allow the instances to be in one of two operating modes (called master or slave, but can mean whatever you wish them to mean). Multi-state resources must contain exactly one group or one regular resource.

When configuring resource monitoring or constraints, multi-state resources have different requirements than simple resources. For details, see *Pacemaker Explained (Pacemaker 1.1 for Corosync 2.x and crmsh)*, available from <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 Hawk as described in *Procedure 5.5, "Adding Primitive Resources"*.

TABLE 4.1: OPTIONS FOR A PRIMITIVE RESOURCE

Option	Description	Default
<u>priority</u>	If not all resources can be active, the cluster will stop lower priority resources in order to keep higher priority ones active.	<u>0</u>

Option	Description	Default
<u>target-role</u>	In what state should the cluster attempt to keep this resource? Allowed values: <u>stopped</u> , <u>started</u> , <u>master</u> .	<u>started</u>
<u>is-managed</u>	Is the cluster allowed to start and stop the resource? Allowed values: <u>true</u> , <u>false</u> . If the value is set to <u>false</u> , the status of the resource is still monitored and any failures are reported (which is different from setting a resource to <u>maintenance="true"</u>).	<u>true</u>
<u>maintenance</u>	Can the resources be touched manually? Allowed values: <u>true</u> , <u>false</u> . If set to <u>true</u> , to all resources become unmanaged: the cluster will stop monitoring them and thus be oblivious about their status. You can stop or restart cluster resources at will, without the cluster attempting to restart them.	<u>false</u>
<u>resource-stickiness</u>	How much does the resource prefer to stay where it is?	calculated

Option	Description	Default
<u>migration-threshold</u>	How many failures should occur for this resource on a node before making the node ineligible to host this resource?	<u>INFINITY</u> (disabled)
<u>multiple-active</u>	What should the cluster do if it ever finds the resource active on more than one node? Allowed values: <u>block</u> (mark the resource as unmanaged), <u>stop_only</u> , <u>stop_start</u> .	<u>stop_start</u>
<u>failure-timeout</u>	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)?	<u>0</u> (disabled)
<u>allow-migrate</u>	Allow resource migration for resources which support <u>migrate_to</u> / <u>migrate_from</u> actions.	<u>false</u>
<u>remote-node</u>	The name of the remote-node this resource defines. This both enables the resource as a remote node and defines the unique name used to identify the remote node. If no other parameters are set, this value will al-	none (disabled)

Option	Description	Default
	<p>so be assumed as the host name to connect to at <u>remote-port</u> port.</p> <div>  Warning: Use Unique IDs This value must not overlap with any existing resource or node IDs. </div>	
<u>remote-port</u>	Custom port for the guest connection to pacemaker_remote.	<u>3121</u>
<u>remote-addr</u>	The IP address or host-name to connect to if the remote-node's name is not the hostname of the guest.	<u>remote-node</u> (value used as hostname)
<u>remote-connect-timeout</u>	How long before a pending guest connection will time out.	<u>60s</u>

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 Hawk as described in *Procedure 5.5*,

"Adding Primitive Resources". 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`:

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

or (without the optional parts):

```
root # crm ra info resource_agent
```

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

For example, the command

```
root # 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 Multi-state Resources

Note that groups, clones and multi-state resources do not have instance attributes. However, any instance attributes set will be inherited by the group's, clone's or multi-state resource'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"](#).

TABLE 4.2: RESOURCE OPERATION PROPERTIES

Operation	Description
<u>id</u>	Your name for the action. Must be unique. (The ID is not shown).
<u>name</u>	The action to perform. Common values: <u>monitor</u> , <u>start</u> , <u>stop</u> .
<u>interval</u>	How frequently to perform the operation. Unit: seconds
<u>timeout</u>	How long to wait before declaring the action has failed.

Operation	Description
<u>requires</u>	What conditions need to be satisfied before this action occurs. Allowed values: <u>nothing</u> , <u>quorum</u> , <u>fencing</u> . The default depends on whether fencing is enabled and if the resource's class is <u>stonith</u> . For STONITH resources, the default is <u>nothing</u> .
<u>on-fail</u>	<p>The action to take if this action ever fails. Allowed values:</p> <ul style="list-style-type: none"> • <u>ignore</u>: Pretend the resource did not fail. • <u>block</u>: Do not perform any further operations on the resource. • <u>stop</u>: Stop the resource and do not start it elsewhere. • <u>restart</u>: Stop the resource and start it again (possibly on a different node). • <u>fence</u>: Bring down the node on which the resource failed (STONITH). • <u>standby</u>: Move <i>all</i> resources away from the node on which the resource failed.
<u>enabled</u>	If <u>false</u> , the operation is treated as if it does not exist. Allowed values: <u>true</u> , <u>false</u> .
<u>role</u>	Run the operation only if the resource has this role.

Operation	Description
<u>record-pending</u>	Can be set either globally or for individual resources. Makes the CIB reflect the state of “in-flight” operations on resources.
<u>description</u>	Description of the operation.

4.2.9 Timeout Values

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

- op_defaults (global timeout for operations),
- a specific timeout value defined in a resource template,
- a specific timeout value defined for a resource.



Note: Priority of Values

If a *specific* value is defined for a resource, it takes precedence over the global default. A specific value for a resource also takes precedence over a value that is defined in a resource template.

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 CRM executes an initial monitoring for each resource on every node, the so-called probe, which is also executed after the cleanup of a resource. If no specific timeout is configured for the resource's monitoring operation, the CRM will automatically check for any other monitoring operations. If multiple monitoring operations are defined for a resource, the CRM will select the

one with the smallest interval and will use its timeout value as default timeout for probing. If no monitor operation is configured at all, the cluster-wide default applies. The default is 20 seconds (if not specified otherwise by configuring the op_defaults parameter). If you do not want to rely on the automatic calculation or the op_defaults value, define a specific timeout for this monitoring by adding a monitoring operation to the respective resource, with the interval set to 0, for example:

```
crm(live)configure# primitive rscl ocf:pacemaker:Dummy \  
    op monitor interval="0" timeout="60"
```

The probe of rscl will time out in 60s, independent of the global timeout defined in op_defaults, or any other operation timeouts configured.

You can adjust the global default for operations and set any specific timeout values with both crmsh and Hawk. The best practice for determining and setting timeout values is as follows:

PROCEDURE 4.1: DETERMINING TIMEOUT VALUES

1. Check how long it takes your resources to start and stop (under load).
2. If needed, add the op_defaults parameter and set the (default) timeout value accordingly:

- a. For example, set op_defaults to 60 seconds:

```
crm(live)configure# op_defaults timeout=60
```

- b. 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, it 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.
- The failure is reflected in the cluster management tools (Hawk, crm status), 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 5.15, "Adding or Modifying Monitor Operations"*
- crmsh: *Section 6.4.8, "Configuring Resource Monitoring"*

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.

4.4.1.1 Resource Sets

4.4.1.1.1 Using Resource Sets for Defining Constraints

As an alternative format for defining location, colocation or ordering constraints, you can use resource sets, where primitives are grouped together in one set. Previously this was possible either by defining a resource group (which could not always accurately express the design), or by defining each relationship as an individual constraint, causing a constraint explosion as the number of resources and combinations grew. The configuration via resource sets is not necessarily less verbose, but generally easier to understand and maintain, as the following examples show.

EXAMPLE 4.2: A RESOURCE SET FOR LOCATION CONSTRAINTS

For example, you can use the following configuration of a resource set (loc-alice) in the crmsh to place two virtual IPs (vip1 and vip2) on the same node, namely alice:

```
crm(live)configure# primitive vip1 ocf:heartbeat:IPaddr2 params ip=192.168.1.5
crm(live)configure# primitive vip2 ocf:heartbeat:IPaddr2 params ip=192.168.1.6
crm(live)configure# location loc-alice { vip1 vip2 } inf: alice
```

If you want to use resource sets to replace a configuration of colocation constraints, consider the following two examples:

EXAMPLE 4.3: A CHAIN OF COLOCATED RESOURCES

```
<constraints>
  <rsc_colocation id="coloc-1" rsc="B" with-rsc="A" score="INFINITY"/>
  <rsc_colocation id="coloc-2" rsc="C" with-rsc="B" score="INFINITY"/>
  <rsc_colocation id="coloc-3" rsc="D" with-rsc="C" score="INFINITY"/>
</constraints>
```

The same configuration expressed by a resource set:

```
<constraints>
  <rsc_colocation id="coloc-1" score="INFINITY" >
    <resource_set id="colocated-set-example" sequential="true">
      <resource_ref id="A"/>
      <resource_ref id="B"/>
      <resource_ref id="C"/>
      <resource_ref id="D"/>
    </resource_set>
  </rsc_colocation>
</constraints>
```

If you want to use resource sets to replace a configuration of ordering constraints, consider the following two examples:

EXAMPLE 4.4: A CHAIN OF ORDERED RESOURCES

```
<constraints>
  <rsc_order id="order-1" first="A" then="B" />
  <rsc_order id="order-2" first="B" then="C" />
  <rsc_order id="order-3" first="C" then="D" />
</constraints>
```

The same purpose can be achieved by using a resource set with ordered resources:

EXAMPLE 4.5: A CHAIN OF ORDERED RESOURCES EXPRESSED AS RESOURCE SET

```
<constraints>
```

```
<rsc_order id="order-1">
  <resource_set id="ordered-set-example" sequential="true">
    <resource_ref id="A"/>
    <resource_ref id="B"/>
    <resource_ref id="C"/>
    <resource_ref id="D"/>
  </resource_set>
</rsc_order>
</constraints>
```

Sets can be either ordered (`sequential=true`) or unordered (`sequential=false`). Furthermore, the `require-all` attribute can be used to switch between `AND` and `OR` logic.

4.4.1.1.2 Resource Sets for Colocation Constraints Without Dependencies

Sometimes it is useful to be able to place a group of resources on the same node (defining a colocation constraint), but without having hard dependencies between the resources. For example, you want two resources to be placed on the same node, but you do *not* want the cluster to trigger any action (that means restart the other one) if one of them fails. This can be achieved on the `crm` shell by using the `weak bond` command.


Learn how to set these “weak bonds” with your preferred cluster management tool:

- `crmsh`: *Section 6.4.4.3, “Collocating Sets for Resources Without Dependency”*

4.4.1.2 For More Information

Learn how to add the various kinds of constraints with your preferred cluster management tool:

- `Hawk`: *Section 5.3.5, “Configuring Resource Constraints”*
- `crmsh`: *Section 6.4.4, “Configuring Resource Constraints”*

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/> :

- *Pacemaker Explained (Pacemaker 1.1 for Corosync 2.x and crmsh)*, chapter *Resource Constraints*
- *Colocation Explained*
- *Ordering Explained*

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 (see [Section 4.2.4, “Resource Templates”](#)), it can be referenced in the following types of constraints:

- order constraints,
- colocation constraints,
- rsc_ticket constraints (for GEO 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 5.11, “Using Resource Sets for Constraints”](#).

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

However, you can specify the node a resource will fail over to by configuring one or several location constraints and a migration-threshold for that resource.

Learn how to specify failover nodes with your preferred cluster management tool:

- Hawk: [Section 5.3.6, “Specifying Resource Failover Nodes”](#)
- crmsh: [Section 6.4.5, “Specifying Resource Failover Nodes”](#)

EXAMPLE 4.6: MIGRATION THRESHOLD—PROCESS FLOW

For example, let us assume you have configured a location constraint for resource `rsc1` to preferably run on `alice`. 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 with your preferred cluster management tool:

- Hawk: *Section 5.3.6, "Specifying Resource Failover Nodes"*
- crmsh: *Section 6.4.5, "Specifying Resource Failover Nodes"*

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:

- **crmsh:** [Section 6.4.7, “Configuring Placement of Resources Based on Load Impact”](#)

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.

If multiple resources with utilization attributes are grouped or have colocation constraints, the High Availability Extension takes that into account. If possible, the resources will be placed on a node that can fulfill *all* capacity requirements.



Note: Utilization Attributes for Groups

It is impossible to set utilization attributes directly for a resource group. However, to simplify the configuration for a group, you can add a utilization attribute with the total capacity needed to any of the resources in the group.

The High Availability Extension 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`. The following instance attributes are available: `autoset_utilization_cpu` and `autoset_utilization_hv_memory`. Both default to `true`. 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 4.7: EXAMPLE CONFIGURATION FOR LOAD-BALANCED PLACING

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

```
node alice utilization memory="4000"  
node bob utilization memory="4000"
```

```
node charly utilization memory="4000"
primitive xenA ocf:heartbeat:Xen utilization hv_memory="3500" \
    meta priority="10"
primitive xenB ocf:heartbeat:Xen utilization hv_memory="2000" \
    meta priority="1"
primitive xenC ocf:heartbeat:Xen utilization hv_memory="2000" \
    meta priority="1"
primitive xenD ocf:heartbeat:Xen utilization hv_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.

4.4.7 Grouping Resources by Using Tags

Tags are a new feature that has been added to Pacemaker recently. Tags are a way to refer to multiple resources at once, without creating any colocation or ordering relationship between them. This can be useful for grouping conceptually related resources. For example, if you have a number of resources related to a database, create a tag called tag-db and add all resources related to the database to this tag. This allows you to stop or start them all with a single command. Tags can also be used in constraints. For example, the following location constraint loc-db-prefer applies to the set of resources tagged with tag-db:

```
location loc-db-prefer tag-db 100: alice
```

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

- `crmsb`: *Section 6.5.5, “Grouping/Tagging Resources”*

4.5 Managing Services on Remote Hosts

The possibilities for monitoring and managing services on remote hosts has become increasingly important during the last years. While SUSE Linux Enterprise High Availability Extension 11 SP3 offered fine-grained monitoring of services on remote hosts via monitoring plug-ins, the recent addition of the `pacemaker_remote` service now allows SUSE Linux Enterprise High Availability Extension 12 to fully manage and monitor resources on remote hosts just as if they were a real cluster node—without the need to install the cluster stack on the remote machines.

4.5.1 Monitoring Services on Remote Hosts with Monitoring Plug-ins

Monitoring of virtual machines can be done with the VM agent (which only checks if the guest shows up in the hypervisor), or by external scripts called from the VirtualDomain or Xen agent. Up to now, more fine-grained monitoring was only possible with a full setup of the High Availability stack within the virtual machines.

By providing support for monitoring plug-ins (formerly named Nagios plug-ins), the High Availability Extension now also allows you to monitor services on remote hosts. You can collect external statuses on the guests without modifying the guest image. For example, VM guests might run Web services or simple network resources that need to be accessible. With the Nagios resource agents, you can now monitor the Web service or the network resource on the guest. In case these services are not reachable anymore, the High Availability Extension will trigger a restart or migration of the respective guest.

If your guests depend on a service (for example, an NFS server to be used by the guest), the service can either be an ordinary resource managed by the cluster or an external service that is not managed by the cluster but monitored with Nagios resources instead.

To configure the Nagios resources, the following packages must be installed on the host:

- `monitoring-plugins`
- `monitoring-plugins-metadata`

YaST or zypper will resolve any dependencies on further packages, if required.

A typical use case is to configure the monitoring plug-ins as resources belonging to a resource container, which usually is a VM. The container will be restarted if any of its resources has failed. Refer to *Example 4.8, "Configuring Resources for Monitoring Plug-ins"* for a configuration example. Alternatively, Nagios resource agents can also be configured as ordinary resources if you want to use them for monitoring hosts or services via the network.

EXAMPLE 4.8: CONFIGURING RESOURCES FOR MONITORING PLUG-INS

```
primitive vm1 ocf:heartbeat:VirtualDomain \  
    params hypervisor="qemu:///system" config="/etc/libvirt/qemu/vm1.xml" \  
    op start interval="0" timeout="90" \  
    op stop interval="0" timeout="90" \  
    op monitor interval="10" timeout="30" \  
primitive vm1-sshd nagios:check_tcp \  
    params hostname="vm1" port="22" \ ❶ \  
    op start interval="0" timeout="120" \ ❷ \  
    op monitor interval="10" \  
group g-vm1-and-services vm1 vm1-sshd \  
meta container="vm1" ❸
```

- ❶ The supported parameters are same as the long options of a monitoring plug-in. Monitoring plug-ins connect to services with the parameter `hostname`. Therefore the attribute's value must be a resolvable hostname or an IP address.
- ❷ As it takes some time to get the guest operating system up and its services running, the start timeout of the monitoring resource must be long enough.
- ❸ A cluster resource container of type `ocf:heartbeat:Xen`, `ocf:heartbeat:VirtualDomain` or `ocf:heartbeat:lxc`. It can either be a VM or a Linux Container.

The example above contains only one resource for the `check_tcp` plug-in, but multiple resources for different plug-in types can be configured (for example, `check_http` or `check_udp`).

If the hostnames of the services are the same, the `hostname` parameter can also be specified for the group, instead of adding it to the individual primitives. For example:

```
group g-vm1-and-services vm1 vm1-sshd vm1-httpd \
```

```
meta container="vm1" \  
params hostname="vm1"
```

If any of the services monitored by the monitoring plug-ins fail within the VM, the cluster will detect that and restart the container resource (the VM). Which action to take in this case can be configured by specifying the on-fail attribute for the service's monitoring operation. It defaults to restart-container.

Failure counts of services will be taken into account when considering the VM's migration-threshold.

4.5.2 Managing Services on Remote Nodes with `pacemaker_remote`

With the `pacemaker_remote` service, High Availability clusters can easily be extended to virtual nodes or remote baremetal machines that do not need to run the cluster stack to become members of the cluster.

The High Availability Extension can now launch virtual environments (KVM and LXC) as well as launch the resources that live within those virtual environments without requiring the virtual environments to run Pacemaker or Corosync.

For the use case of managing both virtual machines as cluster resources plus the resources that live within the VMs, you can now use the following setup:

- The “normal” (baremetal) cluster nodes run the High Availability Extension.
- The virtual machines run the `pacemaker_remote` service (almost no configuration required on the VM's side).
- The cluster stack on the “normal” cluster nodes launches the VMs and connects to the `pacemaker_remote` service running on the VMs to integrate them as remote nodes into the cluster.

As the remote nodes do not have the cluster stack installed, this has the following implications:

- Remote nodes do not take part in quorum.
- Remote nodes cannot become the DC.
- Remote nodes are not bound to the scalability limits (Corosync has a member limit of 16 nodes).

Find more information about the `remote_pacemaker` service, including multiple use cases with detailed setup instructions in *Pacemaker Remote—Extending High Availability into Virtual Nodes*, available at <http://www.clusterlabs.org/doc/>.

4.6 Monitoring System Health

To avoid a node running out of disk space and thus being no longer able to adequately manage any resources that have been assigned to it, the High Availability Extension provides a resource agent, `ocf:pacemaker:SysInfo`. Use it to monitor a node's health with respect to disk partitions. The SysInfo RA creates a node attribute named `#health_disk` which will be set to `red` if any of the monitored disks' free space is below a specified limit.

To define how the CRM should react in case a node's health reaches a critical state, use the global cluster option `node-health-strategy`.

PROCEDURE 4.2: CONFIGURING SYSTEM HEALTH MONITORING

To automatically move resources away from a node in case the node runs out of disk space, proceed as follows:

1. Configure an `ocf:pacemaker:SysInfo` resource:

```
primitive sysinfo ocf:pacemaker:SysInfo \  
    params disks="/tmp /var" ❶ min_disk_free="100M" ❷ disk_unit="M" ❸ \  
    op monitor interval="15s"
```

- ❶ Which disk partitions to monitor. For example, `/tmp`, `/usr`, `/var`, and `/dev`. To specify multiple partitions as attribute values, separate them with a blank.



Note: / File System Always Monitored

You do not need to specify the root partition (`/`) in `disks`. It is always monitored by default.

- ② The minimum free disk space required for those partitions. Optionally, you can specify the unit to use for measurement (in the example above, M for megabytes is used). If not specified, min_disk_free defaults to the unit defined in the disk_unit parameter.
 - ③ The unit in which to report the disk space.
2. To complete the resource configuration, create a clone of ocf:pacemaker:SysInfo and start it on each cluster node.
 3. Set the node-health-strategy to migrate-on-red:

```
property node-health-strategy="migrate-on-red"
```

In case of a #health_disk attribute set to red, the policy engine adds -INF to the resources' score for that node. This will cause any resources to move away from this node. The STONITH resource will be the last one to be stopped but even if the STONITH resource is not running any more, the node can still be fenced. Fencing has direct access to the CIB and will continue to work.

After a node's health status has turned to red, solve the issue that led to the problem. Then clear the red status to make the node eligible again for running resources. Log in to the cluster node and use one of the following methods:

- Execute the following command:

```
root # crm node status-attr NODE delete #health_disk
```

- Restart Pacemaker and Corosync on that node.
- Reboot the node.

The node will be returned to service and can run resources again.

4.7 Maintenance Mode

Every now and then, you will need to perform testing or maintenance tasks on individual cluster components or the whole cluster—be it changing the cluster configuration, updating software packages for individual nodes, or upgrading the cluster to a higher product version.

With regards to that, High Availability Extension provides maintenance options on several levels:

- for resources
- for nodes
- for the whole cluster



Warning: Risk of Data Loss

If you need to execute any testing or maintenance tasks while services are running under cluster control, make sure to follow this outline:

1. Before you start, set the individual resource, node or the whole cluster to maintenance mode. This helps to avoid unwanted side effects like resources not starting in an orderly fashion, the risk of unsynchronized CIBs across the cluster nodes or data loss.
2. Execute your maintenance task or tests.
3. After you have finished, remove the maintenance mode to start normal cluster operation.

In maintenance mode, you can stop or restart cluster resources at will—the High Availability Extension will not attempt to restart them. All resources automatically become unmanaged, that is, the High Availability Extension will cease monitoring them and thus be oblivious about their status. You can even stop all Pacemaker services on a node, and all the daemons and processes originally started as Pacemaker-managed cluster resources will continue to run. If you attempt to start Pacemaker services on a node while the cluster is in maintenance mode, Pacemaker will initiate a single one-shot monitor operation (a “probe”) for every resource to evaluate which resources currently running on that node. However, it will take no further action other than determining the resources' status.

For details on setting or unsetting maintenance mode with your preferred cluster management tool:

- Hawk: *Section 5.4.5, “Using Maintenance Mode”*
- crmsh: *Section 6.5.6, “Using Maintenance Mode”*

4.8 For More Information

<http://crmsh.github.io/> ↗

Home page of the crm shell (crmsh), the advanced command-line interface for High Availability cluster management.

<http://crmsh.github.io/documentation> ↗

Holds a number of documents about the crm shell, including a *Getting Started* tutorial for basic cluster setup with crmsh and the comprehensive *Manual* for the crm shell. The latter is available at <http://crmsh.github.io/man-2.0/> ↗.

<http://clusterlabs.org/> ↗

Home page of Pacemaker, the cluster resource manager shipped with the High Availability Extension.

<http://www.clusterlabs.org/doc/> ↗

Holds a number of comprehensive manuals and some shorter documents explaining general concepts. For example:

- *Pacemaker Explained (Pacemaker 1.1 for Corosync 2.x and crmsh)* (see the section *Pacemaker 1.1 for Corosync 2.x and crmsh*): Contains comprehensive and very detailed information for reference.
- *Configuring Fencing with crmsh*: How to configure and use STONITH devices.
- *Colocation Explained*
- *Ordering Explained*

<http://linux-ha.org> ↗

Home page of the The High Availability Linux Project.

5 Configuring and Managing Cluster Resources (Web Interface)

To configure and manage cluster resources, either use HA Web Konsole (Hawk), a Web-based user interface, or the `crm shell (crmsh)` command line utility.

Hawk 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 failback 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.

5.1 Hawk—Overview

Hawk's Web interface 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.

5.1.1 Starting Hawk and Logging In

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.

To use Hawk, the respective Web service must be started on the node that you want to connect to via the Web interface.

If you have set up your cluster with the scripts from the `ha-cluster-bootstrap` package, the Hawk service is already started. In that case, skip *Procedure 5.1, "Starting Hawk Services"* and proceed with *Procedure 5.2, "Logging In to the Hawk Web Interface"*.

PROCEDURE 5.1: STARTING HAWK SERVICES

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

```
root # systemctl status hawk.service
```

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

```
root # systemctl start hawk.service
```

If you want Hawk to start automatically at boot time, execute the following command:

```
root # systemctl enable hawk.service
```



Note: User Authentication

Hawk users must be members of the haclient group. The installation creates a Linux user named hacluster, who is added to the haclient group. When using the **ha-cluster-init** script for setup, a default password is set for the hacluster user.

Before starting Hawk, set or change the password for the hacluster user. Alternatively, create a new user which is a member of the haclient group.

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

PROCEDURE 5.2: LOGGING IN TO THE HAWK WEB INTERFACE

The Hawk Web interface uses the HTTPS protocol and port 7630.

1. On any machine, start a Web browser and make sure that JavaScript and cookies are enabled.
2. As URL, enter the IP address or hostname of any cluster node running the Hawk Web service. Alternatively, enter the address of any IPaddr(2) resource that the cluster operator may have configured:

```
https://HOSTNAME_OR_IP_ADDRESS:7630/
```



Note: Certificate Warning

If a certificate warning appears when you try to access the URL for the first time, a self-signed certificate is in use. Self-signed certificates are not considered trustworthy by default.

Ask your cluster operator for the certificate details to verify the certificate.

To proceed anyway, you can add an exception in the browser to bypass the warning.

For information on how to replace the self-signed certificate with a certificate signed by an official Certificate Authority, refer to [Replacing the Self-Signed Certificate](#).

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).

4. Click *Log In*.

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

The information shown is similar to the output of crm status in the crm shell.

5.1.2 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, unclear. 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 or a resource is currently being started or stopped. If you click a pending resource to view its details, Hawk also displays the state to which the resource is currently changing (Starting, Stopping, Moving, Promoting, or Demoting).



Note: Displaying Transitional State for Resources

The transitional state for resources is only shown if the operation property `record-pending` is set to `true`. If you have set up your cluster with the `ha-cluster-init` script, this property is turned on globally by default. To enable it manually, either use the Hawk *Cluster Configuration* screen to add and enable the property below *Operation Defaults* or use the following command:

```
root # crm configure op_defaults record-pending=true
```

- 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).

In addition to the color code, Hawk also displays self-explanatory icons for the state of nodes, resources, tickets and for error messages in all views of the *Cluster Status* screen.

If a resource has failed, an error message with the details is shown in red at the top of the screen. To analyze the causes for the failure, click the error message. This automatically takes you to Hawk's *History Explorer* and triggers the collection of data for a time span of 20 minutes (10 minutes before and 10 minutes after the failure occurred). For more details, refer to [Procedure 5.27, "Viewing Transitions with the History Explorer"](#).

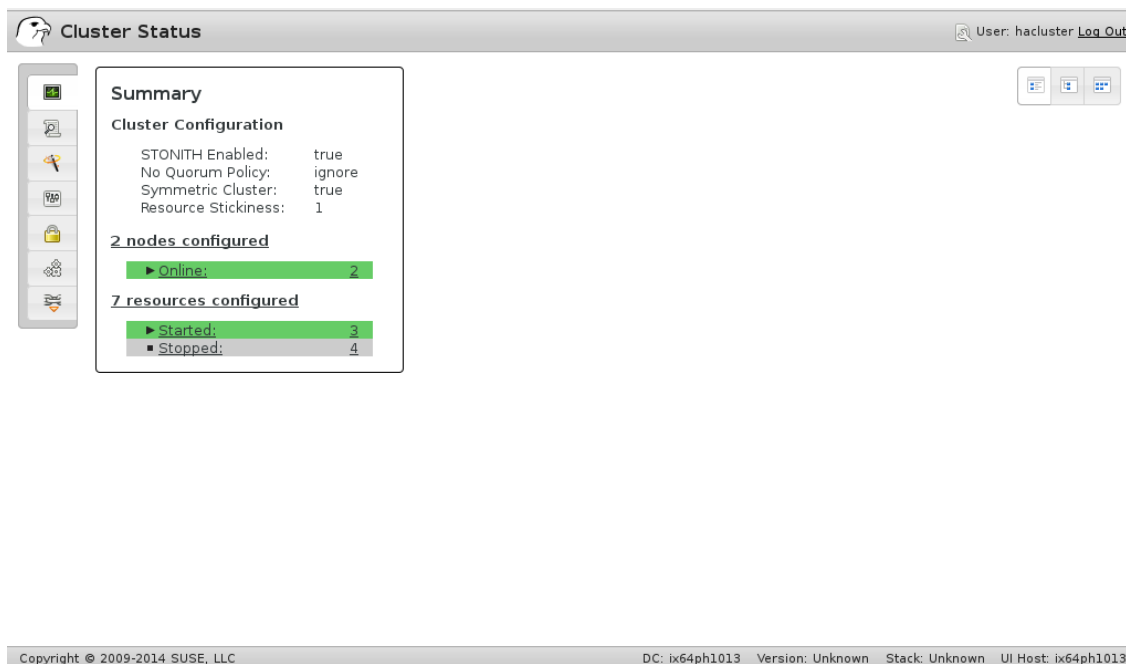


FIGURE 5.1: HAWK—CLUSTER STATUS (SUMMARY VIEW)

The *Cluster Status* screen refreshes itself in near real-time. Choose between the following views, which you can access with the three icons in the upper right corner:

HAWK CLUSTER STATUS 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 GEO clusters (multi-site clusters), 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 a 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. If your setup includes GEO clusters (multi-site clusters), the tree view also shows tickets. 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).

For groups, it is also possible to switch to a view where resources of the same type (within a group) are presented together. Press the ab icon in the resources category to toggle between the normal view where the resources are displayed individually and the

view where they are coalesced by type. For example, if you have 3 resources of the type `ocf:pacemaker:Dummy` in one group, and only one of them is running, the view-by-type view shows an entry like `1/3 ocf:pacemaker:Dummy NODENAME` on a grey background, to indicate only 1 of the three has already started.

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.

The top-level row of the main screen shows the username with which you are logged in. It also allows you to *Log Out* of the Web interface, and to access the following *Tools* from the wrench icon next to the username:

- *Simulator*. For details, refer to [Section 5.4.7, “Exploring Potential Failure Scenarios”](#).
- *Cluster Diagram*. Select this entry for a graphical representation of the nodes and the resources configured in the CIB. The diagram also shows the ordering and colocation between resources and node assignments (scores).

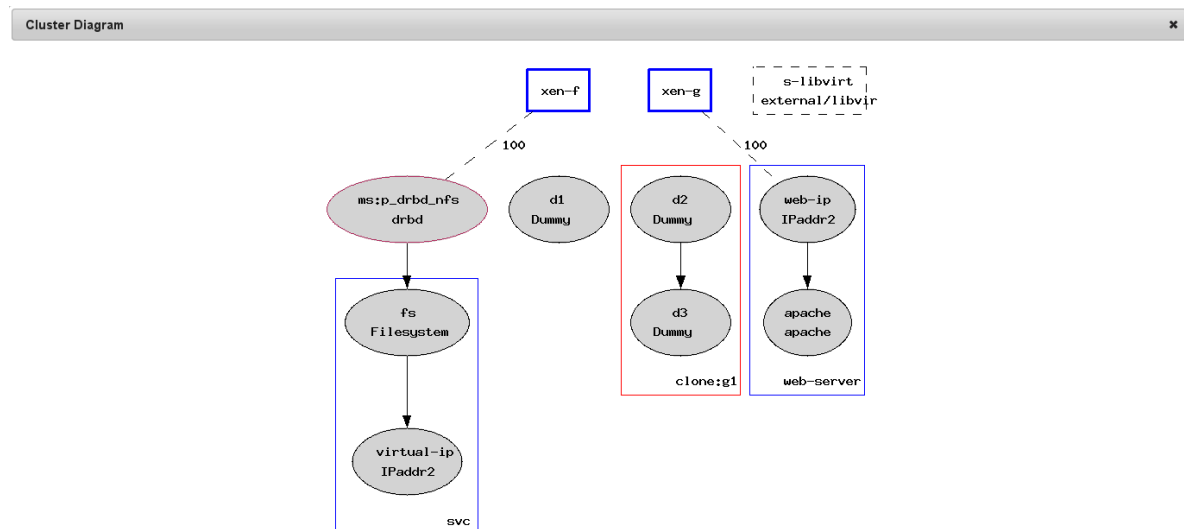


FIGURE 5.2: HAWK—CLUSTER DIAGRAM

- *Cluster Report* (`hb_report`). For details, refer to [Section 5.4.8, “Generating a Cluster Report”](#).

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 any clone, group or master/slave child resource on any of the status screens, select the *Parent* menu item from the context menu. Clicking this will let you start, stop, etc. the top-level clone or group to which that primitive belongs.

For more complex tasks like configuring resources, constraints, or global cluster options, use the navigation bar on the left hand side. From there, you can access the following screens:

- *Cluster Status*: See [Section 5.1.2, “Main Screen: Cluster Status”](#) for details.
- *History Explorer*: See [Procedure 5.27, “Viewing Transitions with the History Explorer”](#) for details.
- *Setup Wizard*: See [Section 5.3.1, “Configuring Resources with the Setup Wizard”](#) for details.
- *Cluster Configuration*: See [Section 5.2, “Configuring Global Cluster Options”](#) for details.
- *Access Control Lists*: See [Chapter 9, Access Control Lists](#) for details.
- *Resources*: See [Section 5.3, “Configuring Cluster Resources”](#) for details.
- *Constraints*: See [Section 5.3, “Configuring Cluster Resources”](#) for details.



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* 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 an hb_report.
- Using the history explorer.
- Viewing recent events for nodes or resources.

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 cluster management tools like 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*
- *Option stonith-enabled*

PROCEDURE 5.3: MODIFYING GLOBAL CLUSTER OPTIONS

1. Start a Web browser and log in to the cluster as described in *Section 5.1.1, “Starting Hawk and Logging In”*.
2. In the left navigation bar, select *Cluster Properties* to view the global cluster options and their current values. Hawk displays the most important parameters with regards to *CRM Configuration*, *Resource Defaults*, and *Operation Defaults*.

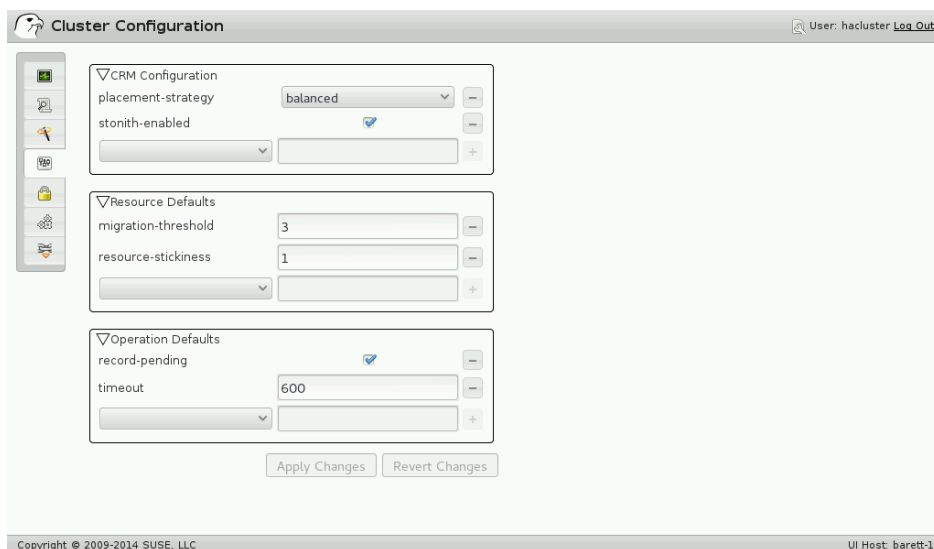


FIGURE 5.3: HAWK—CLUSTER CONFIGURATION

3. Depending on your cluster requirements, adjust the *CRM Configuration*:
 - a. Set *no-quorum-policy* to the appropriate value.
 - b. If you need to disable fencing for any reasons, deselect *stonith-enabled*.

Important: No Support Without STONITH

A cluster without STONITH is not supported.

- c. To remove a property from the CRM configuration, click the minus icon next to the property. If a property is deleted, the cluster will behave as if that property had the default value. For details of the default values, refer to [Section 4.2.6, “Resource Options \(Meta Attributes\)”](#).
 - d. To add a new property for the CRM configuration, choose one from the drop-down list and click the plus icon.
- 4. If you need to change *Resource Defaults* or *Operation Defaults*, proceed as follows:
 - a. To change the value of defaults that are already displayed, just edit the value in the respective input field.
 - b. To add a new resource default or operation default, choose one from the empty drop-down list, click the plus icon and enter a value. If there are default values defined, Hawk proposes them automatically.
 - c. To remove a resource or operation default, click the minus icon next to the parameter. If no values are specified for *Resource Defaults* and *Operation Defaults*, the cluster uses the default values that are documented in [Section 4.2.6, “Resource Options \(Meta Attributes\)”](#) and [Section 4.2.8, “Resource Operations”](#).
- 5. Confirm your changes.

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, 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 the resource types you can create, refer to [Section 4.2.3, “Types of Resources”](#). 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 \(Parameters\)”](#).
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 \(Meta Attributes\)”](#).
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”](#).
When creating a resource, Hawk displays the most important resource operations (monitor, start, and stop).

5.3.1 Configuring Resources with the Setup Wizard

The High Availability Extension comes with a predefined set of templates for some frequently used cluster scenarios, for example, setting up a highly available NFS server. Find the predefined templates in the hawk-templates package. You can also define your own wizard templates. For detailed information, refer to <https://github.com/ClusterLabs/hawk/blob/master/doc/wizard.txt> [↗](#).

PROCEDURE 5.4: USING THE SETUP WIZARD

Hawk provides a wizard that guides you through all configuration steps of a selected template. Follow the instructions on the screen. If you need information about an option, click it to display a short help text in Hawk.



Note: Availability of Templates

Which templates are available to individual Hawk users may differ. The user's permissions to access templates can be regulated via ACL. See [Chapter 9, Access Control Lists](#) for details.

In the following procedure, we will use the wizard to configure an NFS server as example, which can be used as an NFS(v4/v3) fail-over server. The wizard relies on the cluster having been set up using the bootstrap scripts, so that key-based SSH access between nodes has been configured. You will be prompted for the following information:

- The root password for the machine that you are logged in to via Hawk. It needs to be the same as on all cluster nodes that Hawk needs to touch in order to modify their file systems.
- The ID of the base file system resource.
- Details for the NFSv4 file system root.
- Details for an NFSv3 export. A directory exported by an NFS server, which clients can integrate it into their system.
- A floating IP address.

The resulting Pacemaker configuration will contain the following resources:

NFS Kernel Server

Manages the in-kernel Linux NFS daemon that serves locally mounted file systems to clients via the NFS network protocol.

NFSv4 Virtual File System Root

Manages the virtual NFS root export, needed for NFSv4 clients. This resource does not hold any actual NFS-exported data, merely the empty directory (/srv/nfs) that the other NFS export is mounted into.

Non-root NFS Export

Manages the NFSv3 export .

Floating IP Address

A virtual, floating cluster IP address, allowing NFS clients to connect to the service no matter which physical node it is running on.

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, "Starting Hawk and Logging In"](#).
2. In the left navigation bar, select *Setup Wizard*. The *Cluster Setup Wizard* shows a list of available resource templates. If you click an entry, Hawk displays a short help text about the template.

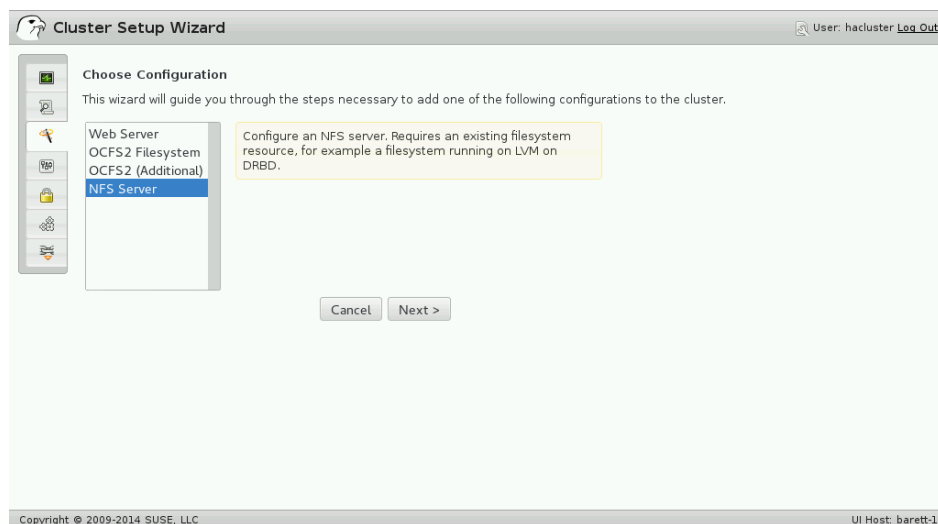


FIGURE 5.4: HAWK—SETUP WIZARD

3. Select the template for the resource you want to configure (in our case: NFS Server) and click *Next*.
4. To configure a highly available NFS server, proceed as follows:
 - a. Enter the root password for the current machine and click *Next*. Without the root password, the configuration wizard is not able to do the necessary configuration changes.
 - b. In the next screen, specify the *Base Filesystem* that is to be exported via NFS by entering its root ID. Click *Next*.
 - c. In the next screen, enter the details for the virtual NFSv4 file system root (needed for NFSv4 clients). Specify the following parameters and click *Next*.
 - Define a *Resource ID* to be used for this cluster resource.
 - Enter a *File System ID*. Hawk proposes 0 by default, as the ID for the root file system must either be 0 or the string root.
 - Specify a *Mount Point*, for example: /srv/nfs.

- Enter a *Client Spec* for client access. For example 10.9.9.0/255.255.255.0. If you keep the value *, which is proposed by Hawk, this would mean to allow all clients from everywhere.
 - Specify the *Mount Options*. For the NFSv4 file system root, Hawk proposes: rw,crossmnt.
- d. In the next screen, enter the details for the exported NFS mount point. Specify the following parameters and click *Next*.
- Define a *Resource ID* to be used for this cluster resource.
 - Enter a *File System ID*. Hawk proposes 1 by default, as the ID for NFS exports that do *not* represent an NFSv4 virtual file system root must be set to a unique positive integer, or a UUID string (32 hexadecimal digits with arbitrary punctuation).
 - Specify a *Mount Point*, for example: /srv/nfs/example.
 - Enter a *Client Spec* for client access. For example 10.9.9.0/255.255.255.0. If you keep the value *, which is proposed by Hawk, this would mean to allow all clients from everywhere.
 - Specify the *Mount Options*. For the NFSv3 export, Hawk proposes: rw,mountpoint.
- e. In the next screen, configure a virtual IP added used to access the NFS mounts. Specify the following parameters.
- Define a *Resource ID* to be used for this cluster resource.
 - Enter an *IP Address* in dotted quad notation.
 - Optionally, enter a *Netmask*. If not specified, it will be determined automatically.
 - For LVS Direct Routing configuration, enable *LVS Support*. Otherwise leave it disabled.
- f. Click *Next*.
- The wizard displays the configuration snippet that will be applied to the CIB.



g. To apply it, click *Next*.

You have successfully configured an NFS(v4/v3) fail-over server.

5.3.2 Creating Simple Cluster Resources

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

PROCEDURE 5.5: ADDING PRIMITIVE RESOURCES

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, "Starting Hawk and Logging In"](#).
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:
 - a. Enter a unique *Resource ID*.
 - b. From the *Class* list, select the resource agent class you want to use for the resource: *lsb*, *ocf*, *service*, or *stonith*. For more information, see [Section 4.2.2, "Supported Resource Agent Classes"](#).
 - c. 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.

- d. 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:
 - a. Enter values for each required parameter. A short help text is displayed as soon as you click the input field next to a parameter.
 - b. To completely remove a parameter, click the minus icon next to the parameter.
 - c. To add another parameter, click the empty 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 5.15, "Adding or Modifying Monitor Operations"](#).
7. Hawk automatically lists the most important meta attributes for the resource, for example target-role.
To modify or add *Meta Attributes*:
 - a. To set a (different) value for an attribute, select one from the drop-down list next to the attribute or edit the value in the input field.
 - b. To completely remove a meta attribute, click the minus icon next to it.
 - c. 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 5.5: HAWK—PRIMITIVE RESOURCE

5.3.3 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 8.3, “STONITH Configuration”](#).

PROCEDURE 5.6: ADDING A STONITH RESOURCE

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, “Starting Hawk and Logging In”](#).
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:
 - a. Enter a unique *Resource ID*.
 - b. From the *Class* list, select the resource agent class *stonith*.
 - c. 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 8, Fencing and STONITH](#).

5.3.4 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"](#).

PROCEDURE 5.7: CREATING RESOURCE TEMPLATES

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, "Starting Hawk and Logging In"](#).
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 5.5: Adding Primitive Resources*, starting with *Step 4.b*.
6. Click *Create Resource* to finish the configuration. A message at the top of the screen shows if the resource template was successfully created.

Create Template User: hacluster [Log Out](#)

Template ID	Class	Provider	Type
dummy_template	ocf	heartbeat	Dummy

Parameters

Operations

Operation	Timeout	Interval	Action
monitor	timeout: 20	interval: 10	[-]
start	timeout: 20		[-]
stop	timeout: 20		[-]

Meta Attributes

[Create Resource](#) [Back](#)

Example stateless resource agent

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.

Copyright © 2009-2014 SUSE, LLC UI Host: barett-1

FIGURE 5.6: HAWK—RESOURCE TEMPLATE

PROCEDURE 5.8: REFERENCING RESOURCE TEMPLATES

1. Start a Web browser and log in to the cluster as described in *Section 5.1.1, "Starting Hawk and Logging In"*.
2. To reference the newly created resource template in a primitive, follow these steps:
 - a. In the left navigation bar, select *Resources*. The *Resources* screen shows categories for all types of resources. It lists all defined resources.
 - b. Select the *Primitive* category and click the plus icon.
 - c. Enter a unique *Resource ID*.
 - d. Activate *Use Template* and, from the drop-down list, select the template to reference.

- e. If needed, specify further *Parameters*, *Operations*, or *Meta Attributes* as described in *Procedure 5.5, "Adding Primitive Resources"*.
3. To reference the newly created resource template in colocal or order constraints, proceed as described in *Procedure 5.10, "Adding or Modifying Colocal or Order Constraints"*.

5.3.5 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"*. 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"*.

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

PROCEDURE 5.9: 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 5.1.1, "Starting Hawk and Logging In"*.
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.

Constraint created successfully

Constraint ID: my_loc_constraint

Resource: g-test, my_primitive (selected), stonith

Show Rule Editor: ☐

Score: INFINITY

Node: barett-2

Buttons: Apply Changes, Revert Changes, Back

Copyright © 2009-2014 SUSE, LLC UI Host: barett-1

FIGURE 5.7: HAWK—LOCATION CONSTRAINT

PROCEDURE 5.10: 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 5.1.1, "Starting Hawk and Logging In"*.
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:
 - a. 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.
 - b. 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.
 - c. 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 dependent resource to run at all.
 - d. Add as many resources as needed for your colocation or order constraint.
 - e. 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.

The screenshot shows the 'Create Colocation Constraint' web interface. At the top, there's a header with a user icon and 'User: hacluster Log Out'. The main area has a sidebar with icons on the left. The main content area contains the following fields and controls:

- Constraint ID:** A text input field containing 'col-r0-on-drbd'.
- Score:** A text input field containing 'INFINITY'.
- Resource Selection:** Two dropdown menus. The first is labeled 'drbd_group' and has a downward arrow. The second is labeled 'ms_drbd' and has 'Master' selected. There are minus and plus buttons next to each dropdown.
- Add resource to constraint:** A section with a dropdown menu and a plus button.
- Buttons:** 'Create Constraint' and 'Back' buttons at the bottom right.

The footer of the interface displays 'Copyright © 2009-2014 SUSE, LLC' on the left and 'UI Host: barett-2' on the right.

FIGURE 5.8: HAWK—COLOCATION CONSTRAINT

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

As of SUSE Linux Enterprise High Availability Extension 12, it is now also possible to use resource sets within location constraints (whereas they could formerly only be used within colocation and ordering constraints).

PROCEDURE 5.11: USING RESOURCE SETS FOR CONSTRAINTS

1. Start a Web browser and log in to the cluster as described in *Section 5.1.1, “Starting Hawk and Logging In”*.
2. To use a resource set within a location constraint:
 - a. Proceed as outlined in *Procedure 5.9, “Adding or Modifying Location Constraints”* from *Step 1* to to *Step 5*.
 - b. Instead of only selecting a single resource, you can select multiple resources by pressing **Ctrl** or **Shift** in addition to the mouse click. This creates a resource set within the location constraint.

- c. Continue by entering a *Score* and by selecting a *Node* for the constraint.
 - d. To remove a resource from the location constraint, press **Ctrl** and click the resource again to deselect it. If you are editing an existing location constraint, click *Apply Changes* to confirm your choice.
3. To use a resource set within a colocation or order constraint:
- a. Proceed as described in *Procedure 5.10, "Adding or Modifying Colocational or Order Constraints"*.
 - b. 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.
 - c. You can also add multiple resources to a resource set or create multiple resource sets.
 - d. To extract a resource from a resource set, click the scissors icon above the respective resource.

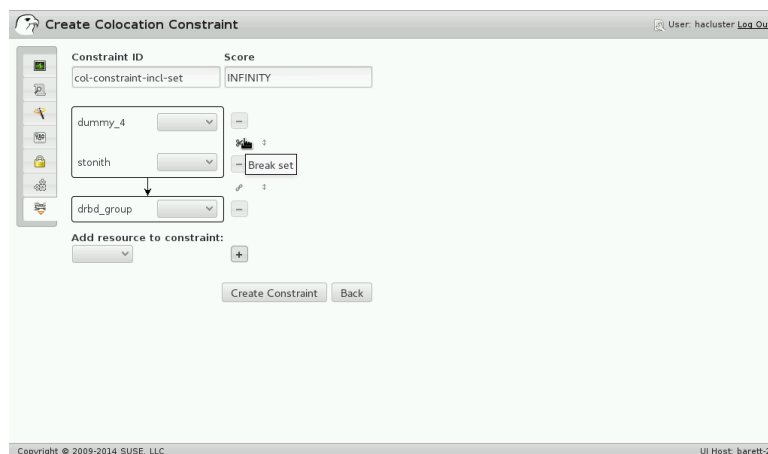



FIGURE 5.9: HAWK—REMOVING A RESOURCE FROM A SET

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

4. 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/> :

- *Pacemaker Explained (Pacemaker 1.1 for Corosync 2.x and crmsh)*, chapter *Resource Constraints*
- *Colocation Explained*
- *Ordering Explained*

PROCEDURE 5.12: REMOVING CONSTRAINTS

1. Start a Web browser and log in to the cluster as described in *Section 5.1.1, "Starting Hawk and Logging In"*.
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*.

5.3.6 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 (a 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 5.1.1, "Starting Hawk and Logging In"*.
2. Configure a location constraint for the resource as described in *Procedure 5.9, "Adding or Modifying Location Constraints"*.
3. Add the migration-threshold meta attribute to the resource as described in *Procedure 5.5: Adding Primitive Resources, Step 7* 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-time-out` meta attribute to the resource as described in *Procedure 5.5: Adding Primitive Resources, Step 7* 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 4.6, "Migration Threshold—Process Flow"*.

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"* for details.

5.3.7 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"*.

PROCEDURE 5.13: SPECIFYING RESOURCE STICKINESS

1. Start a Web browser and log in to the cluster as described in *Section 5.1.1, "Starting Hawk and Logging In"*.
2. Add the `resource-stickiness` meta attribute to the resource as described in *Procedure 5.5: Adding Primitive Resources, Step 7*.
3. Specify a value between `-INFINITY` and `INFINITY` for the resource-stickiness.

5.3.8 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"](#).

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*.

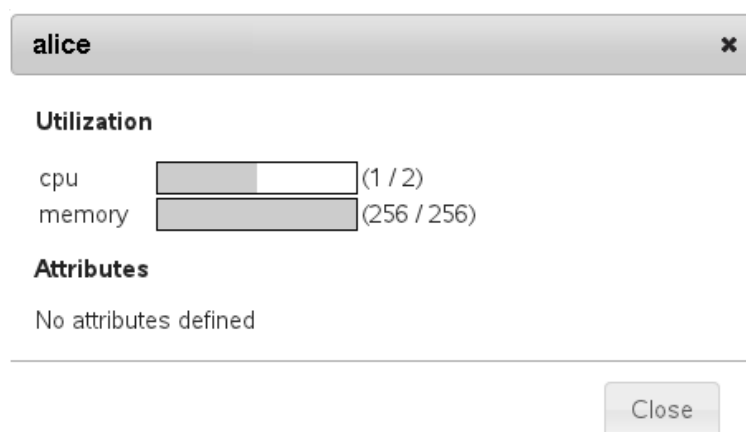


FIGURE 5.10: HAWK—VIEWING A NODE'S CAPACITY VALUES

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"](#).

PROCEDURE 5.14: SETTING THE PLACEMENT STRATEGY

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, “Starting Hawk and Logging In”](#).
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.

5.3.9 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 5.15: ADDING OR MODIFYING MONITOR OPERATIONS

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, “Starting Hawk and Logging In”](#).
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:
 - a. Click the pen icon next to the operation.
 - b. 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?).

The screenshot shows a dialog box titled "monitor". It contains the following fields and controls:

- interval**: Input field with value "120", followed by a minus button (-).
- timeout**: Input field with value "15", followed by a minus button (-).
- start-delay**: Input field with value "15", followed by a minus button (-).
- on-fail**: A dropdown menu showing "stop", followed by a plus button (+).

At the bottom of the dialog are two buttons: "OK" and "Cancel".

- c. 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 by clicking the pen icon.
 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"](#).

To view resource failures, switch to the *Cluster Status* screen in Hawk and select the resource you are interested in. Click the wrench icon next to the resource and select *Show Details*.

5.3.10 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”](#).



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 5.5, “Adding Primitive Resources”](#).

PROCEDURE 5.16: ADDING A RESOURCE GROUP

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, “Starting Hawk and Logging In”](#).
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](#).
7. Click *Create Group* to finish the configuration. A message at the top of the screen shows if the group was successfully created.

FIGURE 5.11: HAWK—RESOURCE GROUP

5.3.11 Configuring a Clone Resource

If you want certain resources to run simultaneously on multiple nodes in your cluster, configure these resources as a 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”](#).



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 5.5, “Adding Primitive Resources”](#) or [Procedure 5.16, “Adding a Resource Group”](#).

PROCEDURE 5.17: ADDING OR MODIFYING CLONES

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, “Starting Hawk and Logging In”](#).

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 5.5: Adding Primitive Resources, Step 7](#).
7. Click *Create Clone* to finish the configuration. A message at the top of the screen shows if the clone was successfully created.

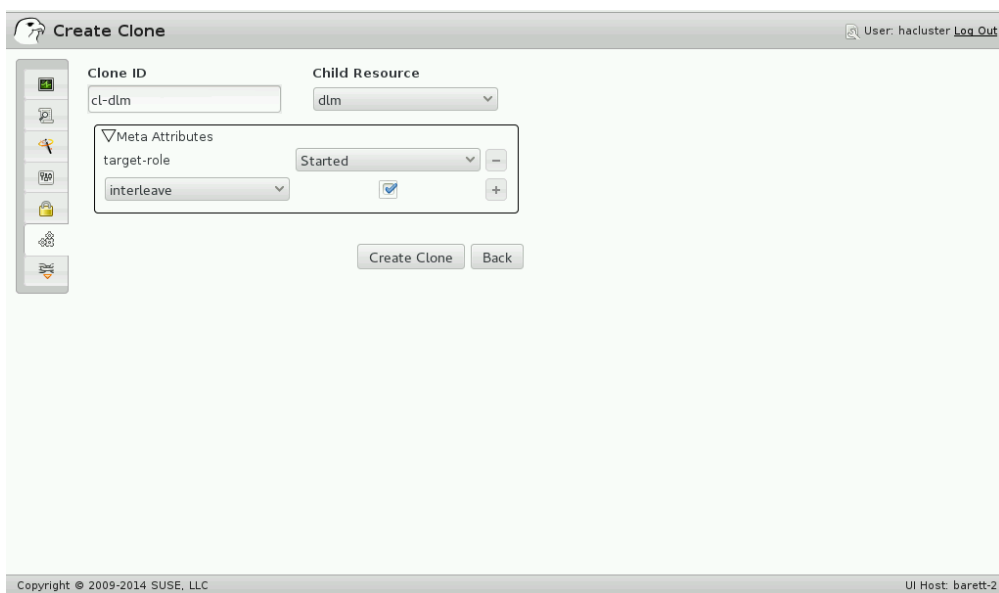


FIGURE 5.12: HAWK—CLONE RESOURCE

5.4 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 5.1.2, “Main Screen: Cluster Status”](#).

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. The detailed

view also shows any attributes set for that resource. For primitive resources (regular primitives, children of groups, clones, or master/slave resources), the following information will be shown additionally:

- the resource's failcount
the last failure timestamp (if the failcount is > 0)
- operation history and timings (call id, operation, last run timestamp, execution time, queue time, return code and last change timestamp)

The screenshot shows a window titled 'stonith' with a close button (X) in the top right corner. The window displays details for two resources: 'barett-1' and 'barett-2'. Under 'barett-1', it shows 'Fail Count: 1000000' and 'Last Failure: 2014-08-04 17:00:56'. Below this is a table of operations. Under 'barett-2', it shows 'Fail Count: 1000000' and 'Last Failure: 2014-08-04 17:01:00', followed by another table of operations. A 'Close' button is at the bottom right.

Call ID	Operation	Last Run	Exec	Queue	RC	Last Change
43	start	2014-08-04 17:00:52	3628ms	0ms	1	2014-08-04 17:00:52
48	stop	2014-08-04 17:00:56	1ms	0ms	0	2014-08-04 17:00:56

Call ID	Operation	Last Run	Exec	Queue	RC	Last Change
48	start	2014-08-04 17:00:56	3425ms	0ms	1	2014-08-04 17:00:56
49	stop	2014-08-04 17:01:01	0ms	0ms	0	2014-08-04 17:01:01

FIGURE 5.13: VIEWING A RESOURCE'S DETAILS

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 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 maintenance mode first as described in *Procedure 5.23, "Applying Maintenance Mode to Resources"*.

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 5.18: STARTING A NEW RESOURCE

1. Start a Web browser and log in to the cluster as described in *Section 5.1.1, "Starting Hawk and Logging In"*.
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.

5.4.2 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 5.19: CLEANING UP A RESOURCE

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, “Starting Hawk and Logging In”](#).
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`.

5.4.3 Removing Cluster Resources

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

PROCEDURE 5.20: REMOVING A CLUSTER RESOURCE

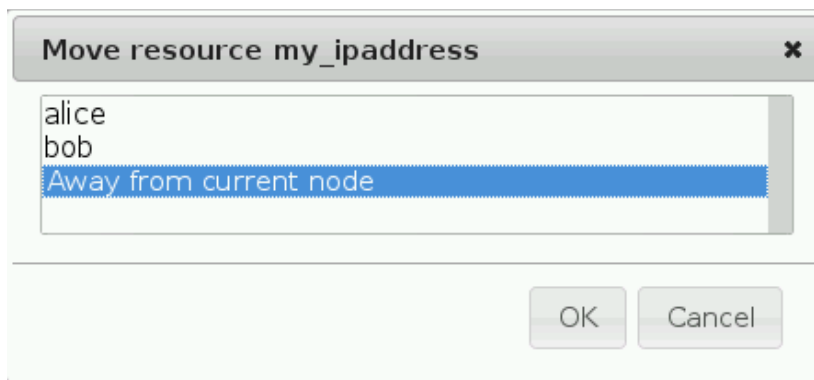
1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, “Starting Hawk and Logging In”](#).
2. In the left navigation bar, select *Cluster Status*.
3. Clean up the resource on all nodes as described in [Procedure 5.19, “Cleaning Up A Resource”](#).
4. In one of the individual resource views, click the wrench icon next to the resource and select *Stop*. To continue, confirm the message that appears.
5. If the resource is stopped, click the wrench icon next to it and select *Delete Resource*.

5.4.4 Migrating Cluster Resources

As mentioned in [Section 5.3.6, “Specifying Resource Failover Nodes”](#), 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 5.21: MANUALLY MIGRATING A RESOURCE

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, “Starting Hawk and Logging In”](#).
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 5.22: CLEARING A MIGRATION CONSTRAINT

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, “Starting Hawk and Logging In”](#).
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* (*Pacemaker 1.1 for Corosync 2.x and crmsh*), available from <http://www.clusterlabs.org/doc/> . Refer to section *Resource Migration*.

5.4.5 Using Maintenance Mode

Every now and then, you will need to perform testing or maintenance tasks on individual cluster components or the whole cluster—be it changing the cluster configuration, updating software packages for individual nodes, or upgrading the cluster to a higher product version.

With regards to that, High Availability Extension provides `maintenance` options on several levels:

- *Applying Maintenance Mode to Resources*
- *Applying Maintenance Mode to Nodes*
- *Applying Maintenance Mode to the Cluster*



Warning: Risk of Data Loss

If you need to execute any testing or maintenance tasks while services are running under cluster control, make sure to follow this outline:

1. Before you start, set the individual resource, node or the whole cluster to maintenance mode. This helps to avoid unwanted side effects like resources not starting in an orderly fashion, the risk of unsynchronized CIBs across the cluster nodes or data loss.
2. Execute your maintenance task or tests.
3. After you have finished, remove the maintenance mode to start normal cluster operation.

For more details on what happens to the resources and the cluster while being in maintenance mode, see *Section 4.7, “Maintenance Mode”*.

PROCEDURE 5.23: APPLYING MAINTENANCE MODE TO RESOURCES

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, "Starting Hawk and Logging In"](#).
2. In the left navigation bar, select *Resources*. Select the resource you want to put in maintenance mode or 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 *maintenance* attribute and click the plus icon to add it.
5. Activate the checkbox next to maintenance to set the maintenance attribute to yes.
6. Confirm your changes.
7. After you have finished the maintenance task for that resource, deactivate the checkbox next to the maintenance attribute for that resource.
From this point on, the resource will be managed by the High Availability Extension software again.

PROCEDURE 5.24: APPLYING MAINTENANCE MODE TO NODES

Sometimes it is necessary to put single nodes into maintenance mode. If your cluster consists of more than 3 nodes, you can easily set one node to maintenance mode, while the other nodes continue their normal operation.

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, "Starting Hawk and Logging In"](#).
2. In the left navigation bar, select *Cluster Status*.
3. In one of the individual nodes' views, click the wrench icon next to the node and select *Maintenance*.
This will add the following instance attribute to the node: maintenance="true". The resources previously running on the maintenance-mode node will become unmanaged. No new resources will be allocated to the node until it leaves the maintenance mode.
4. To deactivate the maintenance mode, click the wrench icon next to the node and select *Ready*.

PROCEDURE 5.25: APPLYING MAINTENANCE MODE TO THE CLUSTER

For setting or unsetting the maintenance mode for the whole cluster, proceed as follows:

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, “Starting Hawk and Logging In”](#).
2. In the left navigation bar, select *Cluster Configuration*.
3. In the *CRM Configuration group*, , select the *maintenance-mode* attribute from the empty drop-down box and click the plus icon to add it.
4. To set maintenance-mode=true, active the checkbox next to maintenance-mode and confirm your changes.
5. After you have finished the maintenance task for the whole cluster, deactivate the checkbox next to the maintenance-mode attribute.

From this point on, High Availability Extension will take over cluster management again.

5.4.6 Viewing the Cluster History

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

PROCEDURE 5.26: VIEWING RECENT EVENTS OF NODES OR RESOURCES

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, “Starting Hawk and Logging In”](#).
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 5.27: VIEWING TRANSITIONS WITH THE HISTORY EXPLORER

The *History Explorer* provides transition information for a time frame that you can define. It also lists its previous runs and allows you to *Delete* reports that you no longer need. The history explorer uses the information provided by hb_report. You can also upload an hb_report archive that has been created offline (on a different cluster) and view the respective transitions with the *History Explorer*. See [Procedure 5.28, “Using the History Explorer Offline”](#).

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, “Starting Hawk and Logging In”](#).
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.

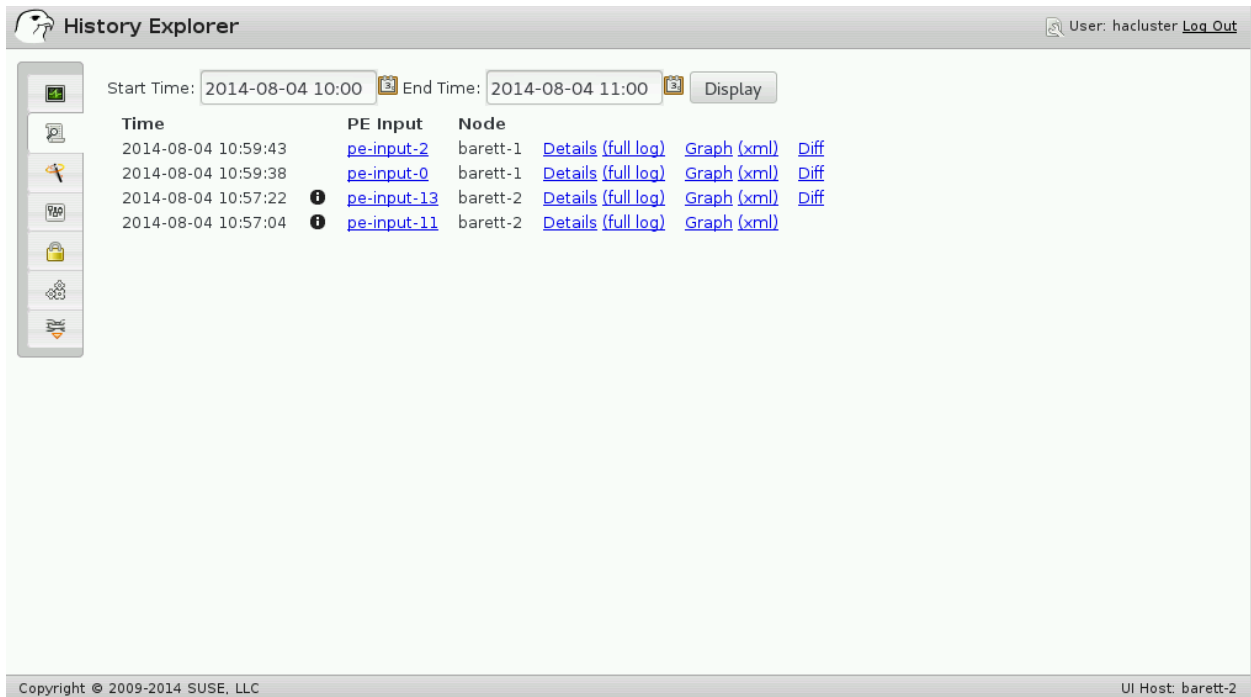


FIGURE 5.14: HAWK—HISTORY REPORT

The following information is displayed:

HISTORY EXPLORER RESULTS

Time

The time line of all past transitions in the cluster.

Information Icon

In some cases, an information icon is displayed between the *Time* and *PE Input* columns. Hover the mouse pointer over the icon to display one of the following messages:

- Input created by different Pacemaker version. In that case, the transition graphs are only approximate since they have been generated by a different PE version.
- Pacemaker version not present in PE Input. This happens during cluster start, before a DC is elected.

PE Input/Node

The pe-input* file for each transition and the node on which it was generated. 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 pe-input* files are only generated on the Designated Coordinator (DC), but as the DC can change, there may be pe-input* files from several nodes. The files show what the Policy Engine (PE) *planned* to do.

Details/Full Log

Opens a pop-up window with snippets of logging data that belong to that particular transition. Different amounts of details are available: Clicking *Details* displays the output of **crm history transition peinput** (including the resource agents' log messages), whereas *Full Log* also includes details from the pengi, crmd, and lrmd and is equivalent to **crm history transition log peinput**.

Graph/XML

A graph and an XML representation 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.

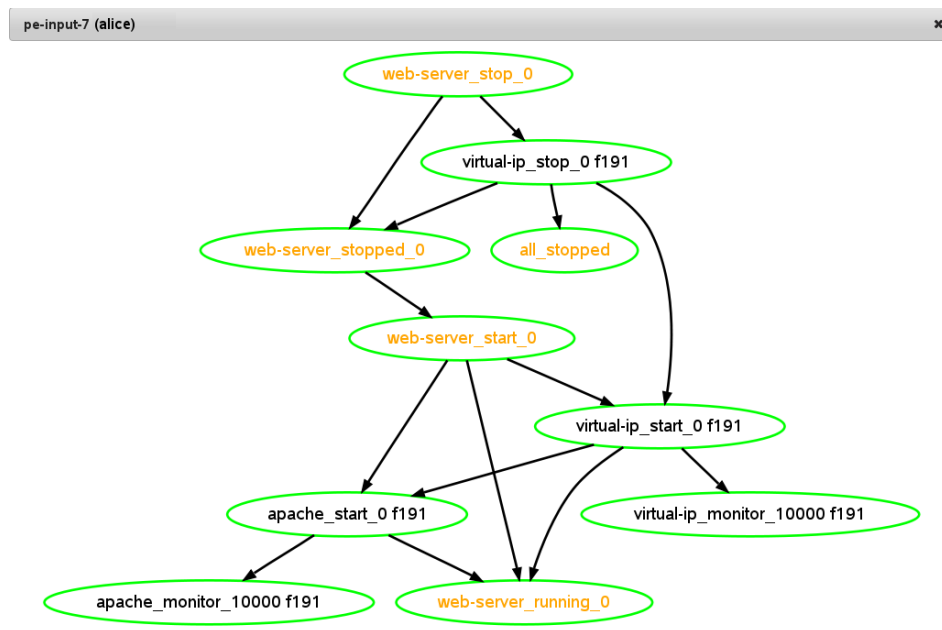


FIGURE 5.15: HAWK HISTORY REPORT—TRANSITION GRAPH

Diff

If two or more pe-inputs are listed, a *Diff* link will appear to the right of each pair of pe-inputs. Clicking it displays the difference of configuration and status.

PROCEDURE 5.28: USING THE HISTORY EXPLORER OFFLINE

If you have Hawk running on any machine, you can also use the *History Explorer* “offline”, that means to view and analyze the transitions of clusters that you are currently not connected to. All you need is a TAR archive with an `hb_report` generated on a SUSE Linux Enterprise High Availability Extension cluster. To upload and analyze it with the *History Explorer* proceed as follows:

1. Start a Web browser and log in to the Hawk Web interface as described in [Section 5.1.1, “Starting Hawk and Logging In”](#).
2. In the left navigation bar, select *History Explorer*. It shows an entry *Upload hb_report tarball*.

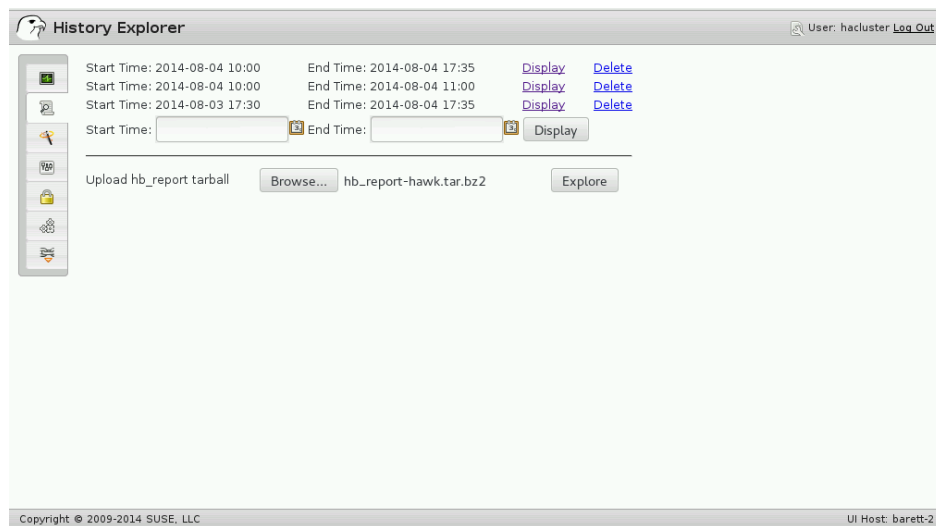


FIGURE 5.16: HAWK—OFFLINE USE OF HISTORY EXPLORER

3. Click *Browse* and select the `hb_report` archive to upload from your file system.
4. Click *Explore* to start the analysis of the archive and to display the information listed in *History Explorer Results*.

5.4.7 Exploring Potential Failure Scenarios

Hawk provides a *Simulator* that allows you to explore failure scenarios before they happen. After switching to the simulator mode, you can change the status of nodes, add or edit resources and constraints, change the cluster configuration, or execute multiple resource operations to see how the cluster would behave should these events occur. As long as the simulator mode is activated, a control dialog will be displayed in the bottom right hand corner of the *Cluster Status* screen. The simulator will collect the changes from all screens and will add them to its internal queue of events. The simulation run with the queued events will not be executed unless it is manually triggered in the control dialog. After the simulation run, you can view and analyze the details of what would have happened (log snippets, transition graph, and CIB states).

PROCEDURE 5.29: SWITCHING TO SIMULATOR MODE

1. Start a Web browser and log in to the cluster as described in *Section 5.1.1, "Starting Hawk and Logging In"*.
2. Activate the simulator mode by clicking the wrench icon in the top-level row (next to the username), and by selecting *Simulator*.

Hawk's background changes color to indicate the simulator is active. A simulator control dialog is displayed in the bottom right hand corner of the *Cluster Status* screen. Its title *Simulator (initial state)* indicates that no simulator run has occurred yet.

3. Fill the simulator's event queue:
 - a. To simulate status change of a node: Click **+Node** in the simulator control dialog. Select the *Node* you want to manipulate and select its target *State*. Confirm your changes to add them to the queue of events listed in the controller dialog.
 - b. To simulate a resource operation: Click **+Op** in the simulator control dialog. 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*. Confirm your changes to add them to the queue of events listed in the controller dialog.
4. Repeat the previous steps for any other node status changes or resource operations you wish to simulate.

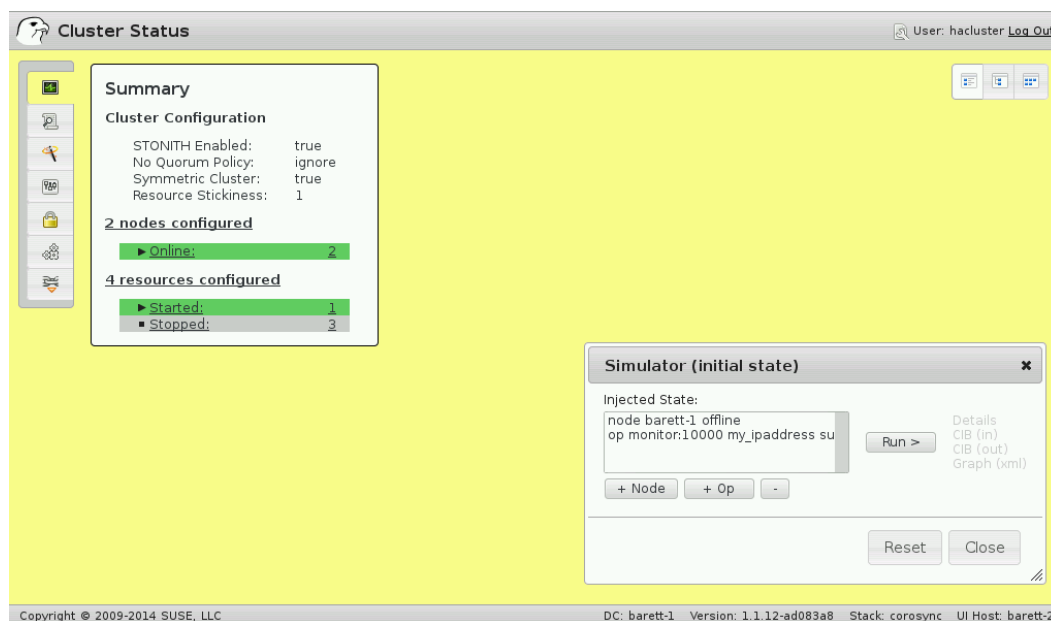


FIGURE 5.17: HAWK—SIMULATOR WITH INJECTED EVENTS

5. To inject other changes that you wish to simulate:
 - a. Switch to one or more of the following Hawk screens: *Cluster Status*, *Setup Wizard*, *Cluster Configuration*, *Resources*, or *Constraints*.



Clicking the *History Explorer* tab will deactivate simulator mode.

- b. Add or modify parameters on the screens as desired.
The simulator will collect the changes from all screens and will add them to its internal queue of events.
 - c. To return to the simulator control dialog, switch to the *Cluster Status* screen or click the wrench icon in the top-level row and click *Simulator* again.
6. If you want to remove an event listed in *Injected State*, select the respective entry and click the minus icon beneath the list.
7. Start the simulation run by clicking *Run* in the simulator control dialog. The *Cluster Status* screen displays the simulated events. For example, if you marked a node as unclear, it will now be shown offline, and all its resources will be stopped. The simulator control dialog changes to *Simulator (final state)*.

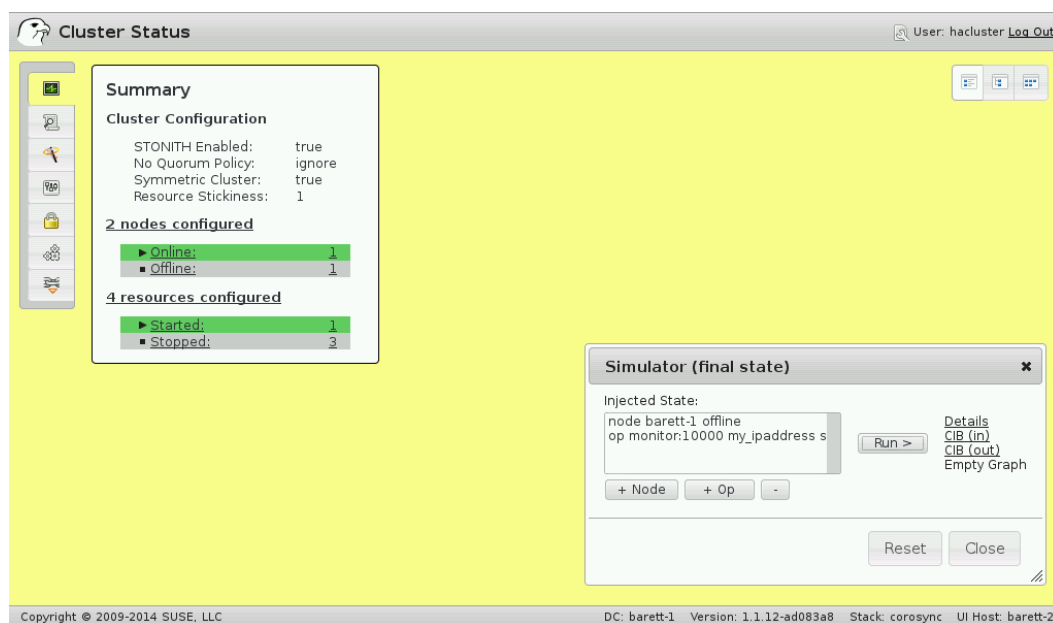


FIGURE 5.18: HAWK—SIMULATOR IN FINAL STATE

8. To view more detailed information about the simulation run:
 - a. Click the *Details* link in the simulator dialog to see log snippets of what occurred.

- b. Click the *Graph* link to show the transition graph.
 - c. Click *CIB (in)* to display the initial CIB state. 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.

5.4.8 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 5.30: GENERATING AN `hb_report`

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, "Starting Hawk and Logging In"](#).
2. Click the wrench icon next to the username in the top-level row, and 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 tools like `hb_report` and `crm_report` cover, refer to [How can I create a report with an analysis of all my cluster nodes?](#).

5.5 Monitoring Multiple Clusters

You can use Hawk as a single point of administration for monitoring multiple clusters. Hawk's *Cluster Dashboard* allows you to view a summary of multiple clusters, with each summary listing the number of nodes, resources, tickets (if you use GEO clusters), and their state. The summary also shows if any failures have appeared in the respective cluster.

The cluster information displayed in the *Cluster Dashboard* is stored in a persistent cookie. This means you need to decide which Hawk instance you want to view the *Cluster Dashboard* on, and always use that one. The machine you are running Hawk on does not even have to be part of any cluster for that purpose—it can be a separate, unrelated system.

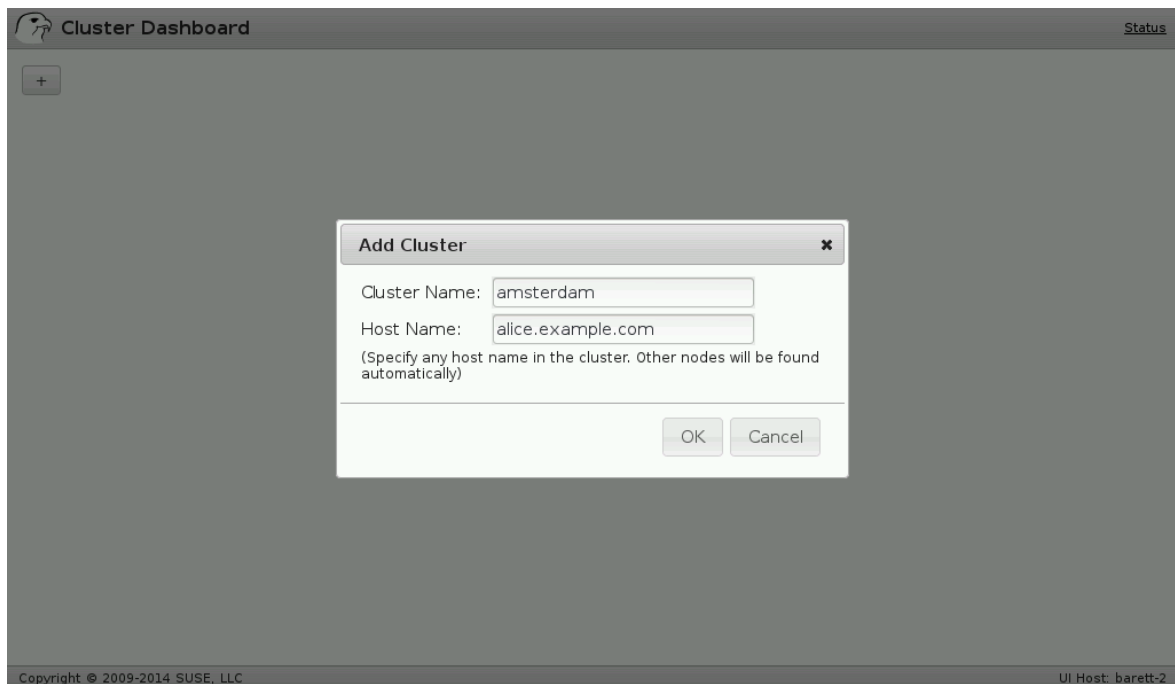
PROCEDURE 5.31: MONITORING MULTIPLE CLUSTERS WITH HAWK

PREREQUISITES

- All clusters to be monitored from Hawk's *Cluster Dashboard* must be running SUSE Linux Enterprise High Availability Extension 12. It is not possible to monitor clusters that are running earlier versions of SUSE Linux Enterprise High Availability Extension.
 - If you did not replace the self-signed certificate for Hawk on every cluster node with your own certificate (or a certificate signed by an official Certificate Authority), you must log in to Hawk on *every* node in *every* cluster at least once. Verify the certificate (and add an exception in the browser to bypass the warning).
 - If you are using Mozilla Firefox, you must change its preferences to *Accept third-party cookies*. Otherwise cookies from monitored clusters will not be set, thus preventing login to the clusters you are trying to monitor.
1. Start the Hawk Web service on a machine you want to use for monitoring multiple clusters.
 2. Start a Web browser and as URL enter the IP address or hostname of the machine that runs Hawk:

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

3. On the Hawk login screen, click the *Dashboard* link in the right upper corner. The *Add Cluster* dialog appears.



4. Enter a custom *Cluster Name* with which to identify the cluster the *Cluster Dashboard*.
5. Enter the *Host Name* of one of the cluster nodes and confirm your changes.
The *Cluster Dashboard* opens and shows a summary of the cluster you just added.
6. To add more clusters to the dashboard, click the plus icon and enter the details for the next cluster.

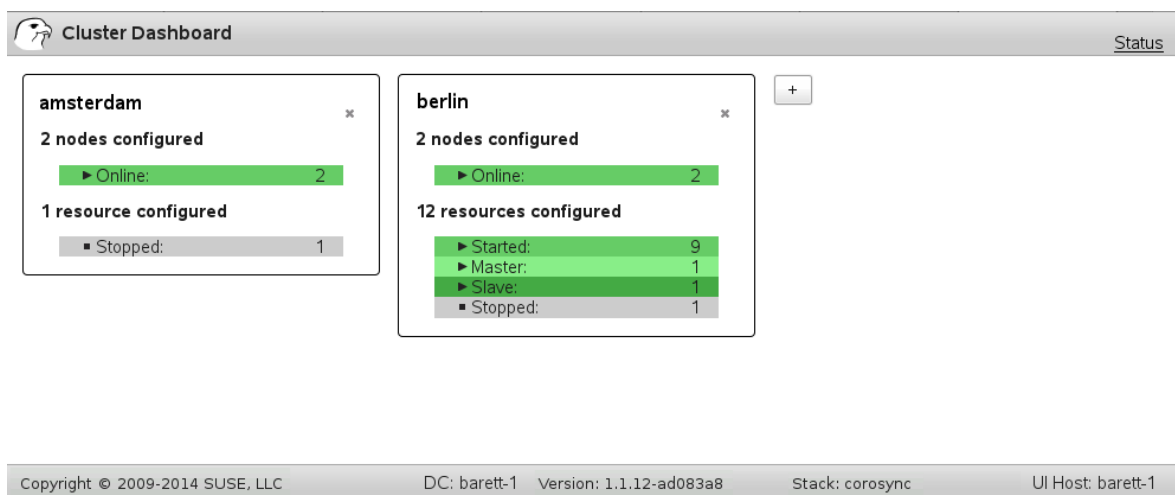


FIGURE 5.19: HAWK—CLUSTER DASHBOARD

7. To remove a cluster from the dashboard, click the x icon next to the cluster's summary.

8. To view more details about a cluster, click somewhere into the cluster's box on the dashboard.

This opens a new browser window or new browser tab. If you are not currently logged in to the cluster, this takes you to the Hawk login screen. After having logged in, Hawk shows the *Cluster Status* of that cluster in the summary view. From here, you can administrate the cluster with Hawk as usual.

9. As the *Cluster Dashboard* stays open in a separate browser window or tab, you can easily switch between the dashboard and the administration of individual clusters in Hawk.

Any status changes for nodes or resources are reflected almost immediately within the *Cluster Dashboard*.

5.6 Hawk for GEO Clusters

For more details on Hawk features that relate to geographically dispersed clusters (GEO clusters), see the *Quick Start GEO Clustering for SUSE Linux Enterprise High Availability Extension*.

5.7 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. Where Pacemaker logs to is specified in the `logging` section of `/etc/corosync/corosync.conf`.

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 `systemctl stop nscd.service` 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.

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
```

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
```

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.

Cluster Dashboard: Unable to connect to host

If adding clusters to Hawk's dashboard fails, check the prerequisites listed in [Procedure 5.31, "Monitoring Multiple Clusters with Hawk"](#).

Cluster Dashboard: Node Not Accessible

The *Cluster Dashboard* only polls one node in each cluster for status. If the node being polled goes down, the dashboard will cycle to poll another node. In that case, Hawk briefly displays a warning message about that node being inaccessible. The message will disappear after Hawk has found another node to contact to.

6 Configuring and Managing Cluster Resources (Command Line)

To configure and manage cluster resources, either use the `crm` shell (`crmsh`) command line utility or HA Web Konsole (Hawk), a Web-based user interface.

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



Note: User Privileges

Sufficient privileges are necessary 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:

```
root # crm options user hacluster
```

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

6.1 `crmsh`—Overview

The `crm` command has several subcommands which manage resources, CIBs, nodes, resource agents, and others. It offers a thorough help system with embedded examples. All examples follow a naming convention described in [Appendix B](#).



Tip: Distinguish Between Shell Prompt and Interactive `crm` Prompt

To make all the code and examples more readable, this chapter uses the following notations between shell prompts and the interactive `crm` prompt:

- Shell prompt for user `root`:

```
root #
```

- Interactive `crmsh` prompt (displayed in green, if terminal supports colors):

```
crm(live)#
```

6.1.1 Getting Help

Help can be accessed in several ways:

- To output the usage of `crm` and its command line options:

```
root # crm --help
```

- To give a list of all available commands:

```
root # crm help
```

- To access other help sections, not just the command reference:

```
root # crm help topics
```

- To view the extensive help text of the `configure` subcommand:

```
root # crm configure help
```

- To print the syntax, its usage, and examples of a subcommand of `configure`:

```
root # crm configure help group
```

This is also possible:

```
root # crm help configure group
```

Almost all output of the **help** subcommand (do not mix it up with the **--help** option) opens a text viewer. This text viewer allows you to scroll up or down and read the help text more comfortably. To leave the text viewer, press the **Q** key.



Tip: Use Tab Completion in Bash and Interactive Shell

The **crmsh** supports full tab completion in Bash directly, not only for the interactive shell. For example, typing **crm help config** **→** will complete the word just like in the interactive shell.

6.1.2 Executing **crmsh**'s Subcommands

The **crm** command itself 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 subcommands in a script. This can be done in two ways:

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

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

```
# A small script file for crm  
status  
node list
```


Any line starting with the hash symbol (#) is a comment and is ignored. If a line is too long, insert a backslash (\) at the end and continue in the next line. It's recommended to indent lines which belongs to a certain subcommand to improve readability.

- **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 `back`, `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, crmsh 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.

6.1.3 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”*):

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

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

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

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

```
crm(live)ra# list ocf  
AoEtarget      AudibleAlarm    CTDB             ClusterMon  
Delay          Dummy           EvmsSCC          Evmsd  
Filesystem     HealthCPU       HealthSMART      ICP  
IPaddr         IPaddr2         IPsrcaddr        IPv6addr  
LVM            LinuxSCSI       MailTo           ManageRAID  
ManageVE       Pure-FTPd       Raid1            Route  
SAPDatabase    SAPIInstance    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 .



Tip: Use `crm` Directly

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

6.1.4 Using Configuration Templates

Configuration templates are ready-made cluster configurations for `crmsd`. Do not confuse them with the *resource templates* (as described in [Section 6.4.2, "Creating Resource Templates"](#)). 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 interactive shell:

```
root # crm configure
```

2. Create a new configuration from a configuration template:

- a. Switch to the template subcommand:

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

- b. List the available configuration templates:

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

- c. Decide which configuration template you need. As we need an Apache configuration, we choose the apache template and name it g-intranet:

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

3. Define your parameters:

- a. List the just created configuration:

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

- b. 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
```

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

```
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 3.c*):

```
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 g-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:

```
root # crm configure template \
    new g-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.

6.1.5 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 interactive shell:

```
root # crm configure
```

2. Create a new shadow configuration:

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

If you omit the name of the shadow CIB, a temporary name @tmp@ is 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)#
```

6.1.6 Debugging Your Configuration Changes

Before loading your configuration changes back into the cluster, it is recommended to review your changes with **ptest**. The **ptest** command 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:

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

6.1.7 Cluster Diagram

To output a cluster diagram as shown in *Figure 5.2, “Hawk—Cluster Diagram”*, use the command **crm configure graph**. It displays the current configuration on its current window, therefore requiring X11.

If you prefer Scalable Vector Graphics (SVG), use the following command:

```
root # crm configure graph dot config.svg svg
```

6.2 Managing Corosync Configuration

Corosync is the underlying messaging layer for most HA clusters. The **corosync** subcommand provides commands for editing and managing the Corosync configuration.

For example, to list the status of the cluster, use **status**:

```
root # crm corosync status
```

```

Printing ring status.
Local node ID 175704363
RING ID 0
    id      = 10.121.9.43
    status  = ring 0 active with no faults
Quorum information
-----
Date:           Thu May  8 16:41:56 2014
Quorum provider: corosync_votequorum
Nodes:          2
Node ID:        175704363
Ring ID:        4032
Quorate:        Yes

Votequorum information
-----
Expected votes:  2
Highest expected: 2
Total votes:     2
Quorum:          2
Flags:           Quorate

Membership information
-----
    Nodeid      Votes Name
175704363        1 alice.example.com (local)
175704619        1 bob.example.com

```

Very helpful is the **diff** command: It compares the Corosync configuration on all nodes (if not stated otherwise) and prints the difference between:

```

root # crm corosync diff
--- bob
+++ alice
@@ -46,2 +46,2 @@
-     expected_votes: 2

```



```
-      two_node: 1
+      expected_votes: 1
+      two_node: 0
```

For more details, see http://crmsh.nongnu.org/crm.8.html#cmdhelp_corosync.

6.3 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 6.1: MODIFYING GLOBAL CLUSTER OPTIONS WITH `crm`

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

```
root # crm configure
```

2. Use the following commands to set the options for two-node clusters only:

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



Important: No Support Without STONITH

A cluster without STONITH is not supported.

3. Show your changes:

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

4. Commit your changes and exit:

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

6.4 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”](#).

6.4.1 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”](#)). To add a new resource to the cluster, proceed as follows:

1. Log in as root and start the crm tool:

```
root # 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
```

6.4.2 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”](#) for some basic background information. Do not confuse them with the “normal” templates from [Section 6.1.4, “Using Configuration Templates”](#). Use the **rsc_template** command to get familiar with the syntax:

```
root # 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.

6.4.3 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** interactive shell:

```
root # 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/ibmrsa
external/ibmrsa-telnet  external/ipmi       external/ippower9258
```

external/kdumpcheck	external/libvirt	external/nut
external/rackpdu	external/riloe	external/sbd
external/vcenter	external/vmware	external/xen0
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="alice"
    ipaddr="192.168.1.221" \
    userid="admin" passwd="secret" \
    op monitor interval=60m timeout=120s
```

6.4.4 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"](#).

6.4.4.1 Locational Constraints

The **location** command defines on which nodes a resource may be run, may not be run or is preferred to be run.

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 **alice** to 100 would be the following:

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

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 loc-node_pref internal_www \  
    rule 50: #uname eq alice \  
    rule pingd: defined pingd
```

Another use case for location constraints are grouping primitives as a *resource set*. This can be useful if several resources depend on, for example, a ping attribute for network connectivity. In former times, the **-inf/ping** rules had to be duplicated several times in the configuration, making it unnecessarily complex.

The following example creates a resource set **loc-alice**, referencing to the virtual IP addresses **vip1** and **vip2**:

```
crm(live)configure# primitive vip1 ocf:heartbeat:IPAddr2 params ip=192.168.1.5  
crm(live)configure# primitive vip2 ocf:heartbeat:IPAddr2 params ip=192.168.1.6  
crm(live)configure# location loc-alice { vip1 vip2 } inf: alice
```

In some cases it is much more efficient and convenient to use resource patterns for your **location** command. A resource pattern is a regular expression between two slashes. For example, the above virtual IP addresses can be all matched with the following:

```
crm(live)configure# location loc-alice /vip.*/ inf: alice
```

6.4.4.2 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.

6.4.4.3 Collocating Sets for Resources Without Dependency

Sometimes it is useful to be able to place a group of resources on the same node (defining a colocation constraint), but without having hard dependencies between the resources.

Use the command **weak-bond** if you want to place resources on the same node, but without any action if one of them fails.

```
root # crm configure assist weak-bond RES1 RES2
```

The implementation of **weak-bond** creates a dummy resource and a colocation constraint with the given resources automatically.

6.4.4.4 Ordering Constraints

The **order** command defines a sequence of action.

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 command in the **crm** shell to configure an ordering constraint:

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

6.4.4.5 Constraints for the Example Configuration

The example used for this section 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
```



```
drbd_resource:promote filesystem_resource:start
```

6.4.5 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 rscl-alice rscl 100: alice
```

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

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

For an overview, refer to [Section 4.4.4, “Failover Nodes”](#).

6.4.6 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”](#).

6.4.7 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”](#).

To configure the resource's requirements and the capacity a node provides, use utilization attributes. You can name the utilization attributes according to your preferences and define as many name/value pairs as your configuration needs. In certain cases, some agents update the utilization themselves, for example the `VirtualDomain`.

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

PROCEDURE 6.2: ADDING OR MODIFYING UTILIZATION ATTRIBUTES WITH `crm`

1. Log in as `root` and start the `crm` interactive shell:

```
root # 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:

property placement-strategy=default

Utilization values are not taken into account at all, by default. Resources are allocated according to location scoring. If scores are equal, resources are evenly distributed across nodes.

property placement-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.

property placement-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.

property placement-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 crmsh:

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

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

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

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

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

6.4.8 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”](#).

6.4.9 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 id=p.public-ip  
crm(live)configure# primitive Email lsb:exim \  
    params id=p.lsb-exim
```

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

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

To change the order of a group member, use the `modgroup` command from the `configure` subcommand. Use the following commands to move the primitive `Email` before `Public-IP`. (This is just to demonstrate the feature):

```
crm(live)configure# modgroup g-shortcut add p.lsb-exim before p.public-ip
```

In case you want to remove a resource from a group (for example, `Email`), use this command:

```
crm(live)configure# modgroup g-shortcut remove p.lsb-exim
```

For an overview, refer to [Section 4.2.5.1, “Groups”](#).

6.4.10 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”](#).

6.4.10.1 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` interactive shell:

```
root # crm configure
```

2. Configure the primitive, for example:

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

3. Clone the primitive:

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

6.4.10.2 Creating Stateful/Multi-State Clone Resources

To create an stateful clone resource, first create a primitive resource and then the multi-state resource. The multi-state resource must support at least promote and demote operations.

1. Log in as `root` and start the `crm` interactive shell:

```
root # crm configure
```

2. Configure the primitive. Change the intervals if needed:

```
crm(live)configure# primitive my-rsc ocf:myCorp:myAppl \
```

```
op monitor interval=60 \  
op monitor interval=61 role=Master
```

3. Create the multi-state resource:

```
crm(live)configure# ms ms-rsc my-rsc
```

6.5 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 give you an overview.

6.5.1 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** interactive shell:

```
root # crm
```

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
```

6.5.2 Cleaning Up Resources

A resource will be automatically restarted if it fails, but each failure raises the resource's fail-count. 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:

```
root # 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:

```
root # 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.

6.5.3 Removing a Cluster Resource

Proceed as follows to remove a cluster resource:

1. Log in as root and start the crm interactive shell:

```
root # 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
```

6.5.4 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 Hawk or the command line.

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

```
root # crm resource  
crm(live)resource# migrate ipaddress1 bob
```

6.5.5 Grouping/Tagging Resources

Tags are a way to refer to multiple resources at once, without creating any colocation or ordering relationship between them. This can be useful for grouping conceptually related resources. For example, if you have a number of resources related to a database, create a tag called tag-db and add all resources related to the database to this tag:

```
root # crm configure tag-db db1 db2 db3
```

This allows you to start them all with a single command:

```
root # crm resource start tag-db
```

Similarly, you can stop them all too:

```
root # crm resource stop tag-db
```

6.5.6 Using Maintenance Mode

Every now and then, you will need to perform testing or maintenance tasks on individual cluster components or the whole cluster—be it changing the cluster configuration, updating software packages for individual nodes, or upgrading the cluster to a higher product version.

With regards to that, High Availability Extension provides maintenance options on several levels:

Applying Maintenance Mode to your Cluster

In case you want to put the whole cluster in maintenance mode, use the following command:

```
root # crm configure property maintenance-mode=true
```

Applying Maintenance Mode to Nodes

If your cluster consists of more than 3 nodes, you can easily set one node to maintenance mode, while the other nodes continue their normal operation. For example, to put the node alice into maintenance mode, use the **configure** command:

```
root # crm configure edit node alice attributes maintenance="true"
```

The node alice becomes “unmanaged” and other resources will not be allocated to any maintenance mode nodes.

Applying Maintenance Mode to Resources

If you need to set a specific resource into maintenance mode, use the **meta** command. For example, to put the resource ipaddress into maintenance mode, enter:

```
root # crm meta ipaddress set maintenance true
```



Warning: Risk of Data Loss

If you need to execute any testing or maintenance tasks while services are running under cluster control, make sure to follow this outline:

1. Before you start, set the individual resource, node or the whole cluster to maintenance mode. This helps to avoid unwanted side effects like resources not starting in an orderly fashion, the risk of unsynchronized CIBs across the cluster nodes or data loss.
2. Execute your maintenance task or tests.
3. After you have finished, remove the maintenance mode to start normal cluster operation.

For more details on what happens to the resources and the cluster while being in maintenance mode, see [Section 4.7, “Maintenance Mode”](#).

6.5.7 Getting Health Status

The “health” status of a cluster or node can be displayed with so called *scripts*. A script can perform different tasks, they are not targeted to health at all. However, for this subsection, we focus on how getting the health status.

To get all the details about the health command, use describe:

```
root # crm script describe health
```

It shows a description and a list of all parameters and their default values. To execute a script, use run:

```
root # crm script run health verbose=true
```

If you prefer to run only one step from the suite, the describe command shows a list of all available steps in the *Steps* category.

For example, the following command executes the first step of the health command. The output is stored in the health.json file for further investigation:

```
root # crm script run health \  
      step='Collect cluster information' \  
      statefile='health.json'
```

For additional information regarding scripts, see <http://crmsh.github.io/scripts/>.

6.6 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:

```
root # 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:

```
root # 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:

```
root # 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.

6.7 Retrieving History Information

Investigating the cluster history is a complex task. To simplify this task, crmsh 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 5.27, “Viewing Transitions with the History Explorer”*.

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:

```
root # 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 crm_report:

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

To limit crm_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 -l nograph
```

6.8 For More Information

- The `crm` man page.
- Visit the upstream project documentation at <http://crmsh.github.io/documentation> .
- See *Article “Highly Available NFS Storage with DRBD and Pacemaker”* for an exhaustive example.

7 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.

7.1 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: External SSH/STONITH Are Not Supported

It is impossible to know how SSH might react to other system problems. For this reason, external SSH/STONITH agents (like stonith:external/ssh) are not supported for production environments. If you still want to use such agents for testing, install the libglue-devel package.

To get a list of all currently available STONITH devices (from the software side), use the command **crm ra list stonith**. If you do not find your favorite agent, install the -devel package. 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.

7.2 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”](#) for more information. Each resource agent must supported the following operations to control it:

start

start or enable the resource

stop

stop or disable the resource

status

returns the status of the resource

monitor

similar to status, but checks also for unexpected states

validate

validate the resource's 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_Agents⁷. Find special information about several concepts at *Chapter 1, Product Overview*.

7.3 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 7.1: 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 re-tried 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 7.2: 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

OCF Return Code	OCF Alias	Description	Recovery Type
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	<p>The resource is not running. The cluster will not attempt to stop a resource that returns this for any action.</p> <p>This OCF return code may or may not require resource recovery—it depends on what is the expected resource status. If unexpected, then <u>soft</u> recovery.</p>	N/A
8	OCF_RUNNING_MASTER	The resource is running in Master mode.	soft
9	OCF_FAILED_MASTER	The resource is in Master mode but has failed. The resource will be demoted, stopped and then started (and possibly promoted) again.	soft
other	N/A	Custom error code.	soft

8 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 alice”. In an HA cluster, such a state implies that “resource r1 is stopped on all nodes except alice”, 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.

8.1 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.

8.2 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.

8.2.1 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.

8.2.2 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.

8.3 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”](#)). 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 5.3.3, “Creating STONITH Resources”](#), or [Section 6.4.3, “Creating a STONITH Resource”](#).

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
```

8.3.1 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:

```
root # crm < sample.txt
```

For more information about configuring resources with the **crm** command line tool, refer to *Chapter 6, Configuring and Managing Cluster Resources (Command Line)*.



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 8.1: TESTING CONFIGURATION

```
configure
primitive st-null stonith:null \
params hostlist="alice bob"
clone fencing st-null
commit
```

EXAMPLE 8.2: TESTING CONFIGURATION

An alternative configuration:

```
configure
primitive st-alice stonith:null \
params hostlist="alice"
primitive st-bob stonith:null \
params hostlist="bob"
location l-st-alice st-alice -inf: alice
location l-st-bob st-bob -inf: bob
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 8.3: 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="alice bob" \  
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 8.4: 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=alice ipaddr=192.168.0.101 \  
userid=USERID passwd=PASSWORD \  
primitive st-ibmrsa-2 stonith:external/ibmrsa-telnet \  
params nodename=bob ipaddr=192.168.0.102 \  
userid=USERID passwd=PASSWORD \  
location l-st-alice st-ibmrsa-1 -inf: alice \  
location l-st-bob st-ibmrsa-2 -inf: bob \  
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 8.5: 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.

8.3.2 Constraints Versus Clones

As explained in [Section 8.3.1, “Example STONITH Resource Configurations”](#), 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.

8.4 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 [Section 6.4.8, “Configuring Resource Monitoring”](#) for the command line approach.

8.5 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*. 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.

8.6 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”*.

- Use appropriate fencing devices for your setup. For details, also refer to [Section 8.5, “Special Fencing Devices”](#).
- Configure one or 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.

8.7 For More Information

</usr/share/doc/packages/cluster-glue>

In your installed system, this directory contains README files for many STONITH plugins 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 (Pacemaker 1.1 for Corosync 2.x and crmsh)*: 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.

9 Access Control Lists

The cluster administration tools like `crm shell (crmsh)` or `Hawk` can be used by `root` or any user in the group `haclient`. By default, these users have full read/write access. To limit access or assign more fine-grained access rights, you can make use of *Access control lists* (ACLs). Access control lists consist of an ordered set of access rules. Each rule allows read or write access or denies access to a part of the cluster configuration. Rules are typically combined to produce a specific role, then users may be assigned to a role that matches their tasks.

9.1 Requirements and Prerequisites

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

- Ensure you have the same users on all nodes in your cluster, either by using NIS, Active Directory, or by manually adding the same users to all nodes.
- All users for whom you want to modify access rights with ACLs must belong to the `haclient` group.
- All users need to run `crmsh` by its absolute path `/usr/sbin/crm`.
- If non-privileged users want to run `crmsh`, their `PATH` variable needs to be extended with `/usr/sbin`.

Important: Default Access Rights

- ACLs are an optional feature. By default, use of ACLs is disabled.
- If ACLs are not enabled, `root` and all users belonging to the `haclient` group have full read/write access to the cluster configuration.
- Even if ACLs are enabled and configured, both `root` and the default CRM owner `hacluster` *always* have full access to the cluster configuration.

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/>.

9.2 Enabling Use of ACLs In Your Cluster

Before you can start configuring ACLs, you need to *enable* use of ACLs. To do so, use the following command in the `crmsh`:

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

Alternatively, use Hawk as described in [Procedure 9.1, “Enabling Use of ACLs with Hawk”](#).

PROCEDURE 9.1: ENABLING USE OF ACLS WITH HAWK

1. Start a Web browser and log in to the cluster as described in [Section 5.1.1, “Starting Hawk and Logging In”](#).
2. In the left navigation bar, select *Cluster Properties*.
3. In the *CRM Configuration group*, , select the *enable-acl* attribute from the empty drop-down box and click the plus icon to add it.
4. To set enable-acl=true, active the checkbox next to enable-acl and confirm your changes.

9.3 The Basics of ACLs

Access control lists consist of an ordered set of access rules. Each rule allows read or write access or denies access to a part of the cluster configuration. Rules are typically combined to produce a specific role, then users may be assigned to a role that matches their tasks. An ACL role is a set of rules which describe access rights to CIB. A rule consist of the following:

- an access right like read, write, or deny
- 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 convenient to bundle ACLs into roles and assign a specific role to users. It is also possible to configure ACLs directly for individual users.

There are two methods to create ACL rules:

- *Section 9.3.1, “Setting ACL Rules via XPath Expressions”*. You need to know the structure of the underlying XML to create ACL rules.
- *Section 9.3.2, “Setting ACL Rules via Tag Abbreviations”*. Create a shorthand syntax and ACL rules to apply to the matched objects.

9.3.1 Setting ACL Rules via XPath Expressions

To manage ACL rules via XPath, you need to know the structure of the underlying XML. Retrieve the structure with the following command that shows your cluster configuration in XML (see *Example 9.1, “Excerpt of a Cluster Configuration in XML”*):

```
root # crm configure show xml
```

EXAMPLE 9.1: EXCERPT OF A CLUSTER CONFIGURATION IN XML

```
<num_updates="59"
  dc-uuid="175704363"
  crm_feature_set="3.0.9"
  validate-with="pacemaker-2.0"
  epoch="96"
  admin_epoch="0"
  cib-last-written="Fri Aug 8 13:47:28 2014"
  have-quorum="1">
<configuration>
  <crm_config>
    <cluster_property_set id="cib-bootstrap-options">
      <nvpair name="stonith-enabled" value="true" id="cib-bootstrap-options-
stonith-enabled"/>
      <nvpair name="no-quorum-policy" value="ignore" id="cib-bootstrap-options-no-
quorum-policy"/>
      [...]
    </cluster_property_set>
  </crm_config>
  <nodes>
    <node id="175704363" uname="alice"/>
```



```

    <node id="175704619" uname="bob"/>
  </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 expression `/cib/configuration/crm_config`.

As an example, [Table 9.1, “Operator Role—Access Types and XPath Expressions”](#) shows the parameters (access type and XPath expression) to create an “operator” role. Users with this role can only execute the tasks mentioned in the second column—they will not be able to reconfigure any resources (for example, change parameters or operations), nor change the configuration of colocation or ordering constraints.

TABLE 9.1: OPERATOR ROLE—ACCESS TYPES AND XPATH EXPRESSIONS

Type	XPath/Explanation
Write	<pre>//crm_config// nvpair[@name='maintenance-mode']</pre> <p>Turn maintenance mode on or off.</p>
Write	<pre>//op_defaults//nvpair[@name='record- pending']</pre> <p>Choose whether pending operations are recorded.</p>
Write	<pre>//nodes/node//nvpair[@name='standby']</pre> <p>Set node in online or standby mode.</p>
Write	<pre>//resources//nvpair[@name='target- role']</pre>

Type	XPath/Explanation
	Start, stop, promote or demote any resource.
Write	<pre>//resources//nvpair[@name='is-managed']</pre> <p>Select if a resource should be managed or not.</p>
Write	<pre>//constraints/rsc_location</pre> <p>Migrate/move resources from one node to another.</p>
Read	<pre>/cib</pre> <p>View the status of the cluster.</p>

9.3.2 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 abbreviated syntax can be used in `crmsh` and Hawk. The CIB daemon knows how to apply the ACL rules to the matching objects.

9.4 Configuring ACLs with Hawk

The following procedure shows how to configure a read-only access to the cluster configuration by defining a `monitor` role and assigning it to a user. Alternatively, you can use `crmsh` to do so, as described in *Procedure 9.3, “Adding a Monitor Role and Assigning a User with `crmsh`”*.

PROCEDURE 9.2: ADDING A MONITOR ROLE AND ASSIGNING A USER WITH HAWK

1. Start a Web browser and log in to the cluster as described in *Section 5.1.1, “Starting Hawk and Logging In”*.
2. In the left navigation bar, select *Access Control Lists*. The view shows the *Roles* and *Users* that are already defined.
3. To define your ACL role(s):
 - a. Select the *Roles* category and click the plus icon.
 - b. Enter a unique *Role ID*, for example, `monitor`.
 - c. For our example of defining a `monitor` role, select `read` from the *Right* drop-down box.
 - d. In the *Xpath* text box, enter the XPath expression `/cib` and click *Create Role*.
This creates a new role with name `monitor`, sets the `read` rights and applies it to all elements in the CIB by using the XPath expression `/cib`.
 - e. If necessary, you can add more access rights and XPath arguments by clicking the plus icon and specifying the respective parameters. Confirm your changes.
4. Assign the role to a user:
 - a. Select the *Users* category and click the plus icon.
The *Create User* view shows the available roles. It also contains an additional line for configuring individual ACLs rules for that user. The view lets you either assign one or multiple roles to the user *or* define one or more individual rules for the user. Selecting a role will make the line for individual rules disappear and vice versa. Assigning a role plus individual rules is not possible.

- b. Enter a unique *User ID*, for example, tux. Make sure this user belongs to the ha-client group.
 - c. To assign a role to the user, select the respective entries from *Roles*. In our example, select the monitor role you just created.
- To deselect one or multiple roles, click the respective entries once more. If no role is selected, the line for defining individual rules will appear again.

FIGURE 9.1: HAWK—ASSIGNING A ROLE OR RULE TO A USER

- d. If you want to define individual rules instead, select a *Right* and enter the respective Xpath parameters for your rule. Click the plus icon to define additional rules.
- e. Confirm your choice and assign the roles or rules to the user by clicking *Create User*.

To configure access rights for resources or constraints, you can also use the abbreviated syntax as explained in [Section 9.3.2, “Setting ACL Rules via Tag Abbreviations”](#). For example, if you have a primitive resource with the ID rsc1, enter the following values for *Tag* and *Ref*: primitive and rsc1. When entering only rsc1 as *Ref*, it 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.

9.5 Configuring ACLs with crmsh

The following procedure shows how to configure a read-only access to the cluster configuration by defining a monitor role and assigning it to a user.

PROCEDURE 9.3: ADDING A MONITOR ROLE AND ASSIGNING A USER WITH CRMSH

1. Log in as root.
2. Start the interactive mode of crmsh:

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

3. Define your ACL role(s):

- a. Use the role command to define a new role:

```
crm(live)configure# 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 expression /cib. If necessary, you can add more access rights and XPath arguments.

- b. Add additional roles as needed.

4. Assign your roles to one or multiple users. Make sure the users belong to the haclient group.

```
crm(live)configure# acl_target tux monitor
```

5. Check your changes:

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

6. Commit your changes:

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

To configure access rights for resources or constraints, you can also use the abbreviated syntax as explained in [Section 9.3.2, "Setting ACL Rules via Tag Abbreviations"](#). 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.

9.6 For More Information

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

10 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 Corosync, 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.

10.1 Configuring Bonding Devices with YaST

To configure a bonding device, you need to have multiple Ethernet devices that can be aggregated to a single bonding device. Proceed as follows:

1. Start YaST as root and select *Network Devices > Network Settings*.
2. In the *Network Settings*, switch to the *Overview* tab, which shows the available devices.
3. Check if the Ethernet devices to be aggregate to a bonding device have an IP address assigned. If yes, change it:
 - a. Select the device to change and click *Edit*.
 - b. In the *Address* tab of the *Network Card Setup* dialog that opens, select the option *No Link and IP Setup (Bonding Slaves)*.

Network Card Setup

General Address Hardware

Device Type: Ethernet Configuration Name: eth1

☒ No Link and IP Setup (Bonding Slaves) ☐ Use IBFT values

☐ Dynamic Address: DHCP DHCP both version 4 and 6

☐ Statically assigned IP Address

IP Address: Subnet Mask: 255.255.255.0 Hostname:

Additional Addresses

Alias Name	IP Address	Netmask
------------	------------	---------

Add Edit Delete

Help Cancel Back Next

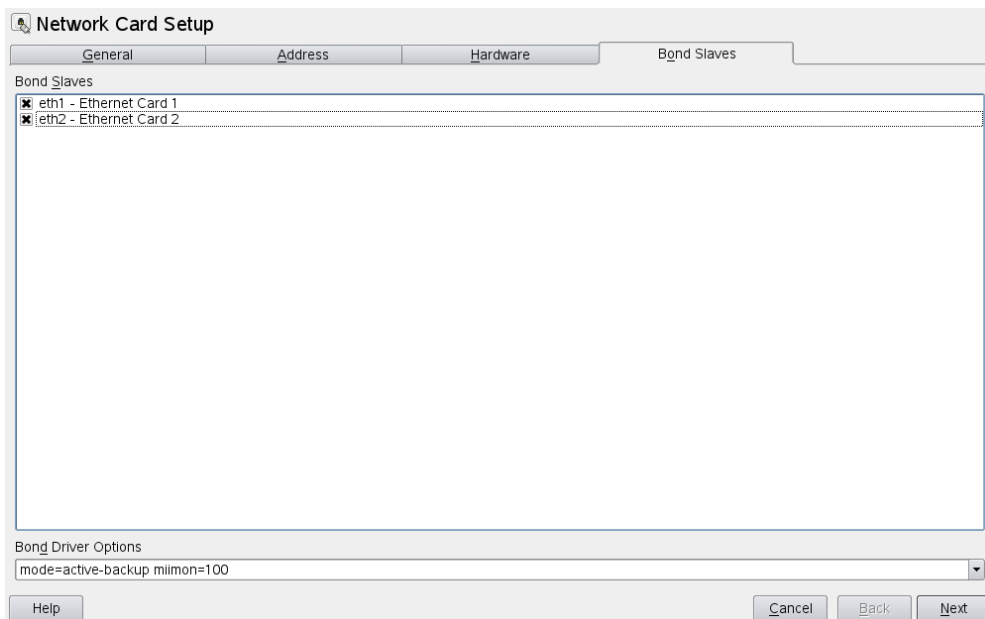
c. Click *Next* to return to the *Overview* tab in the *Network Settings* dialog.

4. To add a new bonding device:

- a. Click *Add* and set the *Device Type* to *Bond*. Proceed with *Next*.
- b. Select how to assign the IP address to the bonding device. Three methods are at your disposal:
 - No Link and IP Setup (Bonding Slaves)
 - Dynamic Address (with DHCP or Zeroconf)
 - Statically assigned IP Address

Use the method that is appropriate for your environment. If Corosync manages virtual IP addresses, select *Statically assigned IP Address* and assign an IP address to the interface.

- c. Switch to the *Bond Slaves* tab.
- d. It shows any Ethernet devices that have been configured as bonding slaves in [Step 3.b](#). To select the Ethernet devices that you want to include into the bond, activate the check box in front of the relevant *Bond Slave*.



- e. Edit the *Bond Driver Options*. The following modes are available:

balance-rr

Provides load balancing and fault tolerance, at the cost of out-of-order packet transmission. This may cause delays, for example, for TCP reassembly.

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.

- f. Make sure to add the parameter `miimon=100` to *Bond Driver Options*. Without this parameter, the link is not checked regularly, so the bonding driver might continue to loose packets on a faulty link.
5. Click *Next* and leave YaST with *OK* to finish the configuration of the bonding device. YaST writes the configuration to `/etc/sysconfig/network/ifcfg-bondDEVICENUMBER`.

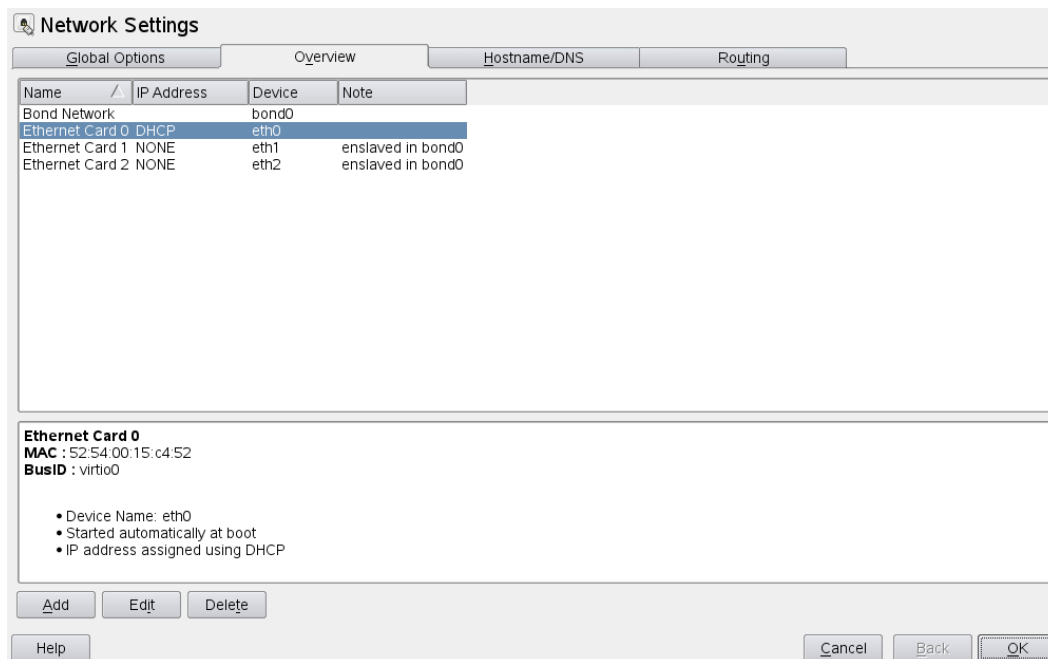
10.2 Hotplugging of Bonding Slaves

Sometimes it is necessary to replace a bonding slave interface with another one, for example, if the respective network device constantly fails. The solution is to set up hotplugging bonding slaves. It is also necessary to change the `udev` rules in order to match the device by bus ID instead of by MAC address. This enables you to replace defective hardware (a network card in the same slot but with a different MAC address), if the hardware allows for that.

PROCEDURE 10.1: CONFIGURING HOTPLUGGING OF BONDING SLAVES WITH YAST

If you prefer manual configuration instead, refer to the SUSE Linux Enterprise Server 12 Administration Guide, chapter *Basic Networking*, section *Hotplugging of Bonding Slaves*.

1. Start YaST as `root` and select *Network Devices* > *Network Settings*.
2. In the *Network Settings*, switch to the *Overview* tab, which shows the already configured devices. If bonding slaves are already configured, the *Note* column shows it.



3. For each of the Ethernet devices that have been aggregated to a bonding device, execute the following steps:
 - a. Select the device to change and click *Edit*. The *Network Card Setup* dialog opens.
 - b. Switch to the *General* tab and make sure that *Activate device* is set to On Hotplug.
 - c. Switch to the *Hardware* tab.
 - d. For the *Udev rules*, click *Change* and select the *BusID* option.
 - e. Click *OK* and *Next* to return to the *Overview* tab in the *Network Settings* dialog. If you click the Ethernet device entry now, the bottom pane shows the device's details, including the bus ID.
4. Click *OK* to confirm your changes and leave the network settings.

At boot time, the network setup does not wait for the hotplug slaves, but for the bond to become ready, which needs at least one available slave. When one of the slave interfaces is removed from the system (unbind from NIC driver, rmmod of the NIC driver or true PCI hotplug removal), the Kernel removes it from the bond automatically. When a new card is added to the system (replacement of the hardware in the slot), udev renames it by applying the bus-based persistent name rule and calls ifup for it. The ifup call automatically joins it into the bond.

10.3 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](#).

For High Availability setups, especially the following options described therein are important: [miimon](#) and [use_carrier](#).

11 Load Balancing

Load Balancing makes a cluster of servers appear as one large, fast server to outside clients. This apparent single server is called a *virtual server*. It consists of one or more load balancers dispatching incoming requests and several real servers running the actual services. With a load balancing setup of High Availability Extension, you can build highly scalable and highly available network services, such as Web, cache, mail, FTP, media and VoIP services.

High Availability Extension supports two technologies for load balancing: Linux Virtual Server (LVS) and HAProxy. The key difference is Linux Virtual Server operates at OSI layer 4 (Transport), configuring the network layer of kernel, while HAProxy operates at layer 7 (Application), running in user space. Thus Linux Virtual Server needs less resources and can handle higher loads, while HAProxy can inspect the traffic, do SSL termination and make dispatching decisions based on the content of the traffic.

This section gives you a conceptual overview of load balancing in combination with high availability, then briefly introduces to Linux Virtual Server and HAProxy and points to further reading.

11.1 Conceptual Overview

The real servers and the load balancers may be interconnected by either high-speed LAN or by geographically dispersed WAN. The load balancers dispatch requests to the different servers. They make parallel services of the cluster appear as one 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 service failures and reconfiguring the whole virtual server system appropriately, as usual.

There are several load balancing strategies. Here are some Layer 4 strategies, suitable for Linux Virtual Server:

- **Round Robin.** The simplest strategy is to direct each connection to a different address, taking turns. For example, a DNS server can have several entries for a given hostname. With DNS roundrobin, the DNS server will return all of them in a rotating order. Thus different clients will see different addresses.
- **Selecting the “best” server.** Although it has several drawbacks, but balancing could be implemented with an “the first server who responds” or “the least loaded server” approach.
- **Balance number of connections per server.** A load balancer between users and servers can divide the number of users across multiple servers.
- **Geolocation.** It's possible to direct clients to a server nearby.

Here are some Layer 7 strategies, suitable for HAProxy:

- **URI.** Inspect the http content and dispatch to a server most suitable for this specific URI.
- **URL parameter, RDP cookie.** Inspect the http content for a session parameter, possibly in post parameters, or the RDP (remote desktop protocol) session cookie, and dispatch to the server serving this session.

Although there is some overlap, HAProxy can be used in scenarios where LVS/ipvsadm are not adequate and vice versa:

- **SSL termination.** the front-end load-balancers can handle the SSL layer and thus the cloud nodes do not need to have access to the SSL keys, or could take advantage of SSL accelerators in the load balancers.
- **Application level.** HAProxy operates at the application level, allowing the load balancing decisions to be influenced by the content stream. This allows for persistence based on cookies and other such filters.

On the other hand, LVS/ipvsadm cannot be fully replaced by HAProxy:

- LVS supports “direct routing”, where the load balancer is only in the inbound stream, whereas the outbound traffic is routed to the clients directly. This allows for potentially much higher throughput in asymmetric environments.
- LVS supports stateful connection table replication (via conntrackd). This allows for load-balancer fail-over that is transparent to the client and server.

11.2 Configuring Load Balancing with Linux Virtual Server

The following sections give an overview of the main LVS components and concepts. Then we explain how to set up Linux Virtual Server on High Availability Extension.

11.2.1 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.

11.2.2 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.

The `ldirectord` uses the `ipvsadm` tool (package `ipvsadm`) to manipulate the virtual server table in the Linux Kernel.

11.2.3 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.

11.2.4 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.

11.2.5 Setting Up IP Load Balancing with YaST

You can configure Kernel-based IP load balancing with the YaST IP Load Balancing 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 11.1, "Simple ldirectord Configuration"*.



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 11.1: 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 *Failure 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 the system to send alerts in case the connection status to any real server changes, enter a valid e-mail address in *Email Alert*.

7. With *Email Alert Frequency*, define after how many seconds the e-mail alert should be repeated if any of the real servers remains inaccessible.
8. 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.
9. 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.
10. 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.
11. 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`.

FIGURE 11.1: YAST IP LOAD BALANCING—GLOBAL PARAMETERS

PROCEDURE 11.2: 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 IP Load Balancing 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 11.2.3, “Packet Forwarding”](#). 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](#), you also need to specify further parameters for authentication. Switch to the *Auth type* tab and enter the details like *Login*, *Password*, *Database*, or *Secret*. For more information, refer to the YaST help text or to the ldirectord man page.
10. Switch to the *Others* tab.
11. Select the *Scheduler* to be used for load balancing. For information on the available schedulers, refer to the ipvsadm(8) man page.
12. 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.

13. Define further parameters, if needed. Confirm your configuration with **OK**. YaST writes the configuration to `/etc/ha.d/ldirectord.cf`.

IPLB - Virtual Servers Configuration

Virtual Server
192.168.0.200:80

Real Servers
192.168.0.110:80 gate
192.168.0.210:80 gate

Check type | Auth type | Others

Check Type: negotiate | Check Port: | Service: http | Check Command: |
Http Method: | Request: test.html | Receive: |
Virtual Host: | Fallback: 127.0.0.1:80

Help | Cancel | OK

FIGURE 11.2: YAST IP LOAD BALANCING—VIRTUAL SERVICES

EXAMPLE 11.1: SIMPLE LDIRECTORD CONFIGURATION

The values shown in *Figure 11.1, “YaST IP Load Balancing—Global Parameters”* and *Figure 11.2, “YaST IP Load Balancing—Virtual Services”*, would lead to the following configuration, defined in `/etc/ha.d/ldirectord.cf`:

```
autoreload = yes ❶  
  checkinterval = 5 ❷  
  checktimeout = 3 ❸  
  quiescent = yes ❹  
  virtual = 192.168.0.200:80 ❺  
  checktype = negotiate ❻  
  fallback = 127.0.0.1:80 ❼  
  protocol = tcp ❽  
  real = 192.168.0.110:80 gate ❾  
  real = 192.168.0.120:80 gate ❾  
  receive = "still alive" ❿
```

```
request = "test.html" 11
scheduler = wlc 12
service = http 13
```

- 1 Defines that `ldirectord` should continuously check the configuration file for modification.
- 2 Interval in which `ldirectord` will connect to each of the real servers to check if they are still online.
- 3 Time in which the real server should have responded after the last check.
- 4 Defines not to remove failed real servers from the Kernel's LVS table, but to set their weight to `0` instead.
- 5 Virtual IP address (VIP) of the LVS. The LVS is available at port `80`.
- 6 Type of check that should be performed to test if the real servers are still alive.
- 7 Server onto which to redirect a Web service all real servers for this service are down.
- 8 Protocol to be used.
- 9 Two real servers defined, both available at port `80`. The packet forwarding method is `gate`, meaning that direct routing is used.
- 10 Regular expression that needs to be matched in the response string from the real server.
- 11 URI to the object that is requested on each real server during the check intervals.
- 12 Selected scheduler to be used for load balancing.
- 13 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 (2) and request `192.168.0.110:80/test.html` or `192.168.0.120:80/test.html` as specified in 9 and 11. If it does not receive the expected `still alive` string (10) from a real server within 3 seconds (3) of the last check, it will remove the real server from the available pool. However, because of the `quiescent=yes` setting (4), 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.

11.2.6 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 3.5.4, “Transferring the Configuration to All Nodes”](#).

11.3 Configuring Load Balancing with HAProxy

The following section gives an overview of the HAProxy and how to set up on High Availability. The load balancer distributes all requests to its backend servers. It is configured as active/passive, meaning if one master fails, the slave becomes the master. In such a scenario, the user will not notice any interrupt.

In this section, we will use the following setup:

- Two load balancers, with the IP addresses `alice` (IP: `192.168.1.100`) and `bob` (IP: `192.168.1.101`)
- A virtual, floating IP address `192.168.1.99`
- Our servers (usually for Web content) `www1.example.com` (IP: `192.168.1.200`) and `www2.example.com` (IP: `192.168.1.201`)

To configure HAProxy, use the following procedure:

1. Install the haproxy package.
2. Create the file /etc/haproxy/haproxy.cfg with the following contents::

```
global ❶
    maxconn 256
    daemon

defaults ❷
    log      global
    mode     http
    option   httplog
    option   dontlognull
    retries  3
    option   redispatch
    maxconn  2000
    timeout  5000
    clitimeout 50000
    srvtimeout 50000

listen my-cluster 192.168.1.99:80 ❸
    bind :80
    mode http
    stats enable
    stats auth someuser:somepassword
    balance leastconn
    cookie JSESSIONID prefix
    option httpclose
    option forwardfor
    option httpchk GET /robots.txt HTTP/1.0
    server webA 192.168.1.200:80 cookie A check
    server webB 192.168.1.201:80 cookie B check
```

- ❶ Section, which contains process-wide and OS-specific options.

maxconn

Maximum per-process number of concurrent connections.

daemon

Recommended mode, HAProxy runs in the background.

- 2 Section, which sets default parameters for all other sections following its declaration. Some important lines:

redispatch

Enable or disable session redistribution in case of connection failure

log

Enables logging of events and traffic.

mode http

Operates in HTTP mode (recommended mode for HAProxy). In this mode, a request will be analyzed before a connection to any server is performed. Request, which are not RFC-compliant, will be rejected.

option forwardfor

Adds the HTTP X-Forwarded-For header into the request. You need this option if you want to preserve the client's IP address.

timeout,

clitimeout,

srvtimeout

- 3 Section, which combines frontend and backend sections in one.

balance leastconn

Defines the load balancing algorithm, see <http://cbonte.github.io/haproxy-dconv/configuration-1.5.html#4-balance>.

stats enable,

stats auth

Enables statistics reporting (by stats enable). The auth option logs statistics with authentication to a specific account.

3. Test your configuration file:

```
root # haproxy -f /etc/haproxy/haproxy.cfg -c
```


4. Add the following line to Csync2's configuration file `/etc/csync2/csync2.cfg` to make sure, HAProxy configuration file is included:

```
include /etc/haproxy/haproxy.cfg
```

5. Synchronize it:

```
root # csync2 -f /etc/haproxy/haproxy.cfg
root # csync2 -xv
```



Note

The Csync2 configuration part assumes that the HA nodes were configured using **ha-cluster-bootstrap**, according to the Administration Guide.

6. Make sure HAProxy is disabled on both load balancers (alice and bob) as it is started by Pacemaker:

```
root # systemctl disable haproxy
```

7. Configure a new CIB:

```
root # crm configure
crm(live)# cib new haproxy-config
crm(haproxy-config)# primitive haproxy systemd:haproxy op monitor interval=10s
crm(haproxy-config)# primitive vip-ww1 IPAddr2 params ip=192.168.1.100
crm(haproxy-config)# primitive vip-ww2 IPAddr2 params ip=192.168.1.101
crm(haproxy-config)# group g-haproxy vip-ww1 vip-ww2 haproxy
```


8. Verify the new CIB and correct any errors:

```
crm(haproxy-config)# verify
```

9. Commit the new CIB:

```
crm(haproxy-config)# cib use live
crm(live)# cib commit haproxy-config
```

11.4 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.

12 GEO Clusters (Multi-Site Clusters)

Apart from local clusters and metro area clusters, SUSE® Linux Enterprise High Availability Extension 12 also supports geographically dispersed clusters (GEO clusters, sometimes also called 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 GEO clusters is available as a separate extension to GEO Clustering for SUSE Linux Enterprise High Availability Extension. For details on how to use and set up GEO clusters, refer to the Quick Start GEO Clustering for SUSE Linux Enterprise High Availability Extension. It is available from <http://www.suse.com/documentation/> or in your installed system under /usr/share/doc/manual/sle-ha-geo-manuals_en/.

III Storage and Data Replication

- 13 OCFS2 **215**
- 14 GFS2 **226**
- 15 DRBD **232**
- 16 Cluster Logical Volume Manager (cLVM) **246**
- 17 Storage Protection **260**
- 18 Samba Clustering **273**
- 19 Disaster Recovery with Rear (Relax-and-Recover) **283**

13 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 Corosync and the Distributed Lock Manager (DLM).

13.1 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.

13.2 OCFS2 Packages and Management Utilities

The OCFS2 Kernel module (`ocfs2`) is installed automatically in the High Availability Extension on SUSE® Linux Enterprise Server 12. 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 13.1: OCFS2 UTILITIES

OCFS2 Utility	Description
<code>debugfs.ocfs2</code>	Examines the state of the OCFS file system for the purpose of debugging.
<code>fsck.ocfs2</code>	Checks the file system for errors and optionally repairs errors.
<code>mkfs.ocfs2</code>	Creates an OCFS2 file system on a device, usually a partition on a shared physical or logical disk.

OCFS2 Utility	Description
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.

13.3 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, and a STONITH resource. OCFS2 uses the cluster membership services from Pacemaker which run in user space. Therefore, DLM needs to be configured as clone resource that is present on each node in the cluster.

The following procedure uses the `crm` shell to configure the cluster resources. Alternatively, you can also use Hawk to configure the resources as described in [Section 13.6, “Configuring OCFS2 Resources With Hawk”](#).



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 13.1: 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"*.
3. Run crm configure.
4. Configure external/sbd as fencing device with /dev/sdb2 being a dedicated partition on the shared storage for heartbeating and fencing:

```
crm(live)configure# primitive sbd_stonith stonith:external/sbd \  
    meta target-role="Started"
```
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 13.2: CONFIGURING A DLM RESOURCE

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 resource for DLM:

```
crm(live)configure# primitive dlm ocf:pacemaker:controld \  
    op monitor interval="60" timeout="60"
```

4. Create a base-group for the DLM resource. As further cloned primitives are created, it will be added to this group.

```
crm(live)configure# group base-group dlm
```

5. Clone the base-group so that it runs on all nodes.


```
crm(live)configure# clone base-clone base-group \  
meta interleave=true target-role=Started
```

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

13.4 Creating OCFS2 Volumes

After you have configured a DLM cluster resource as described in *Section 13.3, “Configuring OCFS2 Services and a STONITH Resource”*, 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 13.3, “Creating and Formatting an OCFS2 Volume”*. The most important parameters for the command are listed in *Table 13.2, “Important OCFS2 Parameters”*. For more information and the command syntax, refer to the **mkfs.ocfs2** man page.

TABLE 13.2: IMPORTANT OCFS2 PARAMETERS

OCFS2 Parameter	Description and Recommendation
Volume Label (<u>-L</u>)	A descriptive name for the volume to make it uniquely identifiable when it is mounted on different nodes. Use the <u>tuneefs.ocfs2</u> utility to modify the label as needed.

OCFS2 Parameter	Description and Recommendation
Cluster Size (<u>-C</u>)	<p>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 <u>mkfs.ocfs2</u> man page.</p>
Number of Node Slots (<u>-N</u>)	<p>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_64) and big-endian architectures (such as s390x).</p> <p>Node-specific files are referred to as local files. A node slot number is appended to the local file. For example: <u>journal:0000</u> belongs to whatever node is assigned to slot number <u>0</u>.</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 <u>tuneefs.ocfs2</u> utility to increase the number of node slots as needed. Note that the value cannot be decreased.</p> <p>In case the <u>-N</u> parameter is not specified, the number of slots is decided based on the size of the file system.</p>
Block Size (<u>-b</u>)	<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 <u>mkfs.ocfs2</u> man page.</p>

OCFS2 Parameter	Description and Recommendation
Specific Features On/Off (<code>--fs-features</code>)	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. For an overview of all available flags, refer to the <code>mkfs.ocfs2</code> man page.
Pre-Defined Features (<code>--fs-feature-level</code>)	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 13.3: 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 status`.
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, enter the following commands:

```
root # mkfs.ocfs2 -N 32 /dev/sdb1
```

13.5 Mounting OCFS2 Volumes

You can either mount an OCFS2 volume manually or with the cluster manager, as described in *Procedure 13.5, "Mounting an OCFS2 Volume with the Cluster Resource Manager"*.

PROCEDURE 13.4: 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 status.
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 cluster resources.

PROCEDURE 13.5: MOUNTING AN OCFS2 VOLUME WITH THE CLUSTER RESOURCE MANAGER

To mount an OCFS2 volume with the High Availability software, configure an ocfs2 file system resource in the cluster. The following procedure uses the crm shell to configure the cluster resources. Alternatively, you can also use Hawk to configure the resources as described in *Section 13.6, "Configuring OCFS2 Resources With Hawk"*.

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:

```
crm(live)configure# 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 ocfs2-1 and dlm primitive to the base-group you created in *Procedure 13.2, "Configuring a DLM Resource"*

```
crm(live)configure# group base-group dlm 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 dlm resource already running.

5. Review your changes with show.

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

13.6 Configuring OCFS2 Resources With Hawk

Instead of configuring the DLM and the file system resource for OCFS2 manually with the `crm` shell, you can also use the OCFS2 template in Hawk's *Setup Wizard*.

Important: Differences Between Manual Configuration and Hawk

The OCFS2 template in the *Setup Wizard* does *not* include the configuration a STONITH resource. If you use the wizard, you still need to create an SBD partition on the shared storage and configure a STONITH resource as described in *Procedure 13.1, "Configuring a STONITH Resource"*.

Using the OCFS2 template in the Hawk *Setup Wizard* also leads to a slightly different resource configuration than the manual configuration described in *Procedure 13.2, "Configuring a DLM Resource"* and *Procedure 13.5, "Mounting an OCFS2 Volume with the Cluster Resource Manager"*.

PROCEDURE 13.6: CONFIGURING OCFS2 RESOURCES WITH HAWK'S SETUP WIZARD

1. Start a Web browser and log in to the cluster as described in *Section 5.1.1, "Starting Hawk and Logging In"*.
2. In the left navigation bar, select *Setup Wizard*.
3. Select the `OCFS2 Filesystem` template and click *Next*.

Hawk proposes values for the following parameters:

- Resource ID
- Block Device
- File System Type

If you need information about an option, click it to display a short help text in Hawk.

4. Complete the information by entering the path to the *Block Device* for your file system and by entering additional *Mount Options*, if necessary.

The screenshot shows the 'Cluster Setup Wizard' window with the title 'OCFs2 Filesystem: Filesystem'. A note at the top states: 'If an OCFs2 filesystem does not already exist on the block device specified here, you will need to run mkfs to create it, prior to starting the filesystem resource. You will also need to create the mountpoint directory on all cluster nodes.' The configuration fields are: Resource ID: 'clusterfs', Block Device: '/dev/sbd1', Mount Point: '/srv/clusterfs', File system type: 'ocfs2', and Mount options: 'acl'. A yellow box highlights the 'Mount options' field with the text: 'Mount options: Any additional options to be given as -o options to the mount command.' Navigation buttons at the bottom are 'Cancel', '< Back', and 'Next >'. The footer shows 'Copyright © 2009-2014 SUSE, LLC' and 'UI Host: barett-2'.

FIGURE 13.1: HAWK SETUP WIZARD—OCFS2 TEMPLATE

5. Click *Next*.

The wizard displays the configuration snippet that will be applied to the CIB.

The screenshot shows the 'Cluster Setup Wizard' window with the title 'OCFs2 Filesystem: Confirm'. It states: 'The following configuration will be applied to the cluster:'. Below this is a code block containing the configuration snippet for the OCFs2 filesystem resource. Navigation buttons at the bottom are 'Cancel', '< Back', and 'Next >'. The footer shows 'Copyright © 2009-2014 SUSE, LLC' and 'UI Host: barett-2'.

```
primitive clusterfs ocf:heartbeat:Filesystem
  params
    device="/dev/sbd1"
    directory="/srv/clusterfs"
    fstype="ocfs2"
    options="acl"
  op start timeout="60" op stop timeout="60"
  op monitor interval="20" timeout="40"
primitive dlm ocf:pacemaker:controld
  op start timeout="90" op stop timeout="100"
  op monitor interval="60" timeout="60"
clone base-clone dlm meta interleave="true"
clone c-clusterfs clusterfs
  meta interleave="true" target-role="Stopped"
order base-then-clusterfs
inf: base-clone c-clusterfs
colocation clusterfs-with-base
inf: c-clusterfs base-clone
```

6. To apply it, click *Next*.

A message on the screen shows if the action has been successful. If everything is according to your wishes, leave the wizard.

13.7 Using Quotas on OCFS2 File Systems

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

13.8 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> ↗

The project's documentation home page.

14 GFS2

Global File System 2 or GFS2 is a shared disk file system for Linux computer clusters. GFS2 allows all nodes to have direct concurrent access to the same shared block storage. GFS2 has no disconnected operating-mode, and no client or server roles. All nodes in a GFS2 cluster function as peers. GFS2 supports up to 32 cluster nodes. Using GFS2 in a cluster requires hardware to allow access to the shared storage, and a lock manager to control access to the storage.

SUSE recommends OCFS2 over GFS2 for your cluster environments if performance is one of your major requirements. Our tests have revealed that OCFS2 performs better as compared to GFS2 in such settings.

14.1 GFS2 Packages and Management Utilities

To use GFS2, make sure `gfs2-utils` and matching `gfs2-kmp-*` package for your Kernel is installed on each node of the cluster.

The `gfs2-utils` package provides the following utilities for management of GFS2 volumes. For syntax information, see their man pages.

TABLE 14.1: GFS2 UTILITIES

GFS2 Utility	Description
<code>fsck.gfs2</code>	Checks the file system for errors and optionally repairs errors.
<code>gfs2_jadd</code>	Adds additional journals to a GFS2 file system.
<code>gfs2_grow</code>	Grow a GFS2 file system.
<code>mkfs.gfs2</code>	Create a GFS2 file system on a device, usually a shared device or partition.
<code>tunegfs2</code>	Allows viewing and manipulating the GFS2 file system parameters such as <code>UUID</code> , <code>label</code> , <code>lockproto</code> and <code>locktable</code> .

14.2 Configuring GFS2 Services and a STONITH Resource

Before you can create GFS2 volumes, you must configure DLM and a STONITH resource. GFS2 uses the cluster membership services from Corosync which run in user space. Therefore, DLM needs to be configured as clone resources that are present on each node in the cluster.

PROCEDURE 14.1: 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"*.
3. Run crm configure.
4. Configure external/sbd as fencing device with /dev/sdb2 being a dedicated partition on the shared storage for heartbeating and fencing:

```
crm(live)configure# primitive sbd_stonith stonith:external/sbd \  
    meta target-role="Started"
```

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 DLM 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 GFS2 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. Enter the following command to create the primitive resources for DLM:

```
root # crm configure primitive dlm ocf:pacemaker:controld \  
    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. If everything is correct, answer y to submit your changes.

3. Review your changes with show:

```
root # crm show
```

14.3 Creating GFS2 Volumes

After you have configured DLM as cluster resources as described in [Section 14.2, “Configuring GFS2 Services and a STONITH Resource”](#), configure your system to use GFS2 and create GFS2 volumes.



Note: GFS2 Volumes for Application and Data Files

We recommend that you generally store application files and data files on different GFS2 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 GFS2 volumes. Leave the devices as free space.

Then create and format the GFS2 volume with the **mkfs.gfs2** as described in [Procedure 14.3, “Creating and Formatting an GFS2 Volume”](#). The most important parameters for the command are listed in [Table 14.2, “Important GFS2 Parameters”](#). For more information and the command syntax, refer to the **mkfs.gfs2** man page.

TABLE 14.2: IMPORTANT GFS2 PARAMETERS

GFS2 Parameter	Description and Recommendation
Lock Protocol Name (<u>-p</u>)	The name of the locking protocol to use. Acceptable locking protocols are <code>lock_dlm</code> (for shared storage) or if you are using GFS2 as a local file system (1 node only), you can specify the <code>lock_nolock</code> protocol. If this option is not specified, <code>lock_dlm</code> protocol will be assumed.
Lock Table Name (<u>-t</u>)	The lock table field appropriate to the lock module you are using. It is <code>clustername:fsname</code> . <code>clustername</code> must match that in the cluster configuration file, <code>/etc/corosync/corosync.conf</code> . Only members of this cluster are permitted to use this file system. <code>fsname</code> is a unique file system name used to distinguish this GFS2 file system from others created (1 to 16 characters).
Number of Journals (<u>-j</u>)	The number of journals for <code>gfs2_mkfs</code> to create. You need at least one journal per machine that will mount the file system. If this option is not specified, one journal will be created.

PROCEDURE 14.3: CREATING AND FORMATTING AN GFS2 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 status`.
3. Create and format the volume using the `mkfs.gfs2` utility. For information about the syntax for this command, refer to the `mkfs.gfs2` man page.
For example, to create a new GFS2 file system on `/dev/sdb1` that supports up to 32 cluster nodes, use the following command:

```
root # mkfs.gfs2 -t hacluster:mygfs2 -p lock_dlm -j 32 /dev/sdb1
```

The `hacluster` name relates to the entry `cluster_name` in the file `/etc/corosync/corosync.conf` (this is the default).

14.4 Mounting GFS2 Volumes

You can either mount an GFS2 volume manually or with the cluster manager, as described in *Procedure 14.5, “Mounting an GFS2 Volume with the Cluster Manager”*.

PROCEDURE 14.4: MANUALLY MOUNTING AN GFS2 VOLUME

1. Open a terminal window and log in as root.
2. Check if the cluster is online with the command crm status.
3. Mount the volume from the command line, using the mount command.



Warning: Manually Mounted GFS2 Devices

If you mount the GFS2 file system manually for testing purposes, make sure to unmount it again before starting to use it by means of cluster resources.

PROCEDURE 14.5: MOUNTING AN GFS2 VOLUME WITH THE CLUSTER MANAGER

To mount an GFS2 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 Hawk 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 GFS2 file system on every node in the cluster:

```
crm(live)configure# primitive gfs2-1 ocf:heartbeat:Filesystem \  
    params device="/dev/sdb1" directory="/mnt/shared" fstype="gfs2" \  
    op monitor interval="20" timeout="40"
```

4. Create a base group that consists of the dlm primitive you created in *Procedure 14.2, “Configuring DLM Resources”* and the gfs2-1 primitive. Clone the group:

```
crm(live)configure# group base-group dlm gfs2-1  
    clone base-clone base-group \  
    meta interleave="true"
```

Due to the base group's internal colocation and ordering, Pacemaker will only start the gfs2-1 resource on nodes that also have an dlm resource already running.

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

15 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 Corosync, 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"*. To communicate with DRBD, users use the high-level command `drbdadm`. For maximum flexibility DRBD comes with the low-level tool `drbdsetup`.

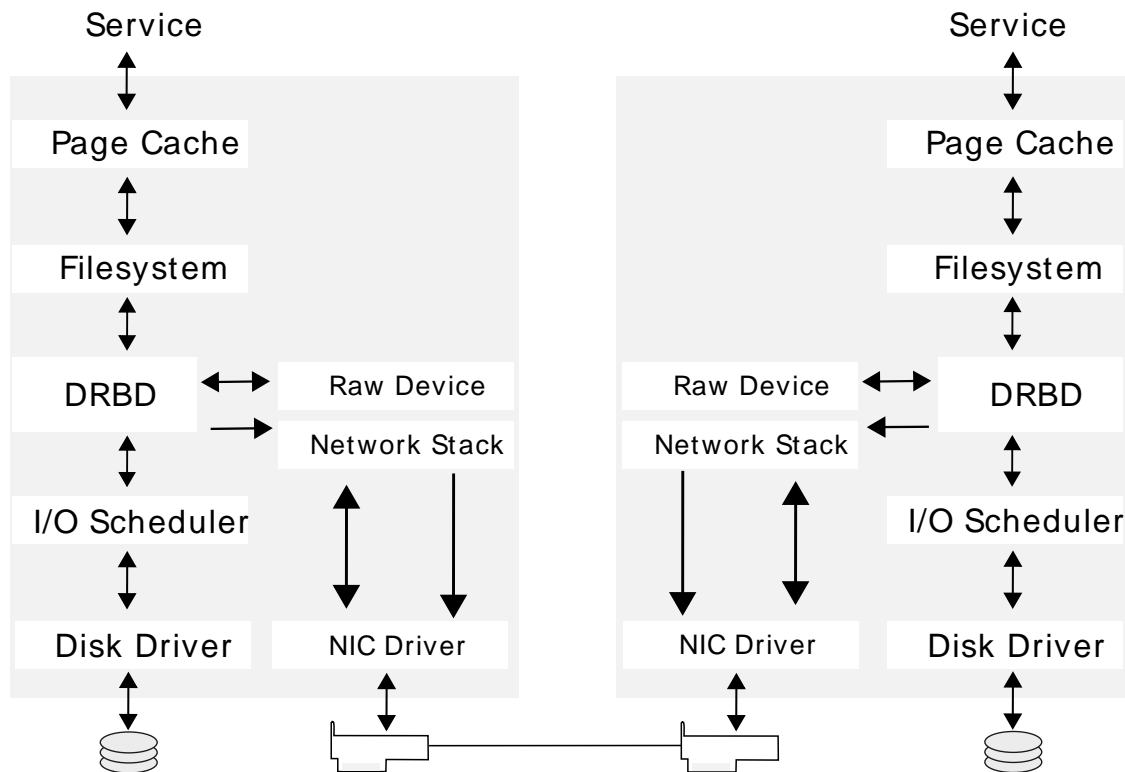


FIGURE 15.1: POSITION OF DRBD WITHIN LINUX

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 the used ports.

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_*N* device and not on the raw device, as DRBD uses the last part of the raw device for metadata. Using the raw device will cause inconsistent data.

With udev integration, you will also get symlinks in the form /dev/drbd/by-res/*RESOURCES* which are easier to use and provide safety against misremembering the devices' minor number.

For example, if the raw device is 1024 MB in size, the DRBD device has only 1023 MB available for data, with about 70 MB hidden and reserved for the metadata. Any attempt to access the remaining kilobytes via `/dev/drbdN` fails because it is not available for user data.

15.2 Installing DRBD Services

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"*. 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, install the package `drbd`.

To simplify the work with `drbdadm` (one part of the `drbd` package), use the Bash completion support. If you want to enable it in your current shell session, insert the following command:

```
root # source /etc/bash_completion.d/drbdadm.sh
```

To use it permanently for `root`, create, or extend 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 and `/dev/sda1` for storage. 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 that they should 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. Put your cluster in maintenance mode, if the cluster is already using DRBD:


```
root # crm configure edit node alice attributes maintenance="true"
```

If you skip this step when your cluster uses already DRBD, a syntax error in the live configuration will lead to a service shutdown.

2. Log in as user root.

3. Change DRBD's configuration files:

a. Open the file /etc/drbd.conf and insert the following lines, if they do not exist yet:

```
include "drbd.d/global_common.conf";  
include "drbd.d/*.res";
```

Beginning with DRBD 8.3 the configuration file is split into separate files, located under the directory /etc/drbd.d/.

b. 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 100;  
    degr-wfc-timeout 120;  
}
```

These options are used to reduce the timeouts when booting, see <http://www.drbd.org/users-guide-emb/re-drbdconf.html> for more details.

c. Create the file /etc/drbd.d/r0.res, change the lines according to your situation, and save it:

```
resource r0 { ❶  
    device /dev/drbd0; ❷  
    disk /dev/sda1; ❸  
    meta-disk internal; ❹  
    on jupiter { ❺  
        address 192.168.1.10:7788; ❻
```

```

}
on venus { ❸
    address 192.168.1.11:7788; ❹
}
syncer {
    rate 7M; ❺
}
}

```

- ❶ Name that allows some association to the service that needs them. For example, nfs, http, mysql_0, postgres_wal, etc.
 - ❷ The device name for DRBD and its minor number.
In the example above, the minor number 0 is used for DRBD. The udev integration scripts will give you a symlink /dev/drbd/by-res/nfs/0. Alternatively, omit the device node name in the configuration and use the following line instead:
device minor 0
 - ❸ The raw 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 host.
 - 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 tells which host this configuration statement applies to.
 - 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 the lower of the disk- and network bandwidth. It only limits the resynchronization, not the replication.
4. Check the syntax of your configuration file(s). If the following command returns an error, verify your files:

```
root # drbdadm dump all
```

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:

```
root # csync2 -xv /etc/drbd.d/
```

If you do not have Csync2 (or do not want to use it), copy the DRBD configuration files manually to the other node:

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

6. Initialize the meta data on *both* systems by entering the following on each node:

```
root # drbdadm create-md r0  
root # systemctl start drbd.service
```

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:

```
root # dd if=/dev/zero of=/dev/sda1 count=16 bs=1M
```

7. Watch the DRBD status by entering the following on each node:

```
root # systemctl status drbd.service
```

You should get something like this:

```
[... version string omitted ...]  
m:res  cs          ro          ds          p  mounted  
fstype  
0:r0    Connected  Secondary/Secondary  Inconsistent/Inconsistent  C
```

8. Start the resync process on your intended primary node (jupiter in this case):

```
root # drbdadm -- --overwrite-data-of-peer primary r0
```

9. Check the status again with `systemctl status drbd.service` and after resynchronization, 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.

10. Create your file system on top of your DRBD device, for example:

```
root # mkfs.ext3 /dev/drbd/by-res/r0/0
```

11. Mount the file system and use it:

```
root # mount /dev/drbd /mnt/
```

12. Reset the cluster's maintenance mode flag:

```
root # crm configure edit node alice attributes maintenance="false"
```

Alternatively, 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 already have a DRBD configuration, YaST warns you. YaST will change your configuration and will save your old DRBD configuration files as *.YaSTsave.
Leave the booting flag in *Start-up Configuration > Booting* as it is (by default it is off); do not change that as Pacemaker manage this service.
2. The actual configuration of the resource is done in *Resource Configuration* (see [Figure 15.2, "Resource Configuration"](#)).

FIGURE 15.2: 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 (mandatory)
<i>Name</i>	The hostname of the relevant node
<i>Address:Port</i>	The IP address and port number (default 7788) for the respective node
<i>Device</i>	The block device path that is used to access the replicated data.
<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 <u>internal</u> or specifies an explicit device extended by an index to hold the meta data needed by DRBD.</p> <p>A real device may also be used for multiple drbd resources. For example, if your <i>Meta-Disk</i> is <u>/dev/sda6[0]</u> for the first resource, you may use <u>/dev/sda6[1]</u> for the second resource.</p>

However, there must be at least 128 MB space for each resource available on this disk. The fixed metadata size limits the maximum data size that you are able to replicate.

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

3. 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:

```
root # csync2 -xv /etc/drbd.d/
```

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

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

4. Initialize and start the DRBD service on both systems by entering the following on each node:

```
root # drbdadm create-md r0  
root # systemctl start drbd.service
```

5. Configure `alice` as the primary node by entering the following on `alice`:

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

6. Check the DRBD service status by entering the following on each node:

```
root # systemctl status drbd.service
```

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

7. 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. It is recommended to use the `/dev/drbd/by-res/RESOURCE` 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.

- a. Open a terminal console, then log in as root.
- b. Create a mount point on jupiter, such as /srv/r0:

```
root # mkdir -p /srv/r0
```

- c. Mount the drbd device:

```
root # mount -o rw /dev/drbd0 /srv/r0
```

- d. Create a file from the primary node:

```
root # touch /srv/r0/from_jupiter
```

- e. Unmount the disk on jupiter:

```
root # umount /srv/r0
```

- f. Downgrade the DRBD service on jupiter by typing the following command on jupiter:

```
root # drbdadm secondary
```

2. Test the DRBD service on venus.

- a. Open a terminal console, then log in as root on venus.
- b. On venus, promote the DRBD service to primary:

```
root # drbdadm primary
```

- c. On venus, check to see if venus is primary:

```
root # systemctl status drbd.service
```

- d. On venus, create a mount point such as /srv/r0mount:

```
root # mkdir /srv/r0mount
```

- e. On venus, mount the DRBD device:

```
root # mount -o rw /dev/drbd_r0 /srv/r0mount
```

- f. Verify that the file you created on jupiter exists:

```
root # ls /srv/r0
```

The /srv/r0mount/from_jupiter 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.

- a. Dismount the disk on venus by typing the following command on venus:

```
root # umount /srv/r0
```

- b. Downgrade the DRBD service on venus by typing the following command on venus:

```
root # drbdadm secondary
```

- c. On jupiter, promote the DRBD service to primary:

```
root # drbdadm primary
```

- d. On jupiter, check to see if jupiter is primary:

```
root # systemctl status drbd.service
```

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 Pacemaker/Corosync. For information about installing and configuring for SUSE Linux Enterprise 12 see *Part II, "Configuration and Administration"*.

15.5 Tuning DRBD

There are several ways to tune DRBD:

1. Use an external disk for your metadata. This might help a small bit, at the cost of maintenance ease.
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. Tune your network connection, by changing the receive and send buffer settings via `sysctl`.
4. Change the `max-buffers`, `max-epoch-size` or both in the DRBD configuration.
5. Increase the `al-extents` value, depending on your IO patterns.
6. If you have a hardware RAID controller with a BBU (*Battery Backup Unit*), you might benefit from setting `no-disk-flushes`, `no-disk-barrier` and/or `no-md-flushes`.
7. Enable read-balancing depending on your workload. See <http://blogs.linbit.com/p/256/read-balancing/> for more details.

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:

```
root # 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.

```
root # 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), and compared to the hostname as stored in the Kernel (see the uname -n output).

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 can be found in the logging data:

```
Split-Brain detected, dropping connection!
```

To resolve this situation, enter the following on the node which has data to be discarded:

```
root # drbdadm secondary r0
root # drbdadm -- --discard-my-data connect r0
```

On the node which has the latest data enter the following:

```
root # drbdadm connect r0
```

That resolves the issue by overwriting one nodes' data with the peer's data, therefor getting a consistent view on both nodes.

15.7 For More Information

The following open source resources are available for DRBD:

- The project home page <http://www.drbd.org>.
- See Article *“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)**, **drbd-disk(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>
- Furthermore, for easier storage administration across your cluster, see the recent announcement about the *DRBD-Manager* at <http://blogs.linbit.com/p/666/drbd-manager>.

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 16.2.3, “Scenario: cLVM With iSCSI on SANs”](#).
- **DRBD.** This solution only provides RAID 0 (striping) and RAID 1 (mirroring). The last method is described in [Section 16.2.4, “Scenario: cLVM With DRBD”](#).

Although MD Devices (Linux Software RAID or **mdadm**) provides all RAID levels, it does not support clusters yet. Therefore it is not covered in the above list.

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 16.2.2, “Creating the Cluster Resources”](#) and then your LVM volumes. Otherwise it is impossible to remove the volumes later.

16.2.1 Configuring Cmirrord

To track mirror log information in a cluster, the `cmirrord` daemon is used. Cluster mirrors are not possible without this daemon running.

We assume that `/dev/sda` and `/dev/sdb` are the shared storage devices. Replace these with your own device name(s), if necessary. Proceed as follows:

1. Create a cluster with at least two nodes.
2. Configure your cluster to run **dlm**, **clvmd**, and STONITH:

```
root # crm configure
crm(live)configure# primitive clvmd ocf:lvm2:clvmd \
    op stop interval="0" timeout="100" \
    op start interval="0" timeout="90" \
    op monitor interval="20" timeout="20"
crm(live)configure# primitive dlm ocf:pacemaker:controld \
    op start interval="0" timeout="90" \
```

```

    op stop interval="0" timeout="100" \
    op monitor interval="60" timeout="60"
crm(live)configure# primitive sbd_stonith stonith:external/sbd
crm(live)configure# group base-group dlm clvmd
crm(live)configure# clone base-clone base-group \
    meta interleave="true"

```

3. Leave crmsh with **exit** and commit your changes.

4. Create a clustered volume group (VG):

```

root # pvcreeate /dev/sda /dev/sdb
root # vgcreate -cy vg /dev/sda /dev/sdb

```

5. Create a mirrored-log logical volume (LV) in your cluster:

```

root # lvcreate -nlv -m1 -l10%VG vg --mirrorlog mirrored

```

6. Use **lvs** to show the progress. If the percentage number has reached 100%, the mirrored disk is successfully synced.

7. To test the clustered volume /dev/vg/lv, use the following steps:

- a. Read or write to /dev/vg/lv.
- b. Deactivate your LV with **lvchange** **-an**.
- c. Activate your LV with **lvchange** **-ay**.
- d. Use **lvconvert** to convert a mirrored log to a disk log.

8. Create a mirrored-log LV in another cluster VG. This is a different volume group from the previous one.

The current cLVM can only handle one physical volume (PV) per mirror side. If one mirror is actually made up of several PVs that need to be concatenated or striped, **lvcreate** does not understand this. For this reason, **lvcreate** and **cmirrord** metadata needs to understand “grouping” of PVs into one side, effectively supporting RAID10.

In order to support RAID10 for **cmirrord**, use the following procedure (assuming that /dev/sda and /dev/sdb are the shared storage devices):

1. Create a volume group (VG):

```
root # pvcreate /dev/sda /dev/sdb
root # vgcreate vg /dev/sda /dev/sdb
```

2. Open the file `/etc/lvm/lvm.conf` and go to the section `allocation`. Set the following line and save the file:

```
mirror_legs_require_separate_pvs = 1
```

3. Add your tags to your PVs:

```
root # pvchange --addtag @a /dev/sda
root # pvchange --addtag @b /dev/sdb
```

A tag is an unordered keyword or term assigned to the metadata of a storage object. Tagging allows you to classify collections of LVM storage objects in ways that you find useful by attaching an unordered list of tags to their metadata.

4. List your tags:

```
root # pvs -o pv_name,vg_name,pv_tags /dev/sd{a,b}
```

You should receive this output:

PV	VG	PV Tags
/dev/sda	vgtest	a
/dev/sdb	vgtest	b

If you need further information regarding LVM, refer to our *Storage Administration Guide* at https://www.suse.com/documentation/sles11/stor_admin/data/lvm.html.

16.2.2 Creating the Cluster Resources

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

- *Creating a DLM Resource*
- *Creating LVM and cLVM 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 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"*. Otherwise, configure a DLM resource and a corresponding base group and base clone as described in *Procedure 13.2, "Configuring a DLM Resource"*.
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:

```
crm(live)configure# primitive clvm ocf:lvm2:clvmd \  
    params daemon_timeout="30"
```

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

```
crm(live)configure# primitive vg1 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*:


```
crm(live)configure# 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 13.2, "Configuring a DLM Resource"*:

- a. Enter

```
crm(live)configure# edit base-group
```

- b. In the vi editor that opens, modify the group as follows and save your changes:

```
crm(live)configure# group base-group dlm clvm vg1 ocfs2-1
```

Important: Setup Without OCFS2

If your setup does not include OCFS2, omit the ocfs2-1 primitive from the base group.

7. Review your changes with **show**.
8. If everything is correct, submit your changes with **commit** and leave the crm live configuration with **exit**.

16.2.3 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"*.

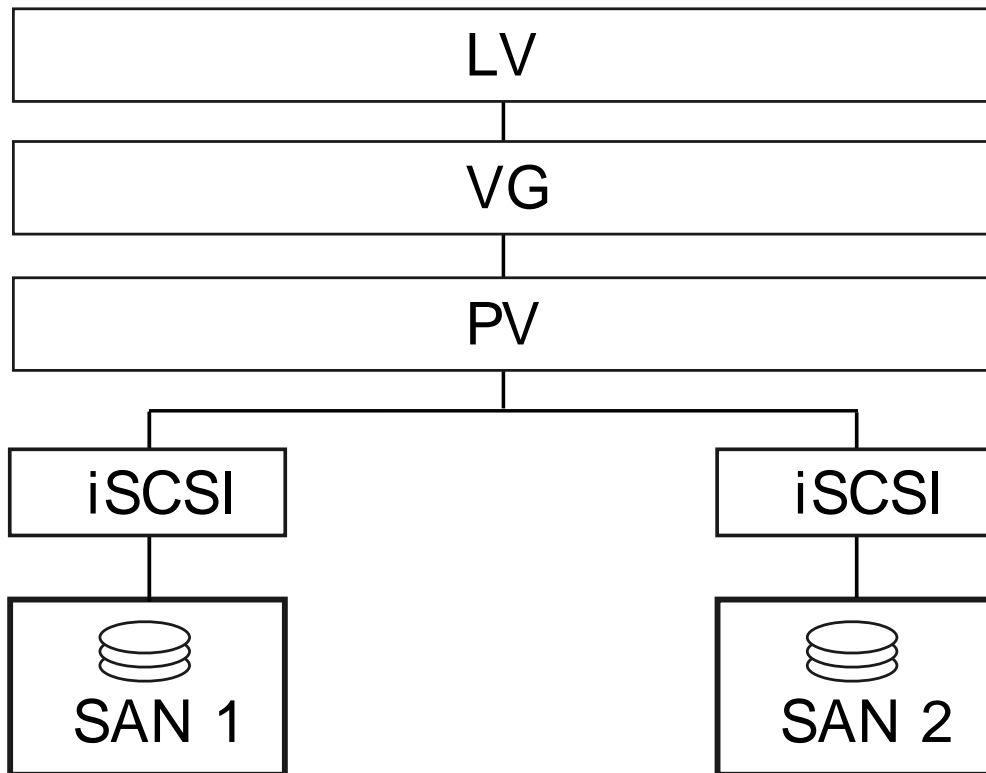


FIGURE 16.1: SETUP OF iSCSI WITH CLVM



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:

- a. Switch to the *Targets* tab.
- b. Click *Add*.
- c. 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>) .

- d. If you want a more descriptive name, you can change it as long as your identifier is unique for your different targets.
 - e. Click *Add*.
 - f. Enter the device name in *Path* and use a *Scsiid*.
 - g. 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)"*). 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:

```
root # 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:

```
root # 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 **ls SCSI**:

```
...
[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"*.
2. Prepare the physical volume for LVM with the command **pvcreate** on the disks /dev/sdd and /dev/sde:

```
root # pvcreate /dev/sdd
root # pvcreate /dev/sde
```

3. Create the cluster-aware volume group on both disks:

```
root # vgcreate --clustered y clustervg /dev/sdd /dev/sde
```

4. Create logical volumes as needed:

```
root # lvcreate --name clusterlv --size 500M clustervg
```

5. Check the physical volume with **pvdisk**:

```
--- 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           0uj3Xm-AI58-lxB1-mWm2-xn51-agM2-0UuHFC
```

6. Check the volume group with **vgdisplay**:

```
--- Volume group ---
VG Name           clustervg
System ID
Format            lvm2
Metadata Areas    2
```

Metadata Sequence No	1
VG Access	read/write
VG Status	resizable
Clustered	yes
Shared	no
MAX LV	0
Cur LV	0
Open LV	0
Max PV	0
Cur PV	2
Act PV	2
VG Size	1016,00 MB
PE Size	4,00 MB
Total PE	254
Alloc PE / Size	0 / 0
Free PE / Size	254 / 1016,00 MB
VG UUID	UCyWw8-2jqV-enuT-KH4d-NXQI-JhH3-J24anD

After you have created the volumes and started your resources you should have a new device named `/dev/dm-0`. It is recommended to use a clustered file system on top of your LVM resource, for example OCFS. For more information, see [Chapter 13, OCFS2](#)

16.2.4 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:
 - a. First, set up a DRBD device as primary/secondary as described in [Procedure 15.1, "Manually Configuring DRBD"](#). Make sure the disk state is up-to-date on both nodes. Check this with `cat /proc/drbd` or with `systemctl status drbd.service`.
 - b. 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;
    }
    ...
}
```

- c. Copy the changed configuration file to the other node, for example:

```
root # scp /etc/drbd.d/r0.res venus:/etc/drbd.d/
```

- d. Run the following commands on *both* nodes:

```
root # drbdadm disconnect r0
root # drbdadm connect r0
root # drbdadm primary r0
```

- e. Check the status of your nodes:

```
root # 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"](#) for detailed instructions. Before proceeding, confirm that these resources have started successfully on your cluster. You may use `crm_mon` or the Web interface 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:

```
root # pvcreate /dev/drbd_r0
```

4. Create a cluster-aware volume group:

```
root # 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:

```
root # 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>.

17 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 of the device is formatted for use with SBD. The size of the partition depends on the block size of the used disk (1 MB for standard SCSI disks with 512 Byte block size, DASD disks with 4 kB block size need 4 MB). 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.

If Pacemaker integration is activated, SBD will not self-fence if device majority is lost. For example, your cluster contains 3 nodes: A, B, and C. Due to a network split, A can only see itself while B and C can still communicate. In this case, there are two cluster partitions, one with quorum due to being the majority (B, C), and one without (A). If this happens while the majority of fencing devices are unreachable, node A would immediately commit suicide, but the nodes B and C would continue to run.

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 on 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*
2. *Setting Up the Software Watchdog*
3. *Starting the SBD Daemon*
4. *Testing SBD*
5. *Configuring the Fencing Resource*

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. Test your chosen timeouts in your specific environment. In case you use a multipath storage with just one SBD device, pay special attention to the failover delays incurred.



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:

```
root # 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:

```
root # 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):

```
root # /usr/sbin/sbd -d /dev/SBD -4 180 ① -1 90 ② 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:

```
root # 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

Watchdog will protect the system against SBD failures, if no other software uses it.

Important

No other software must access the watchdog timer. Some hardware vendors ship systems management software that use the watchdog for system resets (for example, HP ASR daemon). Disable such software, if watchdog is used by SBD.

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”](#), the SBD daemon will start automatically when the respective node is brought online with `systemctl start pacemaker.service`.

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/KERNEL_VERSION/drivers/watchdog. Alternatively, list the drivers that have been installed with your Kernel version with the following command:

```
root # rpm -ql kernel-VERSION | grep watchdog
```

As most watchdog driver names contain strings like wd, wdt, or dog, use the following command to check which driver is currently loaded:

```
root # lsmod | egrep "(wd|dog)"
```

To automatically load the watchdog driver, create the file /etc/modules-load.d/watchdog.conf containing a line with the driver name. For more information refer to the man page modules-load.d.

If you change the timeout for watchdog, the other two values (msgwait and stonith-timeout) must be changed as well. The watchdog timeout depends mostly on your storage latency. This value specifies that the majority of devices must be successfully finished their read operation within this time frame. If not, the node will self-fence.

The following “formula” expresses roughly this relationship between these three values:

EXAMPLE 17.1: CLUSTER TIMINGS WITH SBD AS STONITH DEVICE

```
Timeout (msgwait) = (Timeout (watchdog) * 2)
stonith-timeout = Timeout (msgwait) + 20%
```

For example, if you set the timeout watchdog to 120, you have to set the msgwait to 240 and the stonith-timeout to 288. You can check the output with cs_make_sbd_devices:

```
root # cs_make_sbd_devices --dump
==Dumping header on disk /dev/sdb
Header version      : 2.1
```

```
UUID          : 619127f4-0e06-434c-84a0-ea82036e144c
Number of slots : 255
Sector size    : 512
Timeout (watchdog) : 20
Timeout (allocate) : 2
Timeout (loop)   : 1
Timeout (msgwait) : 40
==Header on disk /dev/sdb is dumped
```

If you setup a new cluster, the **ha-cluster-init** command takes the above considerations into account.

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. Enable the SBD daemon to start at boot time with:

```
root # systemctl enable sbd.service
```

2. Run **ha-cluster-init**. This script ensure that SBD is correctly configured and the configuration file /etc/sysconfig/sbd is added to the list of files that needs to be synchronized with Csync2.

If you want to configure SBD manually, perform the following step:

To make the Corosync init script start and stop SBD, edit the file /etc/sysconfig/sbd and search for the following line, replacing SBD with your SBD device:

```
SBD_DEVICE="/dev/SBD"
```

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"
```

If the SBD device is not accessible, the daemon will fail to start and inhibit Corosync 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 Corosync and Pacemaker on boot.

3. Before proceeding, ensure that SBD has started on all nodes by executing `systemctl restart pacemaker.service`.

17.1.3.4 Testing SBD

1. The following command will dump the node slots and their current messages from the SBD device:

```
root # 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:

```
root # 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:

```

root # crm configure
crm(live)configure# property stonith-enabled="true"
crm(live)configure# property stonith-timeout="40s"
crm(live)configure# primitive stonith_sbd stonith:external/sbd \
    op start interval="0" timeout="15" start-delay="10"
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. The `msgwait` timeout should be longer than the maximum allowed timeout for the underlying IO system. For example, this is 30 seconds for plain SCSI disks. Provided you set the `msgwait` timeout value to 30 seconds, setting `stonith-timeout` to 40 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 Configuring a `sg_persist` Resource

1. Log in as `root` and start a shell.
2. Create the configuration file `/etc/sg_persist.conf`:

```

sg_persist_resource_MDRAID1() {
    devs="/dev/sdd /dev/sde"
    required_devs_nof=2
}

```

3. Run the following commands to create the primitive resources `sg_persist`:

```

root # crm configure

```

```
crm(live)configure# primitive sg ocf:heartbeat:sg_persist \
    params config_file=/etc/sg_persist.conf \
        sg_persist_resource=MDRAID1 \
        reservation_type=1 \
    op monitor interval=60 timeout=60
```

4. Add the sg_persist primitive to a master-slave group:

```
crm(live)configure# ms ms-sg sg \
    meta master-max=1 notify=true
```

5. Set the master on the alice server and the slave on the bob node:

```
crm(live)configure# location ms-sg-alice-loc ms-sg inf: alice
crm(live)configure# location ms-sg-bob-loc ms-sg 100: bob
```

6. Do some tests. When the resource is in master/slave status, on the master server, you can mount and write on /dev/sdc1, while on the slave server you cannot write.

In most cases you may want to use the above resource with the Filesystem resource, for example, OCFS2. In this case, you need to perform the following steps:

1. Add an OCFS2 primitive:

```
crm(live)configure# primitive ocfs2 ocf:heartbeat:Filesystem \
    params device="/dev/sdc1" directory="/mnt/ocfs2" fstype=ocfs2
```

2. Create a clone from a basegroup:

```
crm(live)configure# clone cl-group basegroup
```

3. Add relationship between ms-sg and cl-group:

```
crm(live)configure# colocation ocfs2-group-on-ms-sg inf: cl-group ms-sg:Master
crm(live)configure# order ms-sg-before-ocfs2-group inf: ms-sg:promote cl-group
```

4. Check all your changes with the edit command.
5. Commit your changes.

17.1.3.7 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:

```
root # sfex_init -n 1 /dev/sfex
```

3. Verify that the meta data has been created correctly:

```
root # sfex_stat -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:

```
crm(live)configure# 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:

```
crm(live)configure# order order-sfex-1 inf: sfex_1 filesystem1  
crm(live)configure# colocation colo-sfex-1 inf: filesystem1 sfex_1
```

3. If using group syntax, add the sfex resource as the first resource to the group:

```
crm(live)configure# group LAMP sfex_1 filesystem1 apache ipaddr
```

17.3 For More Information

See http://www.linux-ha.org/wiki/SBD_Fencing and **man sbd**.

18 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”*.



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.

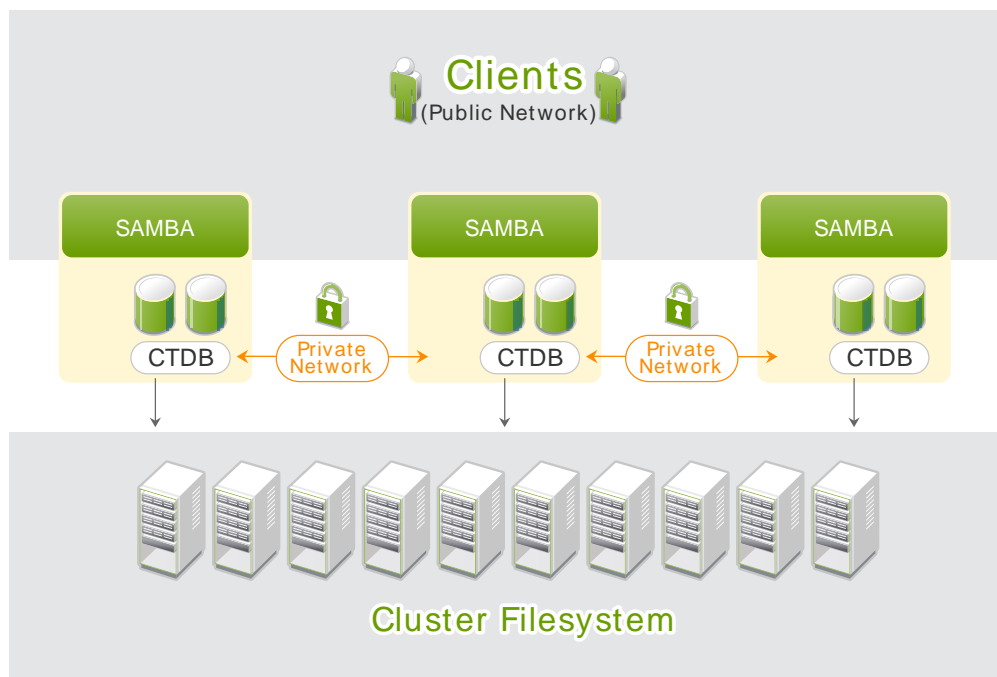


FIGURE 18.1: STRUCTURE OF A CTDB CLUSTER

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`. 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:

PROCEDURE 18.1: SETTING UP A BASIC CLUSTERED SAMBA SERVER

1. Prepare your cluster:

- a. Make sure, the following packages are installed before you proceed: `ctdb`, `tdbtools`, and `samba` (needed for `smb` and `nmb` resources).
- b. Configure your cluster (Pacemaker, OCFS2) as described in this guide in *Part II, "Configuration and Administration"*.
- c. Configure a shared file system, like OCFS2, and mount it, for example, on `/shared`.
- d. If you want to turn on POSIX ACLs, enable it:

- For a new OCFS2 file system use:

```
root # mkfs.ocfs2 --fs-features=xattr ...
```

- For an existing OCFS2 file system use:

```
root # tuneefs.ocfs2 --fs-feature=xattr DEVICE
```

Make sure the `acl` option is specified in the file system resource. Use the `crm` shell as follows:

```
crm(live)configure# primitive ocfs2-3 ocf:heartbeat:Filesystem params  
options="acl" ...
```

- e. Make sure the services `ctdb`, `smb`, and `nmb` are disabled:

```
root # systemctl disable ctdb.service  
root # systemctl disable smb.service  
root # systemctl disable nmb.service
```

- f. Open up port 4379 of your firewall on all nodes. This is needed for CTDB to communicate with other cluster nodes.

2. Create a directory for the CTDB lock on the shared file system:

```
root # 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. Configure Samba. Add the following lines in the `[global]` section of `/etc/samba/smb.conf`. Use the hostname of your choice in place of "CTDB-SERVER" (all nodes in the cluster will appear as one big node with this name, effectively):

```
[global]
# ...
# settings applicable for all CTDB deployments
netbios name = CTDB-SERVER
clustering = yes
idmap config * : backend = tdb2
passdb backend = tdbsam
ctdbd socket = /var/lib/ctdb/ctdb.socket
# settings necessary for CTDB on OCFS2
fileid:algorithm = fsid
vfs objects = fileid
# ...
```

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

```
root # csync2 -xv
```

For more information, see [Procedure 3.9, "Synchronizing the Configuration Files with Csync2"](#).

6. Add a CTDB resource to the cluster:

```
root # crm configure
crm(live)configure# primitive ctdb ocf:heartbeat:CTDB params \
    ctdb_manages_winbind="false" \
    ctdb_manages_samba="false" \
    ctdb_recovery_lock="/shared/samba/ctdb.lock" \
```

```

ctdb_socket="/var/lib/ctdb/ctdb.socket" \
  op monitor interval="10" timeout="20" \
  op start interval="0" timeout="90" \
  op stop interval="0" timeout="100"
crm(live)configure# primitive nmb systemd:nmb \
  op start timeout="60" interval="0" \
  op stop timeout="60" interval="0" \
  op monitor interval="60" timeout="60"
crm(live)configure# primitive smb systemd:smb \
  op start timeout="60" interval="0" \
  op stop timeout="60" interval="0" \
  op monitor interval="60" timeout="60"
crm(live)configure# group ctdb-group ctdb nmb smb
crm(live)configure# clone ctdb-clone ctdb-group meta interleave="true"
crm(live)configure# colocation ctdb-with-clusterfs inf: ctdb-clone fs-clone
crm(live)configure# order clusterfs-then-ctdb inf: fs-clone ctdb-clone
crm(live)configure# commit

```

7. Add a clustered IP address:

```

crm(live)configure# primitive ip ocf:heartbeat:IPaddr2 params ip=192.168.2.222
\
  unique_clone_address="true" \
  op monitor interval="60" \
  meta resource-stickiness="0"
crm(live)configure# clone ip-clone ip \
  meta interleave="true" clone-node-max="2" globally-unique="true"
crm(live)configure# colocation col-with-ctdb 0: ip-clone ctdb-clone
crm(live)configure# order o-with-ctdb 0: ip-clone ctdb-clone
crm(live)configure# commit

```

If `unique_clone_address` is set to `true`, the IPaddr2 resource agent adds a clone ID to the specified address, leading to three different IP addresses. These are usually not needed, but help with load balancing. For further information about this topic, see [Section 11.2, "Configuring Load Balancing with Linux Virtual Server"](#).

8. Commit your change:

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

9. Check the result:

```
root # crm status
Clone Set: base-clone [dlm]
    Started: [ factory-1 ]
    Stopped: [ factory-0 ]
Clone Set: fs-clone [clusterfs]
    Started: [ factory-1 ]
    Stopped: [ factory-0 ]
Clone Set: ctdb-clone [ctdb-group]
    Started: [ factory-1 ]
    Started: [ factory-0 ]
Clone Set: ip-clone [ip] (unique)
    ip:0      (ocf::heartbeat:IPaddr2):      Started factory-0
    ip:1      (ocf::heartbeat:IPaddr2):      Started factory-1
```

10. 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:

```
root # smbclient //192.168.2.222/myshare
```

18.3 Joining an Active Directory Domain

Active Directory (AD) is a directory service for Windows server systems.

The following instructions outline how to join a CTDB cluster to an Active Directory domain:

1. Create a CTDB resource as described in *Procedure 18.1, "Setting Up a Basic Clustered Samba Server"*.
2. Install the `samba-winbind` package.
3. Disable the `winbind` service:

```
root # systemctl disable winbind.service
```

4. Define a winbind cluster resource:

```
root # crm configure
crm(live)configure# primitive winbind systemd:winbind \
    op start timeout="60" interval="0" \
    op stop timeout="60" interval="0" \
    op monitor interval="60" timeout="60"
crm(live)configure# commit
```

5. Edit the `ctdb-group` and insert `winbind` between the `nmb` and `smb` resources:

```
crm(live)configure# edit ctdb-group
```

Save and close the editor with `:w` (`vim`).

6. Consult your Windows Server documentation for instructions on how to set up 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

7. *Procedure 18.2, "Joining Active Directory"*

Finally, join your cluster to the Active Directory server:

PROCEDURE 18.2: JOINING ACTIVE DIRECTORY

1. Make sure the following files are included in Csync2's configuration to become installed on all cluster hosts:

```
/etc/samba/smb.conf
/etc/security/pam_winbind.conf
/etc/krb5.conf
/etc/nsswitch.conf
/etc/security/pam_mount.conf.xml
/etc/pam.d/common-session
```

You can also use YaST's *Configure Csync2* module for this task, see [Section 3.5.4, "Transferring the Configuration to All Nodes"](#).

2. Run YaST and open the *Windows Domain Membership* module from the *Network Services* entry.
3. Enter your domain or workgroup settings and finish with *Ok*.

18.4 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 18.3: 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 ABSPATH/data.txt is available in your shared storage and is therefore accessible on all nodes (ABSPATH indicates an absolute path):

```
ping_pong ABSPATH/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 13, OCFS2*.

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 ↗

19 Disaster Recovery with Rear (Relax-and-Recover)

Relax-and-Recover (formerly “ReaR”, in this chapter abbreviated as Rear) 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 media. 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 disaster recovery media, to *always* test the disaster recovery with an *identical* test machine. Only if this procedure works satisfactorily is your disaster recovery system correctly and reliably set up.

19.1 Conceptual Overview

Rear supports several backup tools (Tivoli Storage Manager, QNetix Galaxy, Symantec NetBackup, EMC NetWorker, HP DataProtector) and can output its recovery medium to a CD or PXE environment. The restore step is possible through NFS, CIFS, or other network file systems.

To use Rear you need at least two identical systems: the main machine where your production 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.

SUSE Linux Enterprise Server ships a disaster recovery system in the `rearVERSION` package.

Rear is used in the following way:

1. **Preparation.** Make a bootable CD, backup system.
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 recovery medium and restore your system from your backup.

To prepare the rescue media, the following steps are performed by Rear:

PREPARATION OF RECOVERY MEDIA BY REAR

1. Gather system information from the target system.
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 recovery medium with system configuration.

When a disaster occurs, the recovery process consists of these actions:

RECOVERY PROCESS

1. Boot from the recovery medium.
2. Restore the disk layout (partitions, RAID configurations, LVM, file systems).
3. Restore system and user data.
4. Restore boot loader.
5. Reboot system.



Note: Rear with Subvolumes and Snapshots in btrfs Filesystems

Due to technical and conceptual issues, Rear and the btrfs tools has some limitations you should be aware of:

- **btrfs with subvolumes, but no snapshots.** at least Rear version 1.15 is required as this version supports restoring the btrfs subvolume structure. The backup and restore of the data should work.
- **btrfs with subvolumes that contain snapshots.** The usual backup and restore cannot work for the data in the snapshot. Ordinary backup tools (for example, **tar**) cannot distinguish between data and btrfs snapshot structure. One workaround is to exclude the data in the btrfs snapshot and store this data and the normal data separately.

19.2 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.

19.3 Setting Up Rear

To configure Rear, add your options into the configuration file `/etc/rear/local.conf`. The former file `/etc/rear/sites.conf` has been removed from the package. If you have such a file from your last setup, Rear will still use it.

Define your output method by using the `OUTPUT` variable. For example, to output your recovery medium as ISO image, use:

```
OUTPUT=ISO
```

After you have changed your configuration file, make sure everything is correct by running the following command:

```
rear dump
```

Examine the output and look for any errors.

Find more information in the man page of Rear (`man rear`).

19.4 Storing your Backup on a NFS Server

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/doc/sles11/book_sle_admin/data/cha_nfs.html.
2. Make sure you NFS server has the appropriate configuration in the `/etc/exports` file:

```
/srv/nfs    *(fsid=0,crossmnt,rw,no_root_squash,sync,no_subtree_check)
```

3. Adapt the configuration file `/etc/rear/local.conf`. Replace the `NETFS_URL` with your own values.

If your NFS host is not an IP address but a hostname, DNS must work when the backup is restored. 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 on a NFS server:
BACKUP=NETFS

# BACKUP_OPTIONS variable contains the NFS mount options and
# with 'mount -o nolock' no rpc.statd (plus rpcbind) are needed:
BACKUP_OPTIONS="nfsvers=3,nolock"

# If the NFS server is not an IP address but a hostname,
# DNS must work in the rear recovery system when the backup is restored.
BACKUP_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

# Files in btrfs subvolumes are excluded by 'tar --one-file-system'
```

```
# so that such files must be explicitly included to be in the backup.
# Files in the following SLE12 default btrfs subvolumes are
# in the below example not included to be in the backup
#  /.snapshots/* /var/tmp/* /tmp/* /var/crash/*
# but files in /home/* are included to be in the backup
# (one line!):
BACKUP_PROG_INCLUDE=( '/home/*' '/var/spool/*' '/var/opt/*' '/var/log/*'
    '/var/lib/pgsql/*' '/var/lib/mailman/*' '/var/lib/named/*' '/usr/local/*'
    '/srv/*' '/boot/grub2/x86_64-efi/*' '/opt/*' '/boot/grub2/i386-pc/*' )
# Avoid that "rear recover" is 'Creating btrfs-filesystem' by default
# also for every mounted btrfs subvolume by excluding the mountpoints
# of the mounted btrfs subvolumes from component recreation
# see /usr/share/doc/packages/rear/user-guide/06-layout-configuration.txt
# and /usr/share/rear/conf/default.conf
# When /home is a separated filesystem remove "fs:/home" from the list below
# (one line!):
EXCLUDE_RECREATE=( "${EXCLUDE_RECREATE[@]}" "fs:/home" "fs:/snapshots"
    "fs:/var/tmp" "fs:/var/spool" "fs:/var/opt" "fs:/var/log"
    "fs:/var/lib/pgsql" "fs:/var/lib/mailman" "fs:/var/lib/named"
    "fs:/usr/local" "fs:/tmp" "fs:/srv" "fs:/var/crash"
    "fs:/boot/grub2/x86_64-efi" "fs:/opt" "fs:/boot/grub2/i386-pc" )
# This option defines a root password to allow SSH connection
# without a public/private key pair
#SSH_ROOT_PASSWORD="password_on_the_rear_recovery_system"
```

4. Prepare the backup by running:

```
rear -v 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 /var/lib/rear/output/rear-*HOSTNAME*.iso and burn it on CD.
2. Boot your test machine with the recovery CD.
3. Select *Recover NAME* from the menu selection and press Enter.

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 recovery CD, in case the media is damaged.

To use EMC Networker as backup tool, use the following configuration file (change the BACKUP_URL accordingly):

```
BACKUP=NSR
OUTPUT=ISO
BACKUP_URL=nfs://dd990.example.com/data/coll/rear/backup
OUTPUT_URL=nfs://dd990.example.com/data/coll/rear/output
NSRSERVER=backupserver.example.com
RETENTION_TIME="Month"
```

19.5 Using the YaST Rear Module

The YaST Rear module can be used to start with a basic setup. Rear saves the recovery image and data on an NFS backend or a USB stick. However, the YaST Rear module does not support recovery from a disaster. This requires some expertise and has to be done manually by the administrator.

To start with a basic setup, proceed as follows:

1. Decide how to start your recovery system. Choose *USB* if you want to boot from a USB stick or *ISO* for CD-ROM respectively.
2. Decide where the backup should be stored. Either select *NFS* or *USB*:
 - Select *NFS* if you have to use a server that offers Network File System. Specify the location as nfs://hostname/directory.
 - Select *USB* if you want to store your data on a USB stick. If no USB devices are shown, attach a USB stick or a USB disk and click *Rescan USB Devices*.
3. If you want a previous backup copy to be saved, check *Keep old backup*.

4. If you want to add additional directories that should also be included in your backup, use the *Advanced* menu and select *Additional Directories in Backup*.
5. In case your rescue system does not boot due to missing Kernel modules, use the *Advanced* menu, select *Additional Kernel Modules in Rescue System*, and add the respective Kernel modules into the list of added modules for your rescue system.
6. Click the *Save and run rear now* button to start the backup process.

After the YaST Rear module is finished with the backup, test your backup to make sure it works as expected.

19.6 For More Information

- http://en.opensuse.org/SDB:Disaster_Recovery ↗
- Manpage for **rear**
- </usr/share/doc/packages/rear/README>

IV Appendix

- A Troubleshooting **291**
- B Naming Conventions **300**
- C Cluster Management Tools **301**
- D Upgrading Your Cluster and Updating Software Packages **303**
- E Documentation Updates **308**

A 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.

A.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 12 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"*.

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)"*.

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 or the bootstrap scripts as described in *Section 3.5, "Manual Cluster Setup (YaST)"* and *Section 3.4, "Automatic Cluster Setup (ha-cluster-bootstrap)"*, 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.

Are Pacemaker and Corosync started on each cluster node?

Usually, starting Pacemaker also starts the Corosync service. To check if both services are running:

```
root # systemctl status pacemaker.service corosync.service
```

In case they are not running, start them by executing the following command:

```
root # systemctl start pacemaker.service
```

A.2 Logging

Where to find the log files?

For the Pacemaker logs, see the settings configured in the logging section of /etc/corosync/corosync.conf. In case the log file specified there should be ignored by Pacemaker, check the logging settings in /etc/sysconfig/pacemaker, Pacemaker's own configuration file. In case PCMK_logfile is configured there, Pacemaker will use the path that is defined by this parameter.

If you need a cluster-wide report showing all relevant log files, see [How can I create a report with an analysis of all my cluster nodes?](#) for more information.

I enabled monitoring but there is no trace of monitoring operations in the logs?

The lrmd 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 the logging data (sudo journalctl -n) 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`):

```
root # crm_mon -o -r
```

The display is refreshed when the status changes (to cancel this press `Ctrl-C`.) An example may look like:

EXAMPLE A.1: STOPPED RESOURCES

```
Last updated: Fri Aug 15 10:42:08 2014
Last change: Fri Aug 15 10:32:19 2014
Stack: corosync
Current DC: bob (175704619) - partition with quorum
Version: 1.1.12-ad083a8
2 Nodes configured
3 Resources configured

Online: [ alice bob ]

Full list of resources:

my_ipaddress      (ocf::heartbeat:Dummy): Started barett-2
my_filesystem     (ocf::heartbeat:Dummy): Stopped
my_webserver      (ocf::heartbeat:Dummy): Stopped

Operations:
* Node bob:
  my_ipaddress: migration-threshold=3
    + (14) start: rc=0 (ok)
    + (15) monitor: interval=10000ms rc=0 (ok)
* Node alice:
```

The *Pacemaker Explained (Pacemaker 1.1 for Corosync 2.x and crmsh)* PDF, available at <http://www.clusterlabs.org/doc/>, covers three different recovery types in the *How are OCF Return Codes Interpreted?* section.

A.3 Resources

How can I clean up my resources?

Use the following commands :

```
root # crm resource list
crm resource cleanup rscid [node]
```

If you leave out the node, the resource is cleaned on all nodes. More information can be found in [Section 6.5.2, "Cleaning Up Resources"](#).

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:

```
root # 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 waiting 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-alice rsc 100: alice
```

A.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.

A.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 the log messages (`sudo journalctl -n`) for the following line:

```
Jan 12 09:58:55 alice lrmd: [3487]: info: RA output: [...]
ERROR: Could not load ocfs2_stackglue
Jan 12 16:04:22 alice 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?

On the crm shell, use either `crm_report` or `hb_report` to create a report. The tools are 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 `crm_report` with the following command:

```
root # crm_report -f 0:00 -n jupiter -n venus
```

The command extracts all information since 0am on the hosts jupiter and venus and creates a tar.bz2 archive named crm_report-DATE.tar.bz2 in the current directory, for example, crm_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 crm_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 a Pacemaker 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). In case you have a non-standard SSH port, use the -X option to add the port (for example, with the port 3479, use -X "-p 3479"). Further options can be found in the man page of crm_report.

After crm_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.

A.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> .

B Naming Conventions

This guide uses the following naming conventions for cluster nodes and names, cluster resources, and constraints.

Cluster Nodes

Cluster nodes use first names:

alice, bob, charly, doro, and eris

Cluster Names

Clusters are named after cities:

amsterdam, berlin, canberra, dublin, fukuoka, gizeh, hanoi, and istanbul

Cluster Resources

Primitives	No prefix
Groups	Prefix <u>g-</u>
Clones	Prefix <u>cl-</u>
Multi-state Resources	Prefix <u>ms-</u>
Tags (for grouping resources)	Prefix <u>tag-</u>

Constraints

Ordering constraints	Prefix <u>o-</u>
Location constraints	Prefix <u>loc-</u>
Colocation constraints	Prefix <u>col-</u>

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 [Appendix A, Troubleshooting](#).



Note: Use crmsh

This tool is for experts only. In most cases the crm shell (crmsh) 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 and Updating Software Packages

This chapter covers two different scenarios: upgrading a cluster to another version of SUSE Linux Enterprise High Availability Extension (either a major release or a service pack) as opposed to updating individual packages on cluster nodes.

D.1 Terminology

In the following, find definitions of the most important terms used in this chapter:

Major Release,

General Availability (GA) Version

The Major Release of SUSE Linux Enterprise (or any software product) is a new version which brings new features and tools, decommissions previously deprecated components and comes with backwards incompatible changes.

Service Pack (SP)

Combines several patches into a form which is easy to install or deploy. Service packs are numbered and usually contain security fixes, updates, upgrades, or enhancements of programs.


Update

Installation of a newer *minor* version of a package.


Upgrade

Installation of a newer *major* version of a package or distribution, which brings *new features*.

D.2 Upgrading your Cluster to the Latest Product Version

Which upgrade path is supported and how to perform the upgrade depends on the current product version your cluster is running on and on the target version you want to migrate to. For general information on this, see the *SUSE Linux Enterprise Server 12 Deployment Guide*, chapter *Updating SUSE Linux Enterprise*. It is available at <http://www.suse.com/documentation/> .

D.2.1 Upgrading from SLE HA 11 SP3 to SLE HA 12

In order to successfully upgrade to SUSE Linux Enterprise High Availability Extension 12, your cluster needs to run the latest versions of SUSE Linux Enterprise Server and SUSE Linux Enterprise High Availability Extension (11 SP3). If your cluster is still based on an older product version, upgrade it to SUSE Linux Enterprise Server and SUSE Linux Enterprise High Availability Extension 11 SP3 first. Find information on this in the *High Availability Guide* for SUSE Linux Enterprise High Availability Extension 11, chapter *Upgrading Your Cluster to the Latest Product Version*. It is available at <http://www.suse.com/documentation/> .

Due to major changes in various components of the High Availability Extension 12 (for example, `/etc/corosync/corosync.conf`, disk formats of OCFS2), performing a rolling upgrade is not supported for this scenario. All cluster nodes must be offline and the cluster needs to be migrated as a whole as described in *Procedure D.1, “Upgrading the Cluster to SLE HA 12”*. Mixed clusters running on SUSE Linux Enterprise High Availability Extension 11/SUSE Linux Enterprise High Availability Extension 12 are not supported.

PROCEDURE D.1: UPGRADING THE CLUSTER TO SLE HA 12


Important: Required Preparations Before Upgrading

- Ensure that your system back-up is up to date and restorable.
- Test the upgrade procedure on a staging instance of your cluster setup first, before performing it in a production environment.
This gives you an estimation of the time frame required for the maintenance window. It also helps to detect and solve any unexpected problems that might arise.

Execute the following steps for each cluster node:

1. Log in to each cluster node and stop the cluster stack with:

```
root # rcopenais stop
```

2. For each cluster node, perform an upgrade from SUSE Linux Enterprise Server 11 SP3 to SUSE Linux Enterprise Server 12 and from SUSE Linux Enterprise High Availability Extension 11 SP3 to SUSE Linux Enterprise High Availability Extension 12. If you want to make use of GEO clustering, install the respective add-on as described in the *GEO Clustering for SUSE Linux Enterprise High Availability Extension Quick Start*. For information on how to upgrade your product, see the *SUSE Linux Enterprise Server 12 Deployment Guide*, chapter *Updating SUSE Linux Enterprise*. It is available at <http://www.suse.com/documentation/> .
3. After the upgrade process has finished, reboot each node with version 12 of SUSE Linux Enterprise Server and SUSE Linux Enterprise High Availability Extension.
4. If you use OCFS2 in your cluster setup, update the on-device structure by executing the following command:

```
root # tuneefs.ocfs2 --update-cluster-stack PATH_TO_DEVICE
```

It adds additional parameters to the disk which are needed for the updated OCFS2 version that is shipped with SUSE Linux Enterprise High Availability Extension 12.

5. To update `/etc/corosync/corosync.conf` for Corosync version 2:
 - a. Log in to one node and start the YaST cluster module.
 - b. Switch to the *Communication Channels* category and enter values for the following new parameters: *Cluster Name* and *Expected Votes*. For details, see *Procedure 3.5, "Defining the First Communication Channel"*.
If YaST should detect any other options that are invalid or missing according to Corosync version 2, it will prompt you to change them.
 - c. Confirm your changes in YaST to update `/etc/corosync/corosync.conf`.
 - d. If Csync2 is configured for your cluster, use the following command to push the updated Corosync configuration to the other cluster nodes:

```
root # csync2 -xv
```

For details on Csync2, see [Section 3.5.4, “Transferring the Configuration to All Nodes”](#). Alternatively, synchronize the updated Corosync configuration by manually copying [/etc/corosync/corosync.conf](#) to all cluster nodes.

6. Log in to each node and start the cluster stack with:

```
root # systemctl start pacemaker.service
```

7. Check the cluster status with **crm status** or with Hawk.

If you have an existing GEO cluster setup and want to upgrade it to run with High Availability Extension 12 and GEO Clustering for SUSE Linux Enterprise High Availability Extension 12, see the additional instructions in the *Quick Start* for GEO Clustering for SUSE Linux Enterprise High Availability Extension. It is available at <http://www.suse.com/documentation/> [↗](#). Refer to the section *Upgrading from SLE HA 11 SP3 to SLE HA 12*.



Note: Reverting after Upgrade

After the upgrade process to product version 12, reverting back to product version 11 is *not* supported.

D.3 Updating Software Packages on Cluster Nodes

Before installing any package updates on a node, check the following:

- Does the update affect any packages belonging to SUSE Linux Enterprise High Availability Extension or the GEO Clustering for SUSE Linux Enterprise High Availability Extension add-on? If yes: Stop the cluster stack on the node before starting the software update.

```
root # systemctl stop pacemaker.service
```

- Does the package update require a reboot? If yes: Stop the cluster stack on the node before starting the software update:

```
root # systemctl stop pacemaker.service
```


If none of the situations above do apply, you do not need to stop the cluster stack. In that case, put the cluster into maintenance mode before starting the software update:

```
root # crm configure property maintenance-mode=true
```

For more details on maintenance mode, see [Section 4.7, “Maintenance Mode”](#).



Warning: Active Cluster Stack During Update

If the cluster resource manager on a node is active during the software update, this can lead to unpredictable results like fencing of active nodes.

After the update has been successfully installed, remove the cluster maintenance mode:

```
root # crm configure property maintenance-mode=true
```

or restart the cluster stack on the respective node with:

```
root # systemctl start pacemaker.service
```

D.4 For More Information

For detailed information about any changes and new features of the product you are upgrading to, refer to its release notes. They are available from <https://www.suse.com/releasenotes/> [↗](#).

E Documentation Updates






This chapter lists content changes for this document since the release of SUSE® Linux Enterprise High Availability Extension 11 SP3.


This manual was updated on the following dates:

- *Section E.1, “October, 2014 (Initial Release of SUSE Linux Enterprise High Availability Extension 12)”*

E.1 October, 2014 (Initial Release of SUSE Linux Enterprise High Availability Extension 12)

General

- Adjusted *Section 3.4, “Automatic Cluster Setup (ha-cluster-bootstrap)”* due to the renaming of the bootstrap scripts `sleha-init`, `sleha-join`, and `sleha-remove` to `ha-cluster-init`, `ha-cluster-join`, and `ha-cluster-remove`, respectively. Also the location of the log file has changed from `/var/log/sleha-bootstrap.log` to `/var/log/ha-cluster-bootstrap.log` (http://bugzilla.novell.com/show_bug.cgi?id=853772 ).
- Replaced any occurrence of `chkconfig` and `rc*` commands with systemd equivalents (http://bugzilla.novell.com/show_bug.cgi?id=853533 ).
- Introduced a consistent naming scheme for cluster names, node names, cluster resources, and constraints and applied it to the documentation. See *Appendix B, Naming Conventions*. (Fate#314938).
- Removed Pacemaker GUI and any references pointing to it (Fate#312948, https://bugzilla.novell.com/show_bug.cgi?id=853508 ).
- Replaced OpenAIS with Corosync (https://bugzilla.novell.com/show_bug.cgi?id=853507 , https://bugzilla.novell.com/show_bug.cgi?id=853556 ).
- Removed all occurrences of the following parameters as they are obsolete: `default-resource-stickiness`, `is-managed-default`, and `default-action-timeout`.
- Improved the consistency of crm shell examples.

- Adjusted changed package names throughout the whole manual (Fate#314807).
- Removed list of resource agents from the appendix.
- Moved documentation for GEO Clustering for SUSE Linux Enterprise High Availability Extension into a separate document (Fate#316121). See the new *GEO Clustering for SUSE Linux Enterprise High Availability Extension Quick Start*, available from <http://www.suse.com/documentation/> .
- Removed *Example of Setting Up a Simple Testing Resource* from the appendix as this can be done now by the Hawk wizard.
- Changed terminology for master-slave resources, which are now called multi-state resources in the upstream documentation.
- Updated the screenshots.
- Mentioned both hb_report and crm_report as command line tools for creating detailed cluster reports.
- Numerous small fixes and additions to the manual based on technical feedback.

Chapter 1, Product Overview

- Changed terminology from multi-site clusters to geographically dispersed (or GEO) clusters for consistency reasons.
- Added section about availability of SUSE Linux Enterprise High Availability Extension and GEO Clustering for SUSE Linux Enterprise High Availability Extension as add-on products: *Section 1.1, "Availability as Add-On/Extension"*.

Chapter 2, System Requirements and Recommendations

- Restructured contents.
- Mentioned how to create a cluster report when using a non-standard SSH port (Fate#314906). See *SSH*.

Chapter 3, Installation and Basic Setup

- Updated *Section 3.5, "Manual Cluster Setup (YaST)"* to reflect the software changes to the *YaST Cluster Module*.
- Csync2 also accepts a combination of hostname and IP address for each cluster node. See *Section 3.5.4, "Transferring the Configuration to All Nodes"* (Fate#314956).

- *Section 3.6, “Mass Deployment with AutoYaST”* mentions the option to clone DRBD settings via AutoYaST (Fate#315128).
- *Procedure 3.2, “Automatically Setting Up the First Node”* mentions `-t` option for `ha-cluster-init` to perform additional cluster configurations based on templates). (https://bugzilla.novell.com/show_bug.cgi?id=821123) ↗

Chapter 4, Configuration and Administration Basics

- *Section 4.1.3, “Option stonith-enabled”* mentions policy change in DLM services when the global cluster option `stonith-enabled` is set to `false` (Fate#315195).
- *Section 4.4.7, “Grouping Resources by Using Tags”* describes a new option to group conceptually related resources, without creating any colocation or ordering relationship between them (Fate#315101).
- *Section 4.4.1.1, “Resource Sets”* explains the concept of resource sets as an alternative format for defining constraints. As of SUSE Linux Enterprise High Availability Extension 12, resource sets can now also be used within location constraints (whereas they could formerly only be used within colocation and ordering constraints).
- Restructured *Section 4.7, “Maintenance Mode”* to also cover the option of setting a whole cluster to maintenance mode (https://bugzilla.novell.com/show_bug.cgi?id=829864 ↗).
- Added attributes for `pacemaker_remote` service to *Table 4.1, “Options for a Primitive Resource”* and added new section: *Section 4.5, “Managing Services on Remote Hosts”* (https://bugzilla.novell.com/show_bug.cgi?id=853535 ↗).
- Added new resource agent classes to *Section 4.2.2, “Supported Resource Agent Classes”*: `systemd`, `service`, and `nagios` (https://bugzilla.novell.com/show_bug.cgi?id=853520 ↗).
- Updated *Section 4.5.1, “Monitoring Services on Remote Hosts with Monitoring Plug-ins”* to reflect the package name changes from `nagios-plugins` and `nagios-plugins-metadata` to `monitoring-plugins` and `monitoring-plugins-metadata`, respectively (Fate#317780).

Chapter 5, Configuring and Managing Cluster Resources (Web Interface)

- Updated *Section 5.1.2, “Main Screen: Cluster Status”* to reflect additional information that is now presented there (Fate#316303, Fate#316303, https://bugzilla.novell.com/show_bug.cgi?id=883529 ↗).
- Updated the chapter to reflected the new features that have been described in *Chapter 4, Configuration and Administration Basics*.
- *Procedure 5.28, “Using the History Explorer Offline”* describes a new Hawk function to view and analyze the transitions of clusters that you are currently not connected to (Fate#313354).
- Used example of setting up an NFS server to demonstrate use of Hawk *Cluster Setup Wizard* (Fate#315163). See *Procedure 5.4, “Using the Setup Wizard”*.
- The description of all Hawk functions that are related to GEO clusters has been moved to a separate document. See the new *GEO Clustering for SUSE Linux Enterprise High Availability Extension Quick Start*, available from <http://www.suse.com/documentation/> ↗.

Chapter 6, Configuring and Managing Cluster Resources (Command Line)

- Added the section: *Section 6.4.4.3, “Collocating Sets for Resources Without Dependency”* (Fate#314917).
- Added a section about the health status (Fate#316464): *Section 6.5.7, “Getting Health Status”*.
- Added a section about tagging resources (Fate#315101): *Section 6.5.5, “Grouping/Tagging Resources”*.
- Updated overview section about tab completion and getting help (https://bugzilla.novell.com/show_bug.cgi?id=853643 ↗).

Chapter 9, Access Control Lists

- Restructured chapter.
- Added *Section 9.4, “Configuring ACLs with Hawk”*.

Chapter 11, Load Balancing

- Added configuration of HAProxy.

Chapter 14, GFS2

- New chapter.

Chapter 15, DRBD

- Reworked *Section 15.2, "Installing DRBD Services"*
- Inserted prompts in shell sessions
- Replaced all `rcdrbd` commands with `systemctl`

Chapter 16, Cluster Logical Volume Manager (cLVM)

- Describe `mirrored` in a clustered environment (Fate#314367): *Section 16.2.1, "Configuring Cmirrord"*.

Chapter 17, Storage Protection

- Added configuration of a `sg_persist` resource (FATE#312345)
- Added section about watchdog (https://bugzilla.novell.com/show_bug.cgi?id=892344 ↗)
- Added short notice about watchdog and other software that access the watchdog timer (https://bugzilla.novell.com/show_bug.cgi?id=891340 ↗)
- Added hint about watchdog driver loaded at boot time (https://bugzilla.novell.com/show_bug.cgi?id=892344 ↗)
- Added hints about `stonith-timeout` (https://bugzilla.novell.com/show_bug.cgi?id=891346 ↗)
- Adjusted *Section 17.1.3.3, "Starting the SBD Daemon"* (https://bugzilla.novell.com/show_bug.cgi?id=891499 ↗)
- Enhanced section about the relationship between different timeouts (https://bugzilla.novell.com/show_bug.cgi?id=891346 ↗)

Chapter 18, Samba Clustering

- Documented separate Samba and Winbind resources (Fate#316336): *Procedure 18.1, "Setting Up a Basic Clustered Samba Server"*.

Chapter 19, Disaster Recovery with Rear (Relax-and-Recover)

- Updated example screen of `/etc/rear/local.conf` (https://bugzilla.novell.com/show_bug.cgi?id=885881 ↗).
- Fixed minor problems (https://bugzilla.novell.com/show_bug.cgi?id=878054 ↗).
- Removed any occurrence of **rear-SUSE** and configuration file `/etc/rear/site.conf`.

Appendix B, Naming Conventions

- New appendix explaining naming scheme.

Appendix D, Upgrading Your Cluster and Updating Software Packages

- Added definition of terms: *Section D.1, "Terminology"*.
- Added *Section D.2.1, "Upgrading from SLE HA 11 SP3 to SLE HA 12"* to *Section D.2, "Upgrading your Cluster to the Latest Product Version"*.
- Added a new section: *Section D.3, "Updating Software Packages on Cluster Nodes"*.

Quick Start for GEO Clustering for SUSE Linux Enterprise High Availability Extension

- Completely extended chapter.
- Updated example for booth configuration file. (https://bugzilla.novell.com/show_bug.cgi?id=863419 ↗).

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. For example, it has *some* services running, but can takeover other services from the other node. Compare to primary/secondary and dual-primary in DRBD speak.

arbitrator

Additional instance in a GEO cluster that helps to reach consensus about decisions such as failover of resources across sites. Arbitrators are single machines that run one or more booth instances in a special mode.

AutoYaST

AutoYaST is a system for installing one or more SUSE Linux Enterprise systems automatically and without user intervention.

bindnetaddr (bind network address)

The network address the Corosync executive should bind to.

booth

The instance that manages the failover process between the sites of a GEO cluster. It aims to get multi-site resources active on one and only one site. This is achieved by using so-called tickets that are treated as failover domain between cluster sites, in case a site should be down.

boothd (booth daemon)

Each of the participating clusters and arbitrators in a GEO cluster runs a service, the boothd. It connects to the booth daemons running at the other sites and exchanges connectivity details.

CCM (consensus cluster membership)

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.

CIB (cluster information base)

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 *DC (designated coordinator)* and replicated to the other nodes. Normal read and write operations on the CIB are serialized through the master CIB.

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 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 into 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*).

concurrency violation

A resource that should be running on only one node in the cluster is running on several nodes.

conntrack tools

Allow interaction with the in-kernel connection tracking system for enabling *stateful* packet inspection for iptables. Used by the High Availability Extension to synchronize the connection status between cluster nodes.

CRM (cluster resource manager)

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, the membership layer, and booth.

crmd (cluster resource manager daemon)

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 ran on), a new one is established.

crmsh

The command line utility crmsh manages your cluster, nodes, and resources.

See *Chapter 6, Configuring and Managing Cluster Resources (Command Line)* for more information.

Csync2

A synchronization tool that can be used to replicate configuration files across all nodes in the cluster, and even across GEO clusters.

DC (designated coordinator)

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 configuration and resource allocation information from the current DC. The DC is elected from all nodes in the cluster after a membership change.

Disaster

Unexpected interruption of critical infrastructure induced by nature, humans, hardware failure, or software bugs.

Disaster Recovery

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.

DLM (distributed lock manager)

DLM coordinates disk access for clustered file systems and administers file locking to increase performance and availability.

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.

existing cluster

The term “existing cluster” is used to refer to any cluster that 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.

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.

geo cluster (geographically dispersed cluster)

See *GEO cluster*.

heartbeat

A CCM, in version 3 an alternative to corosync. Supports more than two communication paths, but not cluster filesystems.

load balancing

The ability to make several servers participate in the same service and do the same work.

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.

LRM (local resource manager)

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.

mcastaddr (multicast address)

IP address to be used for multicasting by the Corosync executive. The IP address can either be IPv4 or IPv6.

mcastport (multicast port)

The port to use for cluster communication.

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.

GEO 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. GEO 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.

PE (policy engine)

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.

RA (resource agent)

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), LSB (Linux Standard Base init scripts), and Heartbeat resource agents. For more information, refer to [Section 4.2.2, “Supported Resource Agent Classes”](#).

Rear (Relax and Recover)

An administrator tool set for creating disaster recovery images.

resource

Any type of service or application that is known to Pacemaker. Examples include an IP address, a file system, or a database.

The term “resource” is also used for DRBD, where it names a set of block devices that are using a common connection for replication.

RRP (redundant ring protocol)

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. Corosync supports the Totem Redundant Ring Protocol.

SBD (STONITH by disk)

In an environment where all nodes have access to shared storage, a small partition is used for disk-based fencing. Needs a hardware watchdog on each node to ensure that misbehaving nodes are really stopped.

SFEX (shared disk file exclusiveness)

SFEX provides storage protection over SAN.

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.

SPOF (single point of failure)

Any component of a cluster that, should it fail, triggers the failure of the entire cluster.

STONITH

The acronym for “Shoot the other node in the head”. It refers to the fencing mechanism that shuts down a misbehaving node to prevent it from causing trouble in a cluster.

switchover

Planned, on-demand moving of services to other nodes in a cluster. See *failover*.

ticket

A component used in GEO 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.

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 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 draw-

ing 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.