This guide is intended for administrators who need to set up, configure, and maintain clusters with SUSE® Linux Enterprise High Availability Extension. For quick and efficient configuration and administration, the product includes both a graphical user interface and a command line interface (CLI). For performing key tasks, both approaches are covered in this guide. Thus, you can choose the appropriate tool that matches your needs.
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 trademarks, see http://www.suse.com/company/legal/. All other third-party trademarks are the property of their respective owners. Trademark symbols (®, ™ etc.) denote trademarks of SUSE and its affiliates. Asterisks (*) denote third-party trademarks.

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 xvi
1 Available Documentation xvii
2 Improving the Documentation xviii
3 Documentation Conventions xix
4 Product Life Cycle and Support xx
   Support Statement for SUSE Linux Enterprise High Availability
   Extension xxi • Technology Previews xxii

I INSTALLATION, SETUP AND UPGRADE 1

1 Product Overview 2
1.1 Availability as 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 13
   Architecture Layers 13 • Process Flow 15

2 System Requirements and Recommendations 16
2.1 Hardware Requirements 16
2.2 Software Requirements 17
2.3 Storage Requirements 18
6 Configuration and Administration Basics 57

6.1 Use Case Scenarios 57

6.2 Quorum Determination 58
Global Cluster Options 59 • Global Option no-quorum-policy 59 • Global Option stonith-enabled 60 • Corosync
Configuration for Two-Node Clusters 60 • Corosync Configuration for N-Node Clusters 61

6.3 Cluster Resources 62
Resource Management 62 • Supported Resource Agent Classes 63 • Types of Resources 65 • Resource Templates 65 • Advanced Resource Types 66 • Resource Options (Meta Attributes) 69 • Instance Attributes (Parameters) 72 • Resource Operations 74 • Timeout Values 76

6.4 Resource Monitoring 77

6.5 Resource Constraints 78
Types of Constraints 78 • Scores and Infinity 81 • Resource Templates and Constraints 82 • Failover Nodes 83 • Failback Nodes 84 • Placing Resources Based on Their Load Impact 85 • Grouping Resources by Using Tags 88

6.6 Managing Services on Remote Hosts 88
Monitoring Services on Remote Hosts with Monitoring Plug-ins 89 • Managing Services on Remote Nodes with pacemaker_remote 90

6.7 Monitoring System Health 91

6.8 For More Information 93

7 Configuring and Managing Cluster Resources with Hawk2 94

7.1 Hawk2 Requirements 94

7.2 Logging In 95
7.3 Hawk2 Overview: Main Elements 96
    Left Navigation Bar 97 • Top-Level Row 98

7.4 Configuring Global Cluster Options 99

7.5 Configuring Cluster Resources 101
    Showing the Current Cluster Configuration (CIB) 102 • Adding Resources with the Wizard 102 • Adding Simple Resources 103 • Adding Resource Templates 105 • Modifying Resources 105 • Adding STONITH Resources 107 • Adding Cluster Resource Groups 108 • Adding Clone Resources 110 • Adding Multi-state Resources 111 • Grouping Resources by Using Tags 112 • Configuring Resource Monitoring 113

7.6 Configuring Constraints 116
    Adding Location Constraints 116 • Adding Colocation Constraints 117 • Adding Order Constraints 119 • Using Resource Sets for Constraints 121 • For More Information 122 • Specifying Resource Failover Nodes 123 • Specifying Resource Failback Nodes (Resource Stickiness) 124 • Configuring Placement of Resources Based on Load Impact 125

7.7 Managing Cluster Resources 128
    Editing Resources and Groups 128 • Starting Resources 129 • Cleaning Up Resources 130 • Removing Cluster Resources 130 • Migrating Cluster Resources 131

7.8 Monitoring Clusters 132
    Monitoring a Single Cluster 132 • Monitoring Multiple Clusters 133

7.9 Using the Batch Mode 137

7.10 Viewing the Cluster History 141
    Viewing Recent Events of Nodes or Resources 141 • Using the History Explorer for Cluster Reports 142 • Viewing Transition Details in the History Explorer 144

7.11 Verifying Cluster Health 146
8 Configuring and Managing Cluster Resources (Command Line) 147

8.1 crmsh—Overview 147
  Getting Help 148 • Executing crmsh's Subcommands 149 • Displaying Information about OCF Resource Agents 151 • Using crmsh's Shell Scripts 152 • Using crmsh's Cluster Scripts 153 • Using Configuration Templates 156 • Testing with Shadow Configuration 158 • Debugging Your Configuration Changes 159 • Cluster Diagram 159

8.2 Managing Corosync Configuration 159

8.3 Configuring Cluster Resources 160
  Loading Cluster Resources from a File 161 • Creating Cluster Resources 161 • Creating Resource Templates 162 • Creating a STONITH Resource 163 • Configuring Resource Constraints 164 • Specifying Resource Failover Nodes 167 • Specifying Resource Failback Nodes (Resource Stickiness) 167 • Configuring Placement of Resources Based on Load Impact 168 • Configuring Resource Monitoring 170 • Configuring a Cluster Resource Group 171 • Configuring a Clone Resource 171

8.4 Managing Cluster Resources 173
  Showing Cluster Resources 173 • Starting a New Cluster Resource 174 • Stopping a Cluster Resource 175 • Cleaning Up Resources 175 • Removing a Cluster Resource 176 • Migrating a Cluster Resource 176 • Grouping/Tagging Resources 177 • Getting Health Status 177

8.5 Setting Passwords Independent of cib.xml 178

8.6 Retrieving History Information 178

8.7 For More Information 180

9 Adding or Modifying Resource Agents 181

9.1 STONITH Agents 181

9.2 Writing OCF Resource Agents 181

9.3 OCF Return Codes and Failure Recovery 183
# QDevice and QNetd

## 12 Conceptual Overview

### 12.1 Conceptual Overview

### 12.2 Requirements and Prerequisites

### 12.3 Setting Up the QNetd Server

### 12.4 Connecting QDevice Clients to the QNetd Server

### 12.5 Setting Up a QDevice with Heuristics

### 12.6 Checking and Showing Quorum Status

### 12.7 For More Information

# Access Control Lists

## 13 Conceptual Overview

### 13.1 Requirements and Prerequisites

### 13.2 Conceptual Overview

### 13.3 Enabling Use of ACLs in Your Cluster

### 13.4 Creating a Read-Only Monitor Role

- Creating a Read-Only Monitor Role with Hawk2
- Creating a Read-Only Monitor Role with crmsh

### 13.5 Removing a User

- Removing a User with Hawk2
- Removing a User with crmsh

### 13.6 Removing an Existing Role

- Removing an Existing Role with Hawk2
- Removing an Existing Role with crmsh

### 13.7 Setting ACL Rules via XPath Expressions

### 13.8 Setting ACL Rules via Abbreviations

# Network Device Bonding

## 14 Configuring Bonding Devices with YaST

### 14.1 Configuring Bonding Devices with YaST

### 14.2 Hotplugging devices into a bond

### 14.3 For More Information
15 Load Balancing 240

15.1 Conceptual Overview 240

15.2 Configuring Load Balancing with Linux Virtual Server 242
   Director 242 • User Space Controller and Daemons 242 • Packet Forwarding 243 • Scheduling Algorithms 243 • Setting Up IP Load Balancing with YaST 244 • Further Setup 249

15.3 Configuring Load Balancing with HAProxy 249

15.4 For More Information 253

16 Geo Clusters (Multi-Site Clusters) 254

17 Executing Maintenance Tasks 255

17.1 Implications of Taking Down a Cluster Node 255

17.2 Different Options for Maintenance Tasks 257

17.3 Preparing and Finishing Maintenance Work 258

17.4 Putting the Cluster in Maintenance Mode 259

17.5 Putting a Node in Maintenance Mode 260

17.6 Putting a Node in Standby Mode 260

17.7 Putting a Resource into Maintenance Mode 261

17.8 Putting a Resource into Unmanaged Mode 262

17.9 Rebooting a Cluster Node While in Maintenance Mode 262

III STORAGE AND DATA REPLICATION 264

18 Distributed Lock Manager (DLM) 265

18.1 Protocols for DLM Communication 265

18.2 Configuring DLM Cluster Resources 265

19 OCFS2 267

19.1 Features and Benefits 267
19.2  OCFS2 Packages and Management Utilities  268
19.3  Configuring OCFS2 Services and a STONITH Resource  269
19.4  Creating OCFS2 Volumes  270
19.5  Mounting OCFS2 Volumes  273
19.6  Configuring OCFS2 Resources With Hawk2  274
19.7  Using Quotas on OCFS2 File Systems  275
19.8  For More Information  276

20  GFS2  277
20.1  GFS2 Packages and Management Utilities  277
20.2  Configuring GFS2 Services and a STONITH Resource  278
20.3  Creating GFS2 Volumes  279
20.4  Mounting GFS2 Volumes  280

21  DRBD  282
21.1  Conceptual Overview  282
21.2  Installing DRBD Services  284
21.3  Setting Up DRBD Service  284
     Configuring DRBD Manually  286  •  Configuring DRBD with
     YaST  288  •  Initializing and Formatting DRBD Resource  290
21.4  Migrating from DRBD 8 to DRBD 9  291
21.5  Creating a Stacked DRBD Device  292
21.6  Using Resource-Level Fencing with STONITH  294
21.7  Testing the DRBD Service  295
21.8  Monitoring DRBD Devices  297
21.9  Tuning DRBD  297
24.3 Joining an Active Directory Domain 326
24.4 Debugging and Testing Clustered Samba 327
24.5 For More Information 329

25  Disaster Recovery with Rear (Relax-and-Recover) 330
25.1 Conceptual Overview 330
  Creating a Disaster Recovery Plan 330  •  What Does Disaster Recovery
  Mean? 331  •  How Does Disaster Recovery With Rear Work? 331  •  Rear
  Requirements 331  •  Rear Version Updates 331  •  Limitations with
  Btrfs 332  •  Scenarios and Backup Tools 333  •  Basic Steps 333

25.2 Setting Up Rear and Your Backup Solution 334
25.3 Creating the Recovery Installation System 336
25.4 Testing the Recovery Process 336
25.5 Recovering from Disaster 337
25.6 For More Information 337

IV  APPENDIX 338

A  Troubleshooting 339
A.1 Installation and First Steps 339
A.2 Logging 340
A.3 Resources 341
A.4 STONITH and Fencing 343
A.5 History 343
A.6 Hawk2 345
A.7 Miscellaneous 345
A.8 For More Information 348
B  Naming Conventions 349
C  Cluster Management Tools (Command Line) 350
D  Running Cluster Reports Without root Access 352
   D.1  Creating a Local User Account 352
   D.2  Configuring a Passwordless SSH Account 353
   D.3  Configuring sudo 355
   D.4  Generating a Cluster Report 357

Glossary 358
E  GNU Licenses 365
About This Guide

This guide is intended for administrators who need to set up, configure, and main-
tain clusters with SUSE® Linux Enterprise High Availability Extension. For quick
and efficient configuration and administration, the product includes both a graph-
ical user interface and a command line interface (CLI). For performing key tasks,
both approaches are covered in this guide. Thus, you can choose the appropriate
tool that matches your needs.

This guide is divided into the following parts:

Installation, Setup and Upgrade

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. Learn how to upgrade your cluster to the most recent release version or how
to update individual packages.

Configuration and Administration

Add, configure and manage cluster resources with either the Web interface (Hawk2), or
the command line interface (crmsh). To avoid unauthorized access to the cluster configu-
ration, define roles and assign them to certain users for fine-grained control. Learn how to
use load balancing and fencing. If you consider writing your own resource agents or mod-
ifying 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 Cluster Logical Volume Manager (Cluster LVM). For replication
of your data, use DRBD®. It lets you 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

Contains an overview of common problems and their solution. Presents the naming con-
ventions used in this documentation with regard to clusters, resources and constraints.
Contains a glossary with HA-specific terminology.
1 Available Documentation

Note: Online Documentation and Latest Updates

Documentation for our products is available at https://documentation.suse.com/, where you can also find the latest updates, and browse or download the documentation in various formats. The latest documentation updates can usually be found in the English language version.

The following documentation is available for this product:

Installation and Setup Quick Start

This document guides you through the setup of a very basic two-node cluster, using the bootstrap scripts provided by the ha-cluster-bootstrap package. This includes the configuration of a virtual IP address as a cluster resource and the use of SBD on shared storage as a node fencing mechanism.

Administration Guide

This guide is intended for administrators who need to set up, configure, and maintain clusters with SUSE® Linux Enterprise High Availability Extension. For quick and efficient configuration and administration, the product includes both a graphical user interface and a command line interface (CLI). For performing key tasks, both approaches are covered in this guide. Thus, you can choose the appropriate tool that matches your needs.

Highly Available NFS Storage with DRBD and Pacemaker

This document describes how to set up highly available NFS storage in a two-node cluster, using the following components: DRBD® (Distributed Replicated Block Device), LVM (Logical Volume Manager), and Pacemaker as cluster resource manager.

Pacemaker Remote Quick Start

This document guides you through the setup of a High Availability cluster with a remote node or a guest node, managed by Pacemaker and pacemaker_remote. Remote in pacemaker_remote does not refer to physical distance, but to the special status of nodes that do not run the complete cluster stack and thus are not regular members of the cluster.

Geo Clustering Quick Start
Geo clustering protects workloads across globally distributed data centers. This document guides you through the basic setup of a Geo cluster, using the Geo bootstrap scripts provided by the ha-cluster-bootstrap package.

**Geo Clustering Guide**

This document covers the setup options and parameters for Geo clusters and their components, such as booth ticket manager, the specific Csync2 setup, and the configuration of the required cluster resources (and how to transfer them to other sites in case of changes). Learn how to monitor and manage Geo clusters from command line or with the Hawk2 Web interface.

## 2 Improving the Documentation

Your feedback and contributions to this documentation are welcome. The following channels for giving feedback are available:

### Service Requests and Support

For services and support options available for your product, see [https://www.suse.com/support/](https://www.suse.com/support/).

To open a service request, you need a SUSE subscription registered at SUSE Customer Center. Go to [https://scc.suse.com/support/requests](https://scc.suse.com/support/requests), log in, and click *Create New*.

### Bug Reports

Report issues with the documentation at [https://bugzilla.suse.com/](https://bugzilla.suse.com/). To simplify this process, you can use the *Report Documentation Bug* links next to headlines in the HTML version of this document. These preselect the right product and category in Bugzilla and add a link to the current section. You can start typing your bug report right away. A Bugzilla account is required.

### Contributions

To contribute to this documentation, use the *Edit Source* links next to headlines in the HTML version of this document. They take you to the source code on GitHub, where you can open a pull request. A GitHub account is required.

💡 **Note: *Edit Source* only available for English**

The *Edit Source* links are only available for the English version of each document. For all other languages, use the *Report Documentation Bug* links instead.
For more information about the documentation environment used for this document-

**Mail**

You can also report errors and send feedback concerning the documentation to [doc-
team@suse.com](mailto:doc-team@suse.com). Include the document title, the product version, and the publication date of the document. Additionally, include the relevant section number and title (or provide the URL) and provide a concise description of the problem.

## 3 Documentation Conventions

The following notices and typographical conventions are used in this documentation:

- **tux > command**

  Commands that can be run by any user, including the *root* user.

- **root # command**

  Commands that must be run with *root* privileges. Often you can also prefix these com-
  mands with the *sudo* command to run them.

- **crm(live)#**

  Commands executed in the interactive crm shell. For details, see *Chapter 8, Configuring and Managing Cluster Resources (Command Line).*

- **/etc/passwd**: directory names and file names

- **PLACEHOLDER**: replace *PLACEHOLDER* with the actual value

- **PATH**: the environment variable PATH

- **ls, --help**: commands, options, and parameters

- **user**: users or groups

- **packagename**: name of a package

- **Alt, Alt–F1**: a key to press or a key combination; keys are shown in uppercase as on a keyboard
For an overview of naming conventions with regard to cluster nodes and names, resources, and constraints, see Appendix B, Naming Conventions.

4 Product Life Cycle and Support

SUSE products are supported for up to 13 years. To check the life cycle dates for your product, see https://www.suse.com/lifecycle/.
For SUSE Linux Enterprise, the following life cycles and release cycles apply:

- SUSE Linux Enterprise Server has a 13-year life cycle: 10 years of general support and three years of extended support.
- SUSE Linux Enterprise Desktop has a 10-year life cycle: seven years of general support and three years of extended support.
- Major releases are published every four years. Service packs are published every 12-14 months.
- SUSE supports previous SUSE Linux Enterprise service packs for six months after the release of a new service pack.


Modules have a different life cycle, update policy, and update timeline than their base products. Modules contain software packages and are fully supported parts of SUSE Linux Enterprise High Availability Extension.

### 4.1 Support Statement for SUSE Linux Enterprise High Availability Extension

To receive support, you need an appropriate subscription with SUSE. To view the specific support offerings available to you, go to [https://www.suse.com/support/](https://www.suse.com/support/) and select your product.

The support levels are defined as follows:

**L1**

Problem determination, which means technical support designed to provide compatibility information, usage support, ongoing maintenance, information gathering and basic troubleshooting using available documentation.

**L2**

Problem isolation, which means technical support designed to analyze data, reproduce customer problems, isolate problem area and provide a resolution for problems not resolved by Level 1 or prepare for Level 3.

**L3**
Problem resolution, which means technical support designed to resolve problems by engaging engineering to resolve product defects which have been identified by Level 2 Support.

For contracted customers and partners, SUSE Linux Enterprise High Availability Extension is delivered with L3 support for all packages, except for the following:

- Technology Previews
- Sound, graphics, fonts and artwork.
- Packages that require an additional customer contract.
- Some packages shipped as part of the module Workstation Extension are L2-supported only.
- Packages with names ending in `-devel` (containing header files and similar developer resources) will only be supported together with their main packages.

SUSE will only support the usage of original packages. That is, packages that are unchanged and not recompiled.

### 4.2 Technology Previews

Technology previews are packages, stacks, or features delivered by SUSE to provide glimpses into upcoming innovations. The previews are included for your convenience to give you the chance to test new technologies within your environment. We would appreciate your feedback! If you test a technology preview, please contact your SUSE representative and let them know about your experience and use cases. Your input is helpful for future development.

However, technology previews come with the following limitations:

- Technology previews are still in development. Therefore, they may be functionally incomplete, unstable, or in other ways *not* suitable for production use.
- Technology previews are *not* supported.
- Technology previews may only be available for specific hardware architectures.
- Details and functionality of technology previews are subject to change. As a result, upgrading to subsequent releases of a technology preview may be impossible and require a fresh installation.

- Technology previews can be dropped at any time. For example, if SUSE discovers that a preview does not meet the customer or market needs, or does not prove to comply with enterprise standards. SUSE does not commit to providing a supported version of such technologies in the future.

For an overview of technology previews shipped with your product, see the release notes at https://www.suse.com/releasenotes/.
I Installation, Setup and Upgrade

1 Product Overview 2
2 System Requirements and Recommendations 16
3 Installing the High Availability Extension 21
4 Using the YaST Cluster Module 24
5 Upgrading Your Cluster and Updating Software Packages 42
1 Product Overview

SUSE® Linux Enterprise High Availability Extension is an integrated suite of open source clustering technologies. It 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 Glossary.

1.1 Availability as Extension

The High Availability Extension is available as an extension to SUSE Linux Enterprise Server 15 SP1.
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 usage.
- Local clusters
- Metro clusters (“stretched” local clusters)
- Geo clusters (geographically dispersed clusters)

⚠️ Important: No support for mixed architectures

All nodes belonging to a cluster should have the same processor platform: x86, IBM Z, or POWER. Clusters of mixed architectures are not supported.

Your cluster can contain up to 32 Linux servers. Using pacemaker_remote, the cluster can be extended to include additional Linux servers beyond this limit. 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, and manage dependencies. They can automatically stop and start services based on highly configurable rules and policies. The High Availability Exten-
sion 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 (OCFS2) and the cluster Logical Volume Manager (cluster LVM2). 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, SUSE Linux Enterprise High Availability Extension also supports CTDB (Cluster 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 15 SP1 ships with Xen, an open source virtualization hypervisor, and with KVM (Kernel-based Virtual Machine). KVM is a virtualization software for Linux which is based on hardware virtualization extensions. The cluster resource manager in the High Availability Extension can recognize, monitor, and manage services running within virtual servers and 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 supports different geographical scenarios, including geographically dispersed clusters (Geo clusters).

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 need to cope with limited network bandwidth and high latency. Storage is replicated asynchronously.

Note: Geo clustering and SAP workloads
Currently Geo clusters do neither support SAP HANA system replication nor SAP S/4HANA and SAP NetWeaver enqueue replication setups.

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 8.1.3, “Displaying Information about OCF Resource Agents”.

1.2.7 User-friendly Administration Tools
The High Availability Extension ships with a set of powerful tools. Use them for basic installation and setup of your cluster and for 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 the Installation and Setup Quick Start. 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 *Chapter 4, Using the YaST Cluster Module*.
- **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 15, Load Balancing*.

**Hawk2**

A user-friendly Web-based interface with which you can monitor and administer your High Availability clusters from Linux or non-Linux machines alike. Hawk2 can be accessed from any machine inside or outside of the cluster by using a (graphical) Web browser. Therefore it is the ideal solution even if the system on which you are working only provides a minimal graphical user interface. For details, *Chapter 7, Configuring and Managing Cluster Resources with Hawk2*.

**crm Shell**

A powerful unified command line interface to configure resources and execute all monitoring or administration tasks. For details, refer to *Chapter 8, 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). Resources can be dynamically switched or moved to any node in the cluster. Resources can be configured to automatically migrate if a node fails, 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. You can 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 and 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 benefits the High Availability Extension can provide.

**Example Cluster Scenario**

Suppose you have configured a three-node cluster, with a Web server installed on each of the
three nodes in the cluster. Each of the nodes 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 con-
nected to each of the nodes in the cluster. The following figure depicts how this setup might look.
During normal cluster operation, each node is in constant communication with the other nodes 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.
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 formerly 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”. It 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, usually 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 depends on how you configured the resources for them. Migrating the services back to Web Server 1 will incur some down-time. Therefore 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. Use cases for this are upgrading or performing scheduled maintenance on Web Server 1, or increasing 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 node fails, another designated node in the cluster automatically mounts the shared disk directories that were previously mounted on the failed node. This gives network users continuous access to the directories on the shared disk subsystem.

**Important: Shared Disk Subsystem with LVM2**

When using a shared disk subsystem with LVM2, 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 figures show how a typical Fibre Channel cluster configuration might look. The green lines depict connections to an Ethernet power switch. Such a device can be controlled over a network and can reboot a node when a ping request fails.
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.
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.
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.
1.5.1.1 Membership and Messaging Layer (Corosync)

This component provides reliable messaging, membership, and quorum information about the cluster. This is handled by the Corosync cluster engine, a group communication system.

1.5.1.2 Cluster Resource Manager (Pacemaker)

Pacemaker as cluster resource manager is the “brain” which reacts to events occurring in the cluster. It is implemented as `pacemaker-controld`, the cluster controller, which coordinates all actions. Events can be nodes that join or leave the cluster, failure of resources, or scheduled activities such as maintenance, for example.

Local Resource Manager

The local resource manager is located between the Pacemaker layer and the resources layer on each node. It is implemented as `pacemaker-execd` daemon. Through this daemon, Pacemaker can start, stop, and monitor resources.

Cluster Information Database (CIB)

On every node, Pacemaker maintains the cluster information database (CIB). It is an XML representation of the cluster configuration (including cluster options, nodes, resources, constraints and the relationship to each other). The CIB also reflects the current cluster status. Each cluster node contains a CIB replica, which is synchronized across the whole cluster. The `pacemaker-based` daemon takes care of reading and writing cluster configuration and status.

Designated Coordinator (DC)

The DC is elected from all nodes in the cluster. This happens if there is no DC yet or if the current DC leaves the cluster for any reason. 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. All other nodes get their configuration and resource allocation information from the current DC.

Policy Engine

The policy engine runs on every node, but the one on the DC is the active one. The engine is implemented as `pacemaker-schedulerd` daemon. When a cluster transition is needed, based on the current state and configuration, `pacemaker-schedulerd` calculates the expected next state of the cluster. It determines what actions need to be scheduled to achieve the next state.
1.5.1.3 Resources and Resource Agents

In a High Availability cluster, the services that need to be highly available are called resources. Resource agents (RAs) are scripts that start, stop, and monitor cluster resources.

1.5.2 Process Flow

The pacemakerd daemon launches and monitors all other related daemons. The daemon that coordinates all actions, pacemaker-controld, has an instance on each cluster node. Pacemaker centralizes all cluster decision-making by electing one of those instances as a primary. Should the elected pacemaker-controld daemon fail, a new primary is established.

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 the crm shell 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 pacemaker-schedulerd then computes the ideal state of the cluster and how it should be achieved. It feeds a list of instructions to the DC. The DC sends commands via the messaging/infrastructure layer which are received by the pacemaker-controld peers on other nodes. Each of them uses its local resource agent executor (implemented as pacemaker-execd) to perform resource modifications. The pacemaker-execd is not cluster-aware and interacts directly with resource agents.

All peer nodes report the results of their operations back to the DC. After 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 pacemaker-schedulerd is invoked again with the new information recorded in the CIB.

In some cases, it may be necessary to power off nodes to protect shared data or complete resource recovery. In a Pacemaker cluster, the implementation of node level fencing is STONITH. For this, Pacemaker comes with a fencing subsystem, pacemaker-fenced. STONITH devices have to be configured as cluster resources (that use specific fencing agents), because this allows to monitor the fencing devices. When clients detect a failure, they send a request to pacemaker-fenced, which then executes the fencing agent to bring down the node.
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.

Servers
1 to 32 Linux servers with software as specified in Section 2.2, “Software Requirements”. The servers can be bare metal or virtual machines. They do not require identical hardware (memory, disk space, etc.), but they must have the same architecture. Cross-platform clusters are not supported.

Using pacemaker_remote, the cluster can be extended to include additional Linux servers beyond the 32-node limit.

Communication Channels
At least two TCP/IP communication media per cluster node. The network equipment must support the communication means you want to use for cluster communication: multicast or unicast. The communication media should support a data rate of 100 Mbit/s or higher. For a supported cluster setup two or more redundant communication paths are required. This can be done via:

• Network Device Bonding (preferred).

• A second communication channel in Corosync.

For details, refer to Chapter 14, Network Device Bonding and Procedure 4.3, “Defining a Redundant Communication Channel”, respectively.

Node Fencing/STONITH
To avoid a “split brain” scenario, clusters need a node fencing mechanism. In a split brain scenario, cluster nodes are divided into two or more groups that do not know about each other (because of a hardware or software failure or because of a cut network connection). A fencing mechanism isolates the node in question (usually by resetting or powering off the node). This is also called STONITH (“Shoot the other node in the head”). A node fencing mechanism can be either a physical device (a power switch) or a mechanism like SBD (STONITH by disk) in combination with a watchdog. Using SBD requires shared storage. Unless SBD is used, each node in the High Availability cluster must have at least one STONITH device. We strongly recommend multiple STONITH devices per node.

**Important: No Support Without STONITH**

- You must have a node fencing mechanism for your cluster.
- The global cluster options `stonith-enabled` and `startup-fencing` must be set to `true`. When you change them, you lose support.

### 2.2 Software Requirements

All nodes that will be part of the cluster need at least the following modules and extensions:

- Base System Module 15 SP1
- Server Applications Module 15 SP1
- SUSE Linux Enterprise High Availability Extension 15 SP1

Depending on the `system roles` you select during installation, the following software patterns are installed by default:

<table>
<thead>
<tr>
<th>System Role</th>
<th>Software Pattern (YaST/Zypper)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HA Node</td>
<td>• High Availability (sles_ha)</td>
</tr>
<tr>
<td></td>
<td>• Enhanced Base System (enhanced_base)</td>
</tr>
</tbody>
</table>
### System Role

<table>
<thead>
<tr>
<th>System Role</th>
<th>Software Pattern (YaST/Zypper)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HA GEO Node</td>
<td>• Geo Clustering for High Availability (ha_geo)</td>
</tr>
<tr>
<td></td>
<td>• Enhanced Base System (enhanced_base)</td>
</tr>
</tbody>
</table>

**Note: Minimal Installation**

An installation via those system roles results in a minimal installation only. You might need to add more packages manually, if required.

For machines that originally had another system role assigned, you need to manually install the `sles_ha` or `ha_geo` patterns and any further packages that you need.

### 2.3 Storage Requirements

Some services require shared storage. If using an external NFS share, it must be reliably accessible from all cluster nodes via redundant communication paths.

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.

- 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—never the backing device. To leverage the redundancy it is possible to use the same NICs as the rest of the cluster.

When using SBD as STONITH mechanism, additional requirements apply for the shared storage. For details, see Section 11.3, “Requirements”.

2.4 Other Requirements and Recommendations

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

Number of Cluster Nodes
For clusters with more than two nodes, it is strongly recommended to use an odd number of cluster nodes to have quorum. For more information about quorum, see Section 6.2, “Quorum Determination”. A regular cluster may contain up to 32 nodes. With the `pacemaker_remote` service, High Availability clusters can be extended to include additional nodes beyond this limit. See Pacemaker Remote Quick Start for more details.

Time Synchronization
Cluster nodes must synchronize to an NTP server outside the cluster. Since SUSE Linux Enterprise High Availability Extension 15, chrony is the default implementation of NTP. For more information, see the Administration Guide for SUSE Linux Enterprise Server 15 SP1 (https://documentation.suse.com/sles/15-SP1/html/SLES-all/cha-ntp.html)

If nodes are not synchronized, the cluster may not work properly. In addition, log files and cluster reports are very hard to analyze without synchronization. If you use the bootstrap scripts, you will be warned if NTP is not configured yet.

Network Interface Card (NIC) Names
Must be identical on all nodes.

Host Name and IP Address

- Use static IP addresses.
- List all cluster nodes in the `/etc/hosts` file with their fully qualified host name and short host name. It is essential that members of the cluster can find each other by name. If the names are not available, internal cluster communication will fail.

For details on how Pacemaker gets the node names, see also http://clusterlabs.org/doc/en-US/Pacemaker/1.1/html/Pacemaker_Explained/s-node-name.html

SSH
All cluster nodes must be able to access each other via SSH. Tools like `crm report` (for troubleshooting) and Hawk2's History Explorer require passwordless SSH access between the nodes, otherwise they can only collect data from the current node.
Note: Regulatory Requirements

If passwordless SSH access does not comply with regulatory requirements, you can use the work-around described in Appendix D, Running Cluster Reports Without root Access for running `crm report`.

For the History Explorer there is currently no alternative for passwordless login.
3 Installing the High Availability Extension

If you are setting up a High Availability cluster with SUSE® Linux Enterprise High Availability Extension for the first time, the easiest way is to start with a basic two-node cluster. You can also use the two-node cluster to run some tests. Afterward, you can add more nodes by cloning existing cluster nodes with AutoYaST. The cloned nodes will have the same packages installed and the same system configuration as the original ones.

If you want to upgrade an existing cluster that runs an older version of SUSE Linux Enterprise High Availability Extension, refer to Chapter 5, Upgrading Your Cluster and Updating Software Packages.

3.1 Manual Installation

For the manual installation of the packages for High Availability Extension refer to Article “Installation and Setup Quick Start”. It leads you through the setup of a basic two-node cluster.

3.2 Mass Installation and Deployment with AutoYaST

After you have installed and set up a two-node cluster, you can extend the cluster by cloning existing nodes with AutoYaST and adding the clones to the cluster.

AutoYaST uses profiles that contains installation and configuration data. A 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).

Important: Identical Hardware

Procedure 3.1, “Cloning a Cluster Node with AutoYaST” assumes you are rolling out SUSE Linux Enterprise High Availability Extension 15 SP1 to a set of machines with identical hardware configurations.

If you need to deploy cluster nodes on non-identical hardware, refer to the SUSE Linux Enterprise 15 SP1 Deployment Guide, chapter Automated Installation, section Rule-Based Autoinstallation.

PROCEDURE 3.1: CLONING A CLUSTER NODE WITH AUTOYAST

1. Make sure the node you want to clone is correctly installed and configured. For details, see the Installation and Setup Quick Start for SUSE Linux Enterprise High Availability Extension or Chapter 4, Using the YaST Cluster Module.

2. Follow the description outlined in the SUSE Linux Enterprise 15 SP1 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 3.2: 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 4.5, “Transferring the Configuration to All Nodes”.

2. To bring the node online, start the cluster services on the cloned node as described in Section 4.8, “Bringing the Cluster Online Node by Node”.

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.
4 Using the YaST Cluster Module

The YaST cluster module allows you to set up a cluster manually (from scratch) or to modify options for an existing cluster.

However, if you prefer an automated approach for setting up a cluster, refer to Article “Installation and Setup Quick Start”. It describes how to install the needed packages and leads you to a basic two-node cluster, which is set up with the ha-cluster-bootstrap scripts.

You can also use a combination of both setup methods, for example: set up one node with YaST cluster and then use one of the bootstrap scripts to integrate more nodes (or vice versa).

4.1 Definition of Terms

Several key terms used in the YaST cluster module and in this chapter are defined below.

**Bind Network Address** *(bindnetaddr)*

The network address the Corosync executive should bind to. To simplify 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.

If *nodelist* with *ringX_addr* is explicitly configured in `/etc/corosync/corosync.conf`, *bindnetaddr* is not strictly required.

**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.
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/.

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 host names 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

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.

Note: Switches and Multicast

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)
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.

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. However, this mode is deprecated.
- If set to `passive`, Corosync sends messages alternatively over the available networks.
- If set to `none`, RRP is disabled.

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).

4.2 YaST Cluster Module

Start YaST and select *High Availability > Cluster*. Alternatively, start the module from command line:

```bash
sudo yast2 cluster
```

The following list shows an overview of the available screens in the YaST cluster module. It also mentions whether the screen contains parameters that are *required* for successful cluster setup or whether its parameters are *optional*.

**Communication Channels (required)**

Allows you to define one or two communication channels for communication between the cluster nodes. As transport protocol, either use multicast (UDP) or unicast (UDPU). For details, see *Section 4.3, “Defining the Communication Channels”*. 
Important: Redundant Communication Paths
For a supported cluster setup two or more redundant communication paths are required. The preferred way is to use network device bonding as described in Chapter 14, Network Device Bonding.
If this is impossible, you need to define a second communication channel in Corosync.

Security (optional but recommended)
Allows you to define the authentication settings for the cluster. HMAC/SHA1 authentication requires a shared secret used to protect and authenticate messages. For details, see Section 4.4, “Defining Authentication Settings”.

Configure Csync2 (optional but recommended)
Csync2 helps you to keep track of configuration changes and to keep files synchronized across the cluster nodes. For details, see Section 4.5, “Transferring the Configuration to All Nodes”.

Configure conntrackd (optional)
Allows you to configure the user space conntrackd. Use the conntrack tools for stateful packet inspection for iptables. For details, see Section 4.6, “Synchronizing Connection Status Between Cluster Nodes”.

Service (required)
Allows you to configure the service for bringing the cluster node online. Define whether to start the cluster services at boot time and whether to open the ports in the firewall that are needed for communication between the nodes. For details, see Section 4.7, “Configuring Services”.

If you start the cluster module for the first time, it appears as a 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.

Note: Settings in the YaST Cluster Module
Some settings in the YaST cluster module apply only to the current node. Other settings may automatically be transferred to all nodes with Csync2. Find detailed information about this in the following sections.
4.3 Defining the Communication Channels

For successful communication between the cluster nodes, define at least one communication channel. As transport protocol, either use multicast (UDP) or unicast (UDPU) as described in Procedure 4.1 or Procedure 4.2, respectively. If you want to define a second, redundant channel (Procedure 4.3), both communication channels must use the same protocol.

Note: Public Clouds: Use Unicast

For deploying SUSE Linux Enterprise High Availability Extension in public cloud platforms, use unicast as transport protocol. Multicast is generally not supported by the cloud platforms themselves.

All settings defined in the YaST Communication Channels screen are written to /etc/corosync/corosync.conf. Find example files for a multicast and a unicast setup in /usr/share/doc/packages/corosync/.

If you are using IPv4 addresses, node IDs are optional. If you are using IPv6 addresses, node IDs are required. Instead of specifying IDs manually for each node, the YaST cluster module contains an option to automatically generate a unique ID for every cluster node.

PROCEDURE 4.1: DEFINING THE FIRST COMMUNICATION CHANNEL (MULTICAST)

When using multicast, 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.

1. Start the YaST cluster module and switch to the Communication Channels category.
2. Set the Transport protocol to Multicast.
3. Define the Bind Network Address. Set the value to the subnet you will use for cluster multicast.
4. Define the Multicast Address.
5. Define the Port.
6. To automatically generate a unique ID for every cluster node keep Auto Generate Node ID enabled.
7. Define a Cluster Name.
8. Enter the number of Expected Votes. This is important for Corosync to calculate quorum in case of a partitioned cluster. By default, each node has 1 vote. The number of Expected Votes must match the number of nodes in your cluster.

9. Confirm your changes.

10. If needed, define a redundant communication channel in Corosync as described in Procedure 4.3, “Defining a Redundant Communication Channel”.

If you want to use unicast instead of multicast for cluster communication, proceed as follows.

PROCEDURE 4.2: DEFINING THE FIRST COMMUNICATION CHANNEL (UNICAST)

1. Start the YaST cluster module and switch to the Communication Channels category.

2. Set the Transport protocol to Unicast.

3. Define the Port.
4. 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.

5. To automatically generate a unique ID for every cluster node keep *Auto Generate Node ID* enabled.

6. Define a **Cluster Name**.

7. Enter the number of *Expected Votes*. This is important for Corosync to calculate *quorum* in case of a partitioned cluster. By default, each node has 1 vote. The number of *Expected Votes* must match the number of nodes in your cluster.

8. Confirm your changes.

9. If needed, define a redundant communication channel in Corosync as described in *Procedure 4.3, “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. If one network fails, the cluster nodes can still communicate via the other network.

The additional communication channel in Corosync will form a second token-passing ring. In `$/etc/corosync/corosync.conf` the first channel you configured is the primary ring and gets the ring number 0. The second ring (redundant channel) gets the ring number 1.

When having defined redundant communication channels in Corosync, use RRP to tell the cluster how to use these interfaces. With RRP, two physically separate networks are used for communication. If one network fails, the cluster nodes can still communicate via the other network.
RRP can have three modes:

- If set to **active**, Corosync uses both interfaces actively. However, this mode is deprecated.
- If set to **passive**, Corosync sends messages alternatively over the available networks.
- If set to **none**, RRP is disabled.

**PROCEDURE 4.3: DEFINING A REDUNDANT COMMUNICATION CHANNEL**

**Important: Redundant Rings and /etc/hosts**

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

1. Start the YaST cluster module and 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, enter the following parameters: the *Bind Network Address* to use, the *Multicast Address* and the *Port* for the redundant channel.
   
   If you use unicast, define the following parameters: the *Bind Network Address* to use, and the *Port*. Enter the IP addresses of all nodes that will be part of the cluster.

4. To tell Corosync how and when to use the different channels, select the *rrp_mode* to use:
   
   - If only one communication channel is defined, *rrp_mode* is automatically disabled (value **none**).
   
   - If set to **active**, Corosync uses both interfaces actively. However, this mode is deprecated.
   
   - If set to **passive**, Corosync sends messages alternatively over the available networks.

   When RRP is used, the High Availability Extension monitors the status of the current rings and automatically re-enables redundant rings after faults.
   
   Alternatively, check the ring status manually with `corosync-cfgtool`. View the available options with `-h`.

5. Confirm your changes.
4.4 Defining Authentication Settings

To define the authentication settings for the cluster, you can use HMAC/SHA1 authentication. This 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 4.4: ENABLING SECURE AUTHENTICATION

1. Start the YaST cluster module and switch to the Security category.


3. For a newly created cluster, click Generate Auth Key File. An authentication key is created and written to /etc/corosync/authkey.
   
   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. Confirm your changes. YaST writes the configuration to /etc/corosync/corosync.conf.
4.5 Transferring the Configuration to All Nodes

Instead of copying the resulting configuration files to all nodes manually, use the `csync2` tool for replication across all nodes in the cluster.

This requires the following basic steps:

1. **Configuring Csync2 with YaST.**
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 to 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 of the system.


### 4.5.1 Configuring Csync2 with YaST

1. Start the YaST cluster module and switch to the Csync2 category.

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

   **Tip: Host Name Resolution**
   
   If host name resolution does not work properly in your network, you can also specify a combination of host name 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*.

5. To *Edit*, *Add* or *Remove* files from the list of files to be synchronized use the respective buttons. You must enter the absolute path name 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. Confirm your changes. YaST writes the Csync2 configuration to `/etc/csync2/csync2.cfg`. 

8. To start the synchronization process now, proceed with Section 4.5.2, “Synchronizing Changes with Csync2”.

4.5.2 Synchronizing Changes with Csync2

To successfully synchronize the files with Csync2, the following requirements must be met:

- The same Csync2 configuration is available on all cluster nodes.
- The same Csync2 authentication key is available on all cluster nodes.
- Csync2 must be running on all cluster nodes.
Before the first Csync2 run, you therefore need to make the following preparations:

PROCEDURE 4.5: PREPARING FOR INITIAL SYNCHRONIZATION WITH CSYNC2

1. Copy the file `/etc/csync2/csync2.cfg` manually to all nodes after you have configured it as described in Section 4.5.1, “Configuring Csync2 with YaST”.

2. Copy the file `/etc/csync2/key_hagroup` that you have generated on one node in Step 3 of Section 4.5.1 to all nodes in the cluster. It is needed for authentication by Csync2. However, do not regenerate the file on the other nodes—it needs to be the same file on all nodes.

3. Execute the following command on all nodes to start the service now:

   ```
   root # systemctl start csync2.socket
   ```

PROCEDURE 4.6: SYNCHRONIZING THE CONFIGURATION FILES WITH CSYNC2

1. To initially synchronize all files once, execute the following command on the machine that you want to copy the configuration from:

   ```
   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 reports 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
   root # 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 machines.

Each time you update files that need to be synchronized, you need to push the changes to the other machines: Run `csync2 -xv` on the machine where you did the changes. If you run the command on any of the other machines with unchanged files, nothing will happen.

### 4.6 Synchronizing Connection Status Between Cluster Nodes

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

**PROCEDURE 4.7: CONFIGURING THE conntrackd WITH YAST**

Use the YaST cluster module to configure the user space `conntrackd` (see Figure 4.5, “YaST Cluster—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. Start the YaST cluster module and switch to the **Configure conntrackd** category.

2. Define the **Multicast Address** to be used for synchronizing the connection status.

3. In **Group Number**, define a numeric ID for the group to synchronize the connection status to.

4. Click **Generate `/etc/conntrackd/conntrackd.conf`** to create the configuration file for `conntrackd`.

5. If you modified any options for an existing cluster, confirm your changes and close the cluster module.

6. For further cluster configuration, click **Next** and proceed with Section 4.7, “Configuring Services”.

7. 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.
After having configured the conntrack tools, you can use them for Linux Virtual Server, see *Load Balancing*.

### 4.7 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 4.8: ENABLING THE CLUSTER SERVICES**

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

```
root #crm cluster start
```

3. To start or stop the cluster services immediately, click the respective button.

4. To open the ports in the firewall that are needed for cluster communication on the current machine, activate *Open Port in Firewall*.

5. Confirm your changes. Note that the configuration only applies to the current machine, not to all cluster nodes.

![Cluster Services Interface](image)

**FIGURE 4.6: YAST CLUSTER—SERVICES**
4.8 Bringing the Cluster Online Node by Node

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

PROCEDURE 4.9: STARTING CLUSTER SERVICES AND CHECKING THE STATUS

1. Log in to an existing node.

2. Check if the service is already running:

   ```
   root # crm cluster status
   ```

   If not, start the cluster services now:

   ```
   root # crm cluster start
   ```

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
   Cluster Summary:
   * Stack: corosync
   * Current DC: alice (version ...) - partition with quorum
   * Last updated: ...
   * Last change: ... by hacluster via crmd on bob
   * 2 nodes configured
   * 1 resource instance configured

   Node List:
   * 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 Hawk2. For more information, see Chapter 8, Configuring and Managing Cluster Resources (Command Line) or Chapter 7, Configuring and Managing Cluster Resources with Hawk2.
5 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. See Section 5.2, “Upgrading your Cluster to the Latest Product Version” versus Section 5.3, “Updating Software Packages on Cluster Nodes.”

If you want to upgrade your cluster, check Section 5.2.1, “Supported Upgrade Paths for SLE HA and SLE HA Geo” and Section 5.2.2, “Required Preparations Before Upgrading” before starting to upgrade.

5.1 Terminology

In the following, find definitions of the most important terms used in this chapter:

**Major Release,**

**General Availability (GA) Version**

A major release is a new product version that brings new features and tools, and decommissions previously deprecated components. It comes with backward incompatible changes.

**Cluster Offline Upgrade**

If a new product version includes major changes that are backward incompatible, the cluster needs to be upgraded by a cluster offline upgrade. You need to take all nodes offline and upgrade the cluster as a whole, before you can bring all nodes back online.

**Cluster Rolling Upgrade**

In a cluster rolling upgrade one cluster node at a time is upgraded while the rest of the cluster is still running. You take the first node offline, upgrade it and bring it back online to join the cluster. Then you continue one by one until all cluster nodes are upgraded to a major version.

**Service Pack (SP)**

Combines several patches into a form that 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, which usually contains security fixes and other important fixes.

Upgrade
Installation of a newer major version of a package or distribution, which brings new features.
See also Cluster Offline Upgrade versus Cluster Rolling Upgrade.

5.2 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 as well as on the target version you want to migrate to.

The High Availability Extension has the same supported upgrade paths as the underlying base system. For a complete overview, see the section Supported Upgrade Paths to SUSE Linux Enterprise Server 15 SP1 in the SUSE Linux Enterprise Server Upgrade Guide.

In addition, the following rules apply, as the High Availability cluster stack offers two methods for upgrading the cluster:

- **Cluster Rolling Upgrade.** A cluster rolling upgrade is only supported within the same major release (from one service pack to the next, or from the GA version of a product to SP1).

- **Cluster Offline Upgrade.** A cluster offline upgrade is required to upgrade from one major release to the next (for example, from SLE HA 12 to SLE HA 15) or from a service pack within one major release to the next major release (for example, from SLE HA 12 SP3 to SLE HA 15).

Section 5.2.1 list the supported upgrade paths and methods for SLE HA (Geo), moving from one version to the next. The column For Details lists the specific upgrade documentation you should refer to (including also the base system and Geo Clustering for SUSE Linux Enterprise High Availability Extension). This documentation is available from:

- https://documentation.suse.com/sles
- https://documentation.suse.com/sle-ha
Important: No Support for Mixed Clusters and Reversion After Upgrade

- Mixed clusters running on SUSE Linux Enterprise High Availability Extension 12/SUSE Linux Enterprise High Availability Extension 15 are not supported.
- After the upgrade process to product version 15, reverting back to product version 12 is not supported.

5.2.1 Supported Upgrade Paths for SLE HA and SLE HA Geo

<table>
<thead>
<tr>
<th>Upgrade From ... To</th>
<th>Upgrade Path</th>
<th>For Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLE HA 11 SP3 to SLE HA (Geo) 12</td>
<td>Cluster Offline Upgrade</td>
<td>• Base System: Deployment Guide for SLES 12, part Updating and Upgrading SUSE Linux Enterprise</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SLE HA: Upgrading from Product Version 11 to 12: Cluster Offline Upgrade</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SLE HA Geo: Geo Clustering Quick Start for SLE HA 12, section Upgrading from SLE HA (Geo) 11 SP3 to SLE HA Geo 12</td>
</tr>
<tr>
<td>SLE HA (Geo) 11 SP4 to SLE HA (Geo) 12 SP1</td>
<td>Cluster Offline Upgrade</td>
<td>• Base System: Deployment Guide for SLES 12 SP1, part Updating and Upgrading SUSE Linux Enterprise</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SLE HA: Upgrading from Product Version 11 to 12: Cluster Offline Upgrade</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SLE HA Geo: Geo Clustering Quick Start for SLE HA 12 SP1, section Upgrading to the Latest Product Version</td>
</tr>
<tr>
<td>Upgrade From ... To</td>
<td>Upgrade Path</td>
<td>For Details</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| SLE HA (Geo) 12 to SLE HA (Geo) 12 SP1 | Cluster Rolling Upgrade | • Base System: *Deployment Guide* for SLES 12 SP1, part *Updating and Upgrading SUSE Linux Enterprise*  
                                 |                                      | • SLE HA: *Performing a Cluster Rolling Upgrade*  
                                 |                                      | • SLE HA Geo: *Geo Clustering Quick Start* for SLE HA 12 SP1, section *Upgrading to the Latest Product Version* |
| SLE HA (Geo) 12 SP1 to SLE HA (Geo) 12 SP2 | Cluster Rolling Upgrade | • Base System: *Deployment Guide* for SLES 12 SP2, part *Updating and Upgrading SUSE Linux Enterprise*  
                                 |                                      | • SLE HA: *Performing a Cluster Rolling Upgrade*  
                                 |                                      | • SLE HA Geo: *Geo Clustering Quick Start* for SLE HA 12 SP2, section *Upgrading to the Latest Product Version*  
                                 |                                      | • DRBD 8 to DRBD 9: *Migrating from DRBD 8 to DRBD 9* |
| SLE HA (Geo) 12 SP2 to SLE HA (Geo) 12 SP3 | Cluster Rolling Upgrade | • Base System: *Deployment Guide* for SLES 12 SP3, part *Updating and Upgrading SUSE Linux Enterprise*  
                                 |                                      | • SLE HA: *Performing a Cluster Rolling Upgrade*  
<pre><code>                             |                                      | • SLE HA Geo: *Geo Clustering Guide* for SLE HA 12 SP3, section *Upgrading to the Latest Product Version* |
</code></pre>
<table>
<thead>
<tr>
<th>Upgrade From ... To</th>
<th>Upgrade Path</th>
<th>For Details</th>
</tr>
</thead>
</table>
| SLE HA (Geo) 12 SP3 to SLE HA (Geo) 12 SP4 | Cluster Rolling Upgrade       | • Base System: Deployment Guide for SLES 12 SP4, part Updating and Upgrading SUSE Linux Enterprise  
• SLE HA: Performing a Cluster Rolling Upgrade  
• SLE HA Geo: Geo Clustering Guide for SLE HA 12 SP4, section Upgrading to the Latest Product Version |
| SLE HA (Geo) 12 SP3 to SLE HA 15      | Cluster Offline Upgrade       | • Base System: Upgrade Guide for SLES 15  
• SLE HA: Upgrading from Product Version 12 to 15: Cluster Offline Upgrade  
• SLE HA Geo: Book “Geo Clustering Guide”, Chapter 10 “Upgrading to the Latest Product Version”  
• Clustered LVM: Online Migration from Mirror LV to Cluster MD |
| SLE HA (Geo) 12 SP4 to SLE HA 15 SP1   | Cluster Offline Upgrade       | • Base System: Upgrade Guide for SLES 15 SP1  
• SLE HA: Upgrading from Product Version 12 to 15: Cluster Offline Upgrade  
• SLE HA Geo: Book “Geo Clustering Guide”, Chapter 10 “Upgrading to the Latest Product Version”  
• Clustered LVM: Online Migration from Mirror LV to Cluster MD |
### Upgrade From ... To

<table>
<thead>
<tr>
<th>Upgrade From ... To</th>
<th>Upgrade Path</th>
<th>For Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLE HA 15 to SLE-HA 15 SP1</td>
<td>Cluster Rolling Upgrade</td>
<td></td>
</tr>
</tbody>
</table>
  - Base System: *Upgrade Guide for SLES 15 SP1*  
  - SLE HA: *Performing a Cluster Rolling Upgrade*  

#### Note: Skipping Service Packs

The easiest upgrade path is consecutively installing all service packs. For the SUSE Linux Enterprise 15 product line (GA and the subsequent service packs) it is also supported to skip one service pack when upgrading. For example, upgrading from SLE HA 15 GA to 15 SP2 or from SLE HA 15 SP1 to 15 SP3 is supported.

### 5.2.2 Required Preparations Before Upgrading

#### Backup

Ensure that your system backup is up to date and restorable.

#### Testing

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.
### 5.2.3 Cluster Offline Upgrade

This section applies to the following scenarios:

- Upgrading from SLE HA 11 SP3 to SLE HA 12—for details see Procedure 5.1, “Upgrading from Product Version 11 to 12: Cluster Offline Upgrade”.
- Upgrading from SLE HA 11 SP4 to SLE HA 12 SP1—for details see Procedure 5.1, “Upgrading from Product Version 11 to 12: Cluster Offline Upgrade”.
- Upgrading from SLE HA 12 SP3 to SLE HA 15—for details see Procedure 5.2, “Upgrading from Product Version 12 to 15: Cluster Offline Upgrade”.
- Upgrading from SLE HA 12 SP4 to SLE HA 15 SP1—for details see Procedure 5.2, “Upgrading from Product Version 12 to 15: Cluster Offline Upgrade”.

If your cluster is still based on an older product version than the ones listed above, first upgrade it to a version of SLES and SLE HA that can be used as a source for upgrading to the desired target version.

**PROCEDURE 5.1: UPGRADING FROM PRODUCT VERSION 11 TO 12: CLUSTER OFFLINE UPGRADE**

The High Availability Extension 12 cluster stack comes with major changes in various components (for example, `/etc/corosync/corosync.conf`, disk formats of OCFS2). Therefore, a cluster rolling upgrade from any SUSE Linux Enterprise High Availability Extension 11 version is not supported. Instead, all cluster nodes must be offline and the cluster needs to be upgraded as a whole as described below.

1. Log in to each cluster node and stop the cluster stack with:
   ```
   root # rcopenais stop
   ```
2. For each cluster node, perform an upgrade to the desired target version of SUSE Linux Enterprise Server and SUSE Linux Enterprise High Availability Extension—see Section 5.2.1, “Supported Upgrade Paths for SLE HA and SLE HA Geo”.
3. After the upgrade process has finished, reboot each node with the upgraded version 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 # o2cluster --update PATH_TO_DEVICE
   ```
It adds additional parameters to the disk. They are needed for the updated OCFS2 version that is shipped with SUSE Linux Enterprise High Availability Extension 12 and 12 SPx.

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 4.1, “Defining the First Communication Channel (Multicast)” or Procedure 4.2, “Defining the First Communication Channel (Unicast)**, respectively.
   
   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. YaST will write them to `/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:

      ```bash
      root # csync2 -xv
      ```

      For details on Csync2, see Section 4.5, “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:

   ```bash
   root # crm cluster start
   ```

7. Check the cluster status with `crm status` or with Hawk2.

8. Configure the following services to start at boot time:

   ```bash
   root # systemctl enable pacemaker
   root # systemctl enable hawk
   root # systemctl enable sbd
   ```

**Note: Upgrading the CIB Syntax Version**

Sometimes new features are only available with the latest CIB syntax version. When you upgrade to a new product version, your CIB syntax version will *not* be upgraded by default.
1. Check your version with:

```
cibadmin -Q | grep validate-with
```

2. Upgrade to the latest CIB syntax version with:

```
root # cibadmin --upgrade --force
```

---

**PROCEDURE 5.2: UPGRADING FROM PRODUCT VERSION 12 TO 15: CLUSTER OFFLINE UPGRADE**

---

**Important: Installation from Scratch**

If you decide to install the cluster nodes from scratch (instead of upgrading them), see Section 2.2, “Software Requirements” for the list of modules required for SUSE Linux Enterprise High Availability Extension 15 SP1. Find more information about modules, extensions and related products in the release notes for SUSE Linux Enterprise Server 15. They are available at https://www.suse.com/releasenotes/.

1. Before starting the offline upgrade to SUSE Linux Enterprise High Availability Extension 15, manually upgrade the CIB syntax in your current cluster as described in Note: Upgrading the CIB Syntax Version.

2. Log in to each cluster node and stop the cluster stack with:

```
root #crm cluster stop
```

3. For each cluster node, perform an upgrade to the desired target version of SUSE Linux Enterprise Server and SUSE Linux Enterprise High Availability Extension—see Section 5.2.1, “Supported Upgrade Paths for SLE HA and SLE HA Geo”.

4. After the upgrade process has finished, log in to each node and boot it with the upgraded version of SUSE Linux Enterprise Server and SUSE Linux Enterprise High Availability Extension.

5. If you use Cluster LVM, you need to migrate from clvmd to lvmlockd. See the man page of `lvmlockd`, section changing a clvm VG to a lockd VG and Section 22.4, “Online Migration from Mirror LV to Cluster MD”.

---
6. Start the cluster stack with:

```
root # crm cluster start
```

7. Check the cluster status with `crm status` or with Hawk2.

5.2.4 Cluster Rolling Upgrade

This section applies to the following scenarios:

- Upgrading from SLE HA 12 to SLE HA 12 SP1
- Upgrading from SLE HA 12 SP1 to SLE HA 12 SP2
- Upgrading from SLE HA 12 SP2 to SLE HA 12 SP3
- Upgrading from SLE HA 12 SP3 to SLE HA 12 SP4
- Upgrading from SLE HA 15 to SLE HA 15 SP1

Use one of the following procedures for your scenario:

- For a more general rolling upgrade, refer to Procedure 5.3.
- For a specific rolling upgrade, refer to Procedure 5.4.

**Warning: Active Cluster Stack**

Before starting an upgrade for a node, stop the cluster stack on that node.

If the cluster resource manager on a node is active during the software update, this can lead to results such as fencing of active nodes.

**Important: Time Limit for Cluster Rolling Upgrade**

The new features shipped with the latest product version will only be available after all cluster nodes have been upgraded to the latest product version. Mixed version clusters are only supported for a short time frame during the cluster rolling upgrade. Complete the cluster rolling upgrade within one week.

Once all the online nodes are running the upgraded version, it is not possible for any other nodes with the old version to (re-)join without having been upgraded.
PROCEDURE 5.3: PERFORMING A CLUSTER ROLLING UPGRADE

1. Log in as root on the node that you want to upgrade and stop the cluster stack:

   ```
   root # crm cluster stop
   ```

2. Perform an upgrade to the desired target version of SUSE Linux Enterprise Server and SUSE Linux Enterprise High Availability Extension. To find the details for the individual upgrade processes, see Section 5.2.1, “Supported Upgrade Paths for SLE HA and SLE HA Geo”.

3. Start the cluster stack on the upgraded node to make the node rejoin the cluster:

   ```
   root # crm cluster start
   ```

4. Take the next node offline and repeat the procedure for that node.

5. Check the cluster status with `crm status` or with Hawk2.

   The Hawk2 Status screen also shows a warning if different CRM versions are detected for your cluster nodes.

---

**Important: Time Limit for Rolling Upgrade**

The new features shipped with the latest product version will only be available after all cluster nodes have been upgraded to the latest product version. Mixed version clusters are only supported for a short time frame during the rolling upgrade. Complete the rolling upgrade within one week.

The Hawk2 Status screen also shows a warning if different CRM versions are detected for your cluster nodes.

Beside an in-place upgrade, many customers prefer a fresh installation even for moving to the next service pack. The following procedure shows a scenario where a two-node cluster with the nodes alice and bob is upgraded to the next service pack (SP):

PROCEDURE 5.4: PERFORMING A CLUSTER-WIDE FRESH INSTALLATION OF A NEW SERVICE PACK

1. Make a backup of your cluster configuration. A minimum set of files are shown in the following list:

   ```
   /etc/corosync/corosync.conf
   /etc/corosync/authkey
   /etc/sysconfig/sbd
   ```
Depending on your resources, you may also need the following files:

/etc/services
/etc/passwd
/etc/shadow
/etc/groups
/etc/drbd/*
/etc/lvm/lvm.conf
/etc/mdadm.conf
/etc/mdadm.SID.conf

2. Start with node alice.

   a. Put the node into standby node. That way, resources can move off the node:

   ```
   root # crm --wait node standby alice reboot
   ```

   With the option `--wait`, the command returns only when the cluster finishes the transition and becomes idle. The `reboot` option has the effect that the node will be already out of standby mode when it is online again. Despite its name, the `reboot` option works as long as the node goes offline and online.

   b. Stop the cluster services on node alice:

   ```
   root # crm cluster stop
   ```

   c. At this point, alice does not have running resources anymore. Upgrade the node alice and reboot it afterward. Cluster services are assumed not to start on boot.

   d. Copy your backup files from Step 1 to the original places.

   e. Bring back node alice into cluster:

   ```
   root # crm cluster start
   ```

   f. Check that resources are fine.

3. Repeat Step 2 for node bob.
5.3 Updating Software Packages on Cluster Nodes

Warning: Active Cluster Stack

Before starting a package update for a node, either *stop* the cluster stack on *that node* or put the node into maintenance mode, depending on whether the cluster stack is affected or not. See Step 1 for details.

If the cluster resource manager on a node is active during the software update, this can lead to results such as fencing of active nodes.

1. 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? If *yes*: Stop the cluster stack on the node before starting the software update:

     ```
     root #crm cluster stop
     ```

   - Does the package update require a reboot? If *yes*: Stop the cluster stack on the node before starting the software update:

     ```
     root #crm cluster stop
     ```

   - If none of the situations above apply, you do not need to stop the cluster stack. In that case, put the node into maintenance mode before starting the software update:

     ```
     root #crm node maintenance NODE_NAME
     ```

     For more details on maintenance mode, see Section 17.2, “Different Options for Maintenance Tasks”.

2. Install the package update using either YaST or Zypper.
3. After the update has been successfully installed:

   - Either start the cluster stack on the respective node (if you stopped it in Step 1):

     ```
     root # crm cluster start
     ```

   - or remove the maintenance flag to bring the node back to normal mode:

     ```
     root # crm node ready NODE_NAME
     ```

4. Check the cluster status with `crm status` or with Hawk2.

5.4 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/](https://www.suse.com/releasenotes/).
II Configuration and Administration

6 Configuration and Administration Basics 57
7 Configuring and Managing Cluster Resources with Hawk2 94
8 Configuring and Managing Cluster Resources (Command Line) 147
9 Adding or Modifying Resource Agents 181
10 Fencing and STONITH 185
11 Storage Protection and SBD 197
12 QDevice and QNetd 218
13 Access Control Lists 225
14 Network Device Bonding 235
15 Load Balancing 240
16 Geo Clusters (Multi-Site Clusters) 254
17 Executing Maintenance Tasks 255
6  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 only 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.

6.1  Use Case Scenarios

In general, clusters fall into one of two categories:

- Two-node clusters
- Clusters with more than two nodes. This usually means an odd number of nodes.

Adding also different topologies, different use cases can be derived. The following use cases are the most common:

Two-node cluster in one location

Configuration: FC SAN or similar shared storage, layer 2 network.

Usage scenario: Embedded clusters that focus on service high availability and not data redundancy for data replication. Such a setup is used for radio stations or assembly line controllers, for example.

Two-node clusters in two locations (most widely used)

Configuration: Symmetrical stretched cluster, FC SAN, and layer 2 network all across two locations.

Usage scenario: Classic stretched clusters, focus on high availability of services and local data redundancy. For databases and enterprise resource planning. One of the most popular setups during the last few years.
Odd number of nodes in three locations

**Configuration:** $2 \times N + 1$ nodes, FC SAN across two main locations. Auxiliary third site with no FC SAN, but acts as a majority maker. Layer 2 network at least across two main locations.

**Usage scenario:** Classic stretched cluster, focus on high availability of services and data redundancy. For example, databases, enterprise resource planning.

### 6.2 Quorum Determination

Whenever communication fails between one or more nodes and the rest of the cluster, a cluster partition occurs. The nodes can only communicate with other nodes in the same partition and are unaware of the separated nodes. A cluster partition is defined as having quorum (being “quorate”) if it has the majority of nodes (or votes). How this is achieved is done by *quorum calculation*. Quorum is a requirement for fencing.

Quorum calculation has changed between SUSE Linux Enterprise High Availability Extension 11 and SUSE Linux Enterprise High Availability Extension 15. For SUSE Linux Enterprise High Availability Extension 11, quorum was calculated by Pacemaker. Starting with SUSE Linux Enterprise High Availability Extension 12, Corosync can handle quorum for two-node clusters directly without changing the Pacemaker configuration.

How quorum is calculated is influenced by the following factors:

**Number of Cluster Nodes**

To keep services running, a cluster with more than two nodes relies on quorum (majority vote) to resolve cluster partitions. Based on the following formula, you can calculate the minimum number of operational nodes required for the cluster to function:

$$N \geq C/2 + 1$$

- $N$ = minimum number of operational nodes
- $C$ = number of cluster nodes

For example, a five-node cluster needs a minimum of three operational nodes (or two nodes which can fail).

We strongly recommend to use either a two-node cluster or an odd number of cluster nodes. Two-node clusters make sense for stretched setups across two sites. Clusters with an odd number of nodes can either be built on one single site or might be spread across three sites.

**Corosync Configuration**
Corosync is a messaging and membership layer, see Section 6.2.4, “Corosync Configuration for Two-Node Clusters” and Section 6.2.5, “Corosync Configuration for N-Node Clusters”.

6.2.1 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 Hawk2 and the crm shell.

The predefined values can usually be kept. However, to make key functions of your cluster work correctly, you need to adjust the following parameters after basic cluster setup:

- Global Option no-quorum-policy
- Global Option stonith-enabled

6.2.2 Global 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**
  
  Setting no-quorum-policy to ignore makes the cluster behave like it has quorum. Resource management is continued.
  
  On SLES 11 this was the recommended setting for a two-node cluster. Starting with SLES 12, this option is obsolete. Based on configuration and conditions, Corosync gives cluster nodes or a single node “quorum”—or not.
  
  For two-node clusters the only meaningful behavior is to always react in case of quorum loss. The first step should always be to try to fence the lost node.

- **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. This option works only in combination with SBD, see Chapter 11, Storage Protection and SBD.

### 6.2.3 Global Option stonith-enabled

This global option defines whether 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 lvmlockd, GFS2, and OCFS2) will fail to start.

⚠️ Important: No Support Without STONITH

A cluster without STONITH is not supported.

### 6.2.4 Corosync Configuration for Two-Node Clusters

When using the bootstrap scripts, the Corosync configuration contains a quorum section with the following options:

**EXAMPLE 6.1: EXCERPT OF COROSYNC CONFIGURATION FOR A TWO-NODE CLUSTER**

```bash
quorum {
    # Enable and configure quorum subsystem (default: off)
}
```
As opposed to SUSE Linux Enterprise 11, the votequorum subsystem in SUSE Linux Enterprise 12 and later is powered by Corosync version 2.x. This means that the `no-quorum-policy=ignore` option must not be used.

By default, when `two_node: 1` is set, the `wait_for_all` option is automatically enabled. If `wait_for_all` is not enabled, the cluster should be started on both nodes in parallel. Otherwise the first node will perform a startup-fencing on the missing second node.

6.2.5  Corosync Configuration for N-Node Clusters

When not using a two-node cluster, we strongly recommend an odd number of nodes for your N-node cluster. With regards to quorum configuration, you have the following options:

- Adding additional nodes with the `ha-cluster-join` command, or
- Adapting the Corosync configuration manually.

If you adjust `/etc/corosync/corosync.conf` manually, use the following settings:

**EXAMPLE 6.2: EXCERPT OF COROSYNC CONFIGURATION FOR AN N-NODE CLUSTER**

```plaintext
quorum {
    provider: corosync_votequorum ① 
    expected_votes: N ② 
    wait_for_all: 1 ③
}
```

① Use the quorum service from Corosync
② The number of votes to expect. This parameter can either be provided inside the `quorum` section, or is automatically calculated when the `nodelist` section is available.
③ Enables the wait for all (WFA) feature. When WFA is enabled, the cluster will be quorate for the first time only after all nodes have become visible. To avoid some startup race conditions, setting `wait_for_all` to 1 may help. For example, in a five-node cluster every node has one vote and thus, `expected_votes` is set to 5. As soon as three or more nodes are visible to each other, the cluster partition becomes quorate and can start operating.
6.3 **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.

### 6.3.1 Resource Management

Before you can use a resource in the cluster, it must be set up. For example, 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 testing or maintenance tasks after the services are already running under cluster control, make sure to put the resources, nodes, or the whole cluster into maintenance mode before you touch any of them manually. For details, see Section 17.2, “Different Options for Maintenance Tasks”.

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:

- **Hawk2**: Chapter 7, Configuring and Managing Cluster Resources with Hawk2
- **crmsh**: Chapter 8, Configuring and Managing Cluster Resources (Command Line)
Important: Resource IDs and Node Names

Cluster resources and cluster nodes should be named differently. Otherwise Hawk2 will fail.

6.3.2 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 promotable clone 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. OCF specifications have strict definitions of which exit codes must be returned by actions, see Section 9.3, “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, 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 8.1.3, “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/iniscrptact.html](http://refspecs.linuxbase.org/LSB_4.1.0/LSB-Core-generic/LSB-Core-generic/iniscrptact.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](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 an 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 6.6.1, “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 10, Fencing and STONITH](#).

The agents supplied with the High Availability Extension are written to OCF specifications.
6.3.3 Types of Resources

The following types of resources can be created:

Primitives

A primitive resource, the most basic type of resource.

Learn how to create primitive resources with your preferred cluster management tool:

- Hawk2: Procedure 7.5, “Adding a Primitive Resource”
- crmsh: Section 8.3.2, “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 6.3.5.1, “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 6.3.5.2, “Clones”.

Promotable clones (also known as multi-state resources) are a special type of clone resources that can be promoted.

6.3.4 Resource Templates

If you want to create lots of resources with similar configurations, defining a resource template is the easiest way. After having been defined, it can be referenced in primitives—or in certain types of constraints, as described in Section 6.5.3, “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:

- Hawk2: Procedure 7.6, “Adding a Resource Template”
- crmsh: Section 8.3.3, “Creating Resource Templates”
6.3.5 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 promotable clone resources.

6.3.5.1 Groups

Some cluster resources depend on other components or resources. They 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 cluster resource groups.

**EXAMPLE 6.3: 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 resource that is combined into a cluster resource group. The resource group would run on one or more servers. In case of a software or hardware malfunction, the group would fail over to another server in the cluster, similar to an individual cluster 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), 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:

- Hawk2: Procedure 7.9, “Adding a Resource Group”
- crmsh: Section 8.3.10, “Configuring a Cluster Resource Group”

6.3.5.2 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 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.

**Promotable Clones (Multi-state Resources)**

Active instances of these resources are divided into two states, active and passive. These are also sometimes called primary and secondary. Promotable clones can be either anonymous or globally unique. See also Section 6.3.5.3, “Promotable Clones (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*, available from http://www.clusterlabs.org/pacemaker/doc/. Refer to section *Clones - Resources That Get Active on Multiple Hosts*.

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

- **Hawk2**: Procedure 7.10, “Adding a Clone Resource”
- **crmsh**: Section 8.3.11, “Configuring a Clone Resource”.

### 6.3.5.3 Promotable Clones (Multi-state Resources)

Promotable clones (formerly known as multi-state resources) are a specialization of clones. They allow the instances to be in one of two operating modes (primary or secondary). Promotable clones must contain exactly one group or one regular resource.

When configuring resource monitoring or constraints, promotable clones have different requirements than simple resources. For details, see *Pacemaker Explained*. The version for Pacemaker 1.1 is available from http://www.clusterlabs.org/pacemaker/doc/. Refer to section *Multi-state - Resources That Have Multiple Modes*. 

6.3.6 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 Hawk2 as described in Procedure 7.5, “Adding a Primitive Resource”.

**TABLE 6.1: OPTIONS FOR A PRIMITIVE RESOURCE**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority</td>
<td>If not all resources can be active, the cluster will stop lower priority</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>resources to keep higher priority ones active.</td>
<td></td>
</tr>
<tr>
<td>target-role</td>
<td>In what state should the cluster attempt to keep this resource? Allowed</td>
<td>started</td>
</tr>
<tr>
<td></td>
<td>values: stopped, started, master.</td>
<td></td>
</tr>
<tr>
<td>is-managed</td>
<td>Is the cluster allowed to start and stop the resource? Allowed values:</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td>true, false. If the value is set to false, the status of the resource is</td>
<td></td>
</tr>
<tr>
<td></td>
<td>still monitored and any failures are reported. This is different from setting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a resource to maintenance=&quot;true&quot;.</td>
<td></td>
</tr>
<tr>
<td>maintenance</td>
<td>Can the resources be touched manually? Allowed values: true, false. If set to</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>true, false. If set to true, all resources become unmanaged: the cluster</td>
<td></td>
</tr>
<tr>
<td></td>
<td>will stop monitoring them and thus be</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>oblivious</td>
<td>oblivious about their status. You can stop or restart cluster resources at will, without the cluster attempting to restart them.</td>
<td></td>
</tr>
<tr>
<td>resource-stickiness</td>
<td>How much does the resource prefer to stay where it is?</td>
<td>calculated</td>
</tr>
<tr>
<td>migration-threshold</td>
<td>How many failures should occur for this resource on a node before making the node ineligible to host this resource?</td>
<td>INFINITY (disabled)</td>
</tr>
<tr>
<td>multiple-active</td>
<td>What should the cluster do if it ever finds the resource active on more than one node? Allowed values: block (mark the resource as unmanaged), stop_only, stop_start.</td>
<td>stop_start</td>
</tr>
<tr>
<td>failure-timeout</td>
<td>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)?</td>
<td>0 (disabled)</td>
</tr>
<tr>
<td>allow-migrate</td>
<td>Allow resource migration for resources which support migrate_to/migrate_from actions.</td>
<td>false</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>remote-node</td>
<td>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 also be assumed as the host name to connect to at remote-port.</td>
<td>none (disabled)</td>
</tr>
<tr>
<td><img src="image" alt="Warning: Use Unique IDs" /></td>
<td>This value must not overlap with any existing resource or node IDs.</td>
<td></td>
</tr>
<tr>
<td>remote-port</td>
<td>Custom port for the guest connection to pacemaker_remote.</td>
<td>3121</td>
</tr>
<tr>
<td>remote-addr</td>
<td>The IP address or host name to connect to if the remote node’s name is not the host name of the guest.</td>
<td>remote-node (value used as host name)</td>
</tr>
<tr>
<td>remote-connect-timeout</td>
<td>How long before a pending guest connection will time out.</td>
<td>60s</td>
</tr>
</tbody>
</table>
6.3.7 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 Hawk2 as described in Procedure 7.5, “Adding a Primitive Resource”. In the `crm` command line utility and in Hawk2, 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 Promotable Clones

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

6.3.8 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 6.4, “Resource Monitoring”.

TABLE 6.2: RESOURCE OPERATION PROPERTIES

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Your name for the action. Must be unique. (The ID is not shown).</td>
</tr>
<tr>
<td>name</td>
<td>The action to perform. Common values: monitor, start, stop.</td>
</tr>
<tr>
<td>interval</td>
<td>How frequently to perform the operation. Unit: seconds</td>
</tr>
<tr>
<td>timeout</td>
<td>How long to wait before declaring the action has failed.</td>
</tr>
<tr>
<td>requires</td>
<td>What conditions need to be satisfied before this action occurs. Allowed values: nothing, quorum, fencing. The default depends on whether fencing is enabled and if the resource's class is stonith. For STONITH resources, the default is nothing.</td>
</tr>
<tr>
<td>Operation</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>on-fail</td>
<td>The action to take if this action ever fails. Allowed values:</td>
</tr>
<tr>
<td></td>
<td>• ignore: Pretend the resource did not fail.</td>
</tr>
<tr>
<td></td>
<td>• block: Do not perform any further operations on the resource.</td>
</tr>
<tr>
<td></td>
<td>• stop: Stop the resource and do not start it elsewhere.</td>
</tr>
<tr>
<td></td>
<td>• restart: Stop the resource and start it again (possibly on a different node).</td>
</tr>
<tr>
<td></td>
<td>• fence: Bring down the node on which the resource failed (STONITH).</td>
</tr>
<tr>
<td></td>
<td>• standby: Move all resources away from the node on which the resource failed.</td>
</tr>
<tr>
<td>enabled</td>
<td>If false, the operation is treated as if it does not exist. Allowed values: true, false.</td>
</tr>
<tr>
<td>role</td>
<td>Run the operation only if the resource has this role.</td>
</tr>
<tr>
<td>record-pending</td>
<td>Can be set either globally or for individual resources. Makes the CIB reflect the state of “in-flight” operations on resources.</td>
</tr>
<tr>
<td>description</td>
<td>Description of the operation.</td>
</tr>
</tbody>
</table>
6.3.9 **Timeout Values**

Timeout 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.

You can adjust the global default for operations and set any specific timeout values with both `crmsh` and Hawk2. The best practice for determining and setting timeout values is as follows:

**PROCEDURE 6.1: 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 Hawk2, it will provide useful timeout proposals for those operations.
6.4 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 (Hawk2, `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, 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.

Monitoring Stopped Resources

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 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` when it is in role="Stopped". When running, it will be monitored every 30 seconds.

Probing

The CRM executes an initial monitoring for each resource on every node, the so-called `probe`. A probe is also executed after the cleanup of a resource. 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, 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 monitoring operation for the **probing** of this resource. Do so by adding a monitoring operation with the `interval` set to 0, for example:

```bash
crm(live)configure# primitive rsc1 ocf:pacemaker:Dummy \
   op monitor interval="0" timeout="60"
```

The probe of **rsc1** will time out in **60s**, independent of the global timeout defined in `op_defaults`, or any other operation timeouts configured. If you did not set `interval="0"` for specifying the probing of the respective resource, the CRM will automatically check for any other monitoring operations defined for that resource and will calculate the timeout value for probing as described above.

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

- **Hawk2**: *Procedure 7.13, “Adding and Modifying an Operation”*
- **crmsh**: *Section 8.3.9, “Configuring Resource Monitoring”*

### 6.5 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.

#### 6.5.1 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.
Important: Restrictions for Constraints and Certain Types of Resources

- Do not create colocation constraints for members of a resource group. Create a colocation constraint pointing to the resource group as a whole instead. All other types of constraints are safe to use for members of a resource group.

- Do not use any constraints on a resource that has a clone resource or a promotable clone resource applied to it. The constraints must apply to the clone or promotable clone resource, not to the child resource.

6.5.1.1 Resource Sets

6.5.1.1.1 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. The latter caused a constraint explosion as the number of resources and combinations grew. The configuration via resource sets is not necessarily less verbose, but is easier to understand and maintain, as the following examples show.

EXAMPLE 6.4: 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, alice:

```plaintext
crm(live)configure# primitive vip1 IPaddr2 params ip=192.168.1.5
crm(live)configure# primitive vip2 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 6.5: A CHAIN OF COLOCATED RESOURCES

```plaintext
<constraints>
```
The same configuration expressed by a resource set:

```xml
<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 6.6: A CHAIN OF ORDERED RESOURCES**

```xml
<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 6.7: A CHAIN OF ORDERED RESOURCES EXPRESSED AS RESOURCE SET**

```xml
<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 (\texttt{sequential=true}) or unordered (\texttt{sequential=false}). Furthermore, the \texttt{require-all} attribute can be used to switch between \texttt{AND} and \texttt{OR} logic.

### 6.5.1.1.2 Resource Sets for Colocation Constraints Without Dependencies

Sometimes it is useful 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 \textit{not} want the cluster to restart the other one if one of them fails. This can be achieved on the crm shell by using the \texttt{weak bond} command.

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

- \texttt{crmsh}: \textit{Section 8.3.5.3, “Collocating Sets for Resources Without Dependency”}

### 6.5.1.2 For More Information

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

- \texttt{Hawk2}: \textit{Section 7.6, “Configuring Constraints”}
- \texttt{crmsh}: \textit{Section 8.3.5, “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 \url{http://www.clusterlabs.org/pacemaker/doc/}:

- \textit{Pacemaker Explained}, chapter \textit{Resource Constraints}
- \textit{Colocation Explained}
- \textit{Ordering Explained}

### 6.5.2 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.

\text{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.

6.5.3 Resource Templates and Constraints

If you have defined a resource template (see Section 6.3.4, “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 7.17, “Using a Resource Set for Constraints”.
6.5.4 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 several 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:

- **Hawk2**: Section 7.6.6, “Specifying Resource Failover Nodes”
- **crmsh**: Section 8.3.6, “Specifying Resource Failover Nodes”

**EXAMPLE 6.8: 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 \( \geq \) migration-threshold then the resource is migrated to the node with the next best preference.

After 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. It would be allowed 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.
For details on using migration thresholds and resetting failcounts with your preferred cluster management tool:

- **Hawk2**: Section 7.6.6, “Specifying Resource Failover Nodes”
- **crmsh**: Section 8.3.6, “Specifying Resource Failover Nodes”

### 6.5.5 Failback Nodes

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

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.
6.5.6 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:

- **Hawk2**: Section 7.6.8, “Configuring Placement of Resources Based on Load Impact”
- **crmsh**: Section 8.3.8, “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 capacities is completely irrelevant for the High Availability Extension; it only 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: `ocf: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.

**Note: Different Resource Agents for Xen and libvirt**

The `ocf:heartbeat:Xen` resource agent should not be used with libvirt, as libvirt expects to be able to modify the machine description file. For libvirt, use the `ocf:heartbeat:VirtualDomain` resource agent.

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: `ocf: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. 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.
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 (to achieve power savings on the remaining nodes).

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. Ensure that resource priorities are properly set so that your most important resources are scheduled first.

**EXAMPLE 6.9: EXAMPLE CONFIGURATION FOR LOAD-BALANCED PLACING**

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

```plaintext
node alice utilization memory="4000"
node bob utilization memory="4000"
node charlie utilization memory="4000"
primitive xenA Xen utilization hv_memory="3500" \
   params xmfile="/etc/xen/shared-vm/vm1" \
   meta priority="10"
primitive xenB Xen utilization hv_memory="2000" \
   params xmfile="/etc/xen/shared-vm/vm2" \
   meta priority="1"
primitive xenC Xen utilization hv_memory="2000" \
   params xmfile="/etc/xen/shared-vm/vm3" \
   meta priority="1"
primitive xenD Xen utilization hv_memory="1000" \
   params xmfile="/etc/xen/shared-vm/vm4" \
   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. \texttt{xenA} would be ensured to be allocated, as would \texttt{xenD}. However, only one of the remaining resources \texttt{xenB} or \texttt{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.

### 6.5.7 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 several resources related to a database, create a tag called \texttt{databases} 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 \texttt{loc-db-prefer} applies to the set of resources tagged with \texttt{databases}:

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

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

- \textbf{Hawk2:} Procedure 7.12, “Adding a Tag”
- \textbf{crmsh:} Section 8.4.7, “Grouping/Tagging Resources”

### 6.6 Managing Services on Remote Hosts

The possibilities for monitoring and managing services on remote hosts has become increasingly important during the last few years. 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 \texttt{pacemaker_remote} service now allows SUSE Linux Enterprise High Availability Extension 15 SP1 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.
6.6.1 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 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 6.10, “Configuring Resources for Monitoring Plug-ins” for a configuration example. Alternatively, Nagios resource agents can also be configured as ordinary resources to use them for monitoring hosts or services via the network.

**EXAMPLE 6.10: CONFIGURING RESOURCES FOR MONITORING PLUG-INS**

```bash
primitive vml VirtualDomain \  
  params hypervisor=qemu:///system config=/etc/libvirt/qemu/vml.xml \  
  op start interval="0" timeout="90" \  
  op stop interval="0" timeout="90" \  
  op monitor interval="10" timeout="30"
primitive vml-sshd nagios:check_tcp \  
  params hostname="vml" port="22" \  
  op start interval="0" timeout="120" \  
```

89 Monitoring Services on Remote Hosts with Monitoring Plug-ins SLE HA 15 SP1
The supported parameters are the 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 host name or an IP address.

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.

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 host names 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:

```bash
group g-vm1-and-services vm1 vm1-sshd 
  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.

### 6.6.2 Managing Services on Remote Nodes with pacemaker_remote

With the `pacemaker_remote` service, High Availability clusters can be extended to virtual nodes or remote bare-metal machines. They 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), plus 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” (bare-metal) 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 by the scalability limits (Corosync has a member limit of 32 nodes).

Find more information about the `remote_pacemaker` service, including multiple use cases with detailed setup instructions in Article “Pacemaker Remote Quick Start”.

### 6.7 Monitoring System Health

To prevent a node from running out of disk space and thus being unable to 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 regard 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 6.2: 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:

```bash
primitive sysinfo ocf:pacemaker:SysInfo \"...
```
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.

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.

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 `pacemaker-schedulerd` 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 anymore, 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 the cluster services on that node.

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

### 6.8 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 several 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/. Find the tutorial at http://crmsh.github.io/start-guide/.

http://clusterlabs.org/

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

http://www.clusterlabs.org/pacemaker/doc/

Holds several comprehensive manuals and some shorter documents explaining general concepts. For example:

- *Pacemaker Explained*: Contains comprehensive and very detailed information for reference.

- *Colocation Explained*

- *Ordering Explained*
7 Configuring and Managing Cluster Resources with Hawk2

To configure and manage cluster resources, either use Hawk2, or the crm shell (crmsh) command line utility. If you upgrade from an earlier version of SUSE® Linux Enterprise High Availability Extension where Hawk was installed, the package will be replaced with the current version, Hawk2.

Hawk2's user-friendly Web interface allows you to monitor and administer your High Availability clusters from Linux or non-Linux machines alike. Hawk2 can be accessed from any machine inside or outside of the cluster by using a (graphical) Web browser.

7.1 Hawk2 Requirements

Before users can log in to Hawk2, the following requirements need to be fulfilled:

**hawk2 Package**
The **hawk2** package must be installed on all cluster nodes you want to connect to with Hawk2.

**Web Browser**
On the machine from which to access a cluster node using Hawk2, you need a (graphical) Web browser (with JavaScript and cookies enabled) to establish the connection.

**Hawk2 Service**
To use Hawk2, the respective Web service must be started on the node that you want to connect to via the Web interface. See Procedure 7.1, “Starting Hawk2 Services”.
If you have set up your cluster with the scripts from the **ha-cluster-bootstrap** package, the Hawk2 service is already enabled.

**Username, Group and Password on Each Cluster Node**
Hawk2 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 `ha-cluster` user. Before starting Hawk2, change it to a secure password. If you did not use the `ha-cluster-init` script, either set a password for the `hacluster` first or create a new user which is a member of the `haclient` group. Do this on every node you will connect to with Hawk2.

**Wildcard certificate handling**

A wildcard certificate is a public key certificate which is valid for multiple sub-domains. For example, a wildcard certificate for `*.example.com` secures the domains `www.example.com`, `login.example.com`, and possibly more.

Hawk2 supports wildcard certificates as well as conventional certificates. A self-signed default private key and certificate is generated by `/srv/www/hawk/bin/generate-ssl-cert`

To use your own certificate (conventional or wildcard), replace the generated certificate at `/etc/ssl/certs/hawk.pem` with your own.

**PROCEDURE 7.1: STARTING HAWK2 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

   ```bash
   root # systemctl status hawk
   ```

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

   ```bash
   root # systemctl start hawk
   ```

   If you want Hawk2 to start automatically at boot time, execute the following command:

   ```bash
   root # systemctl enable hawk
   ```

**7.2 Logging In**

The Hawk2 Web interface uses the HTTPS protocol and port `7630`.

Instead of logging in to an individual cluster node with Hawk2, you can configure a floating, virtual IP address (`IPaddr` or `IPaddr2`) as a cluster resource. It does not need any special configuration. It allows clients to connect to the Hawk service no matter which physical node the service is running on.
When setting up the cluster with the `ha-cluster-bootstrap` scripts, you will be asked whether to configure a virtual IP for cluster administration.

PROCEDURE 7.2: LOGGING IN TO THE HAWK2 WEB INTERFACE

1. On any machine, start a Web browser and enter the following URL:

   ```
   https://HAWKSERVER:7630/
   ```

   Replace `HAWKSERVER` with the IP address or host name of any cluster node running the Hawk Web service. If a virtual IP address has been configured for cluster administration with Hawk2, replace `HAWKSERVER` with the virtual IP address.

   **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.

   To verify the certificate, ask your cluster operator for the certificate details.

   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.

2. On the Hawk2 login screen, enter the `Username` and `Password` of the `hacluster` user (or of any other user that is a member of the `haclient` group).

3. Click Log In.

### 7.3 Hawk2 Overview: Main Elements

After logging in to Hawk2, you will see a navigation bar on the left-hand side and a top-level row with several links on the right-hand side.

**Note: Available Functions in Hawk2**

By default, users logged in as `root` or `hacluster` have full read-write access to all cluster configuration tasks. However, `Access Control Lists` (ACLs) can be used to define fine-grained access permissions.
If ACLs are enabled in the CRM, the available functions in Hawk2 depend on the user role and their assigned access permissions. The *History Explorer* in Hawk2 can only be executed by the user `hacluster`.

### 7.3.1 Left Navigation Bar

#### Monitoring

- **Status**: Displays the current cluster status at a glance (similar to `crm status` on the crmsh). For details, see *Section 7.8.1, “Monitoring a Single Cluster”*. If your cluster includes guest nodes (nodes that run the `pacemaker_remote` daemon), they are displayed, too. The screen refreshes in near real-time: any status changes for nodes or resources are visible almost immediately.

- **Dashboard**: Allows you to monitor multiple clusters (also located on different sites, in case you have a Geo cluster setup). For details, see *Section 7.8.2, “Monitoring Multiple Clusters”*. If your cluster includes guest nodes (nodes that run the `pacemaker_remote` daemon), they are displayed, too. The screen refreshes in near real-time: any status changes for nodes or resources are visible almost immediately.

#### Troubleshooting

- **History**: Opens the *History Explorer* from which you can generate cluster reports. For details, see *Section 7.10, “Viewing the Cluster History”*.

- **Command Log**: Lists the crmsh commands recently executed by Hawk2.

#### Configuration

- **Add Resource**: Opens the resource configuration screen. For details, see *Section 7.5, “Configuring Cluster Resources”*.

- **Add Constraint**: Opens the constraint configuration screen. For details, see *Section 7.6, “Configuring Constraints”*.

- **Wizards**: Allows you to select from several wizards that guide you through the creation of resources for a certain workload, for example, a DRBD block device. For details, see *Section 7.5.2, “Adding Resources with the Wizard”*. 
• **Edit Configuration**: Allows you to edit resources, constraints, node names and attributes, tags, alerts ([http://crmsh.github.io/man/#cmdhelp_configure_alert](http://crmsh.github.io/man/#cmdhelp_configure_alert)), and fencing topologies ([http://crmsh.github.io/man/#cmdhelp_configure_fencing_topology](http://crmsh.github.io/man/#cmdhelp_configure_fencing_topology)).

• **Cluster Configuration**: Allows you to modify global cluster options and resource and operation defaults. For details, see *Section 7.4, “Configuring Global Cluster Options”*.  

• **Access Control ➤ Roles**: Opens a screen where you can create roles for access control lists (sets of rules describing access rights to the CIB). For details, see *Procedure 13.2, “Adding a Monitor Role with Hawk2”*.  

• **Access Control ➤ Targets**: Opens a screen where you can create targets (system users) for access control lists and assign roles to them. For details, see *Procedure 13.3, “Assigning a Role to a Target with Hawk2”*.  

### 7.3.2 Top-Level Row

Hawk2’s top-level row shows the following entries:

• **Batch**: Click to switch to batch mode. This allows you to simulate and stage changes and to apply them as a single transaction. For details, see *Section 7.9, “Using the Batch Mode”*.  

• **USERNAME**: Allows you to set preferences for Hawk2 (for example, the language for the Web interface, or whether to display a warning if STONITH is disabled).  

• **Help**: Access the SUSE Linux Enterprise High Availability Extension documentation, read the release notes or report a bug.  

• **Logout**: Click to log out.
7.4 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 Hawk2 and crmsh. The predefined values can usually be kept. However, to ensure the key functions of your cluster work correctly, you need to adjust the following parameters after basic cluster setup:

- **Global Option no-quorum-policy**
- **Global Option stonith-enabled**

**PROCEDURE 7.3: MODIFYING GLOBAL CLUSTER OPTIONS**

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. From the left navigation bar, select Configuration › Cluster Configuration. The Cluster Configuration screen opens. It displays the global cluster options and their current values. To display a short description of the parameter on the right-hand side of the screen, hover your mouse over a parameter.

   ![FIGURE 7.1: HAWK2—CLUSTER CONFIGURATION](image-url)
3. Check the values for `no-quorum-policy` and `stonith-enabled` and adjust them, if necessary:

   a. Set `no-quorum-policy` to the appropriate value. See Section 6.2.2, “Global Option no-quorum-policy” for more details.

   b. If you need to disable fencing for any reason, set `stonith-enabled` to `no`. By default, it is set to `true`, because using STONITH devices is necessary for normal cluster operation. According to the default value, the cluster will refuse to start any resources if no STONITH resources have been configured.

   ![Important: No Support Without STONITH](image)

   - You must have a node fencing mechanism for your cluster.
   - The global cluster options `stonith-enabled` and `startup-fencing` must be set to `true`. When you change them, you lose support.

   c. To remove a parameter from the cluster configuration, click the Minus icon next to the parameter. If a parameter is deleted, the cluster will behave as if that parameter had the default value.

   d. To add a new parameter to the cluster configuration, choose one from the drop-down box.

4. If you need to change Resource Defaults or Operation Defaults, proceed as follows:

   a. To adjust a value, either select a different value from the drop-down box or edit the value directly.

   b. To add a new resource default or operation default, choose one from the empty drop-down box and enter a value. If there are default values, Hawk2 proposes them automatically.

   c. To remove a parameter, click the Minus icon next to it. If no values are specified for Resource Defaults and Operation Defaults, the cluster uses the default values that are documented in Section 6.3.6, “Resource Options (Meta Attributes)” and Section 6.3.8, “Resource Operations”.

5. Confirm your changes.
7.5 Configuring Cluster Resources

A cluster administrator needs to create cluster resources for every resource or application that runs on the 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 6.3.3, “Types of Resources”. After you have specified the resource basics (ID, class, provider, and type), Hawk2 shows the following categories:

Parameters (Instance Attributes)
Determines which instance of a service the resource controls. For more information, refer to Section 6.3.7, “Instance Attributes (Parameters)”. When creating a resource, Hawk2 automatically shows any required parameters. Edit them to get a valid resource configuration.

Operations
Needed for resource monitoring. For more information, refer to Section 6.3.8, “Resource Operations”. When creating a resource, Hawk2 displays the most important resource operations (monitor, start, and stop).

Meta Attributes
Tells the CRM how to treat a specific resource. For more information, refer to Section 6.3.6, “Resource Options (Meta Attributes)”. When creating a resource, Hawk2 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).

Utilization
Tells the CRM what capacity a certain resource requires from a node. For more information, refer to Section 7.6.8, “Configuring Placement of Resources Based on Load Impact”.

You can adjust the entries and values in those categories either during resource creation or later.
7.5.1 Showing the Current Cluster Configuration (CIB)

Sometimes a cluster administrator needs to know the cluster configuration. Hawk2 can show the current configuration in crm shell syntax, as XML and as a graph. To view the cluster configuration in crm shell syntax, from the left navigation bar select Configuration > Edit Configuration and click Show. To show the configuration in raw XML instead, click XML. Click Graph for a graphical representation of the nodes and resources configured in the CIB. It also shows the relationships between resources.

7.5.2 Adding Resources with the Wizard

The Hawk2 wizard is a convenient way of setting up simple resources like a virtual IP address or an SBD STONITH resource, for example. It is also useful for complex configurations that include multiple resources, like the resource configuration for a DRBD block device or an Apache Web server. The wizard guides you through the configuration steps and provides information about the parameters you need to enter.

PROCEDURE 7.4: USING THE RESOURCE WIZARD

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. From the left navigation bar, select Configuration > Wizards.

3. Expand the individual categories by clicking the arrow down icon next to them and select the desired wizard.

4. Follow the instructions on the screen. After the last configuration step, Verify the values you have entered. Hawk2 shows which actions it is going to perform and what the configuration looks like. Depending on the configuration, you might be prompted for the root password before you can Apply the configuration.
7.5.3 Adding Simple Resources

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

PROCEDURE 7.5: ADDING A PRIMITIVE RESOURCE

1. Log in to Hawk2:
   
   \[ https://HAWKSERVER:7630/ \]

2. From the left navigation bar, select Configuration > Add Resource > Primitive.

3. Enter a unique Resource ID.

4. In case a resource template exists on which you want to base the resource configuration, select the respective Template. For details about configuring templates, see Procedure 7.6, “Adding a Resource Template”.

5. Select the resource agent Class you want to use: lsb, ocf, service, stonith, or systemd. For more information, see Section 6.3.2, “Supported Resource Agent Classes”.

6. 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.
7. 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. With that, you have specified the resource basics.

**Note**

The selection you get in the *Type* list depends on the *Class* (and for OCF resources also on the *Provider*) you have chosen.

---

8. To keep the *Parameters*, *Operations*, and *Meta Attributes* as suggested by Hawk2, click *Create* to finish the configuration. A message at the top of the screen shows if the action has been successful.

To adjust the parameters, operations, or meta attributes, refer to *Section 7.5.5, “Modifying Resources”*. To configure *Utilization* attributes for the resource, see *Procedure 7.21, “Configuring the Capacity a Resource Requires”*. 

---

**FIGURE 7.3: HAWK2—PRIMITIVE RESOURCE**
7.5.4 Adding Resource Templates

To create lots of resources with similar configurations, defining a resource template is the easiest way. After being 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 6.5.3, “Resource Templates and Constraints”.

PROCEDURE 7.6: ADDING A RESOURCE TEMPLATE

Resource templates are configured like primitive resources.

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. From the left navigation bar, select Configuration > Add Resource > Template.

3. Enter a unique Resource ID.


7.5.5 Modifying Resources

If you have created a resource, you can edit its configuration at any time by adjusting parameters, operations, or meta attributes as needed.

PROCEDURE 7.7: MODIFYING PARAMETERS, OPERATIONS, OR META ATTRIBUTES FOR A RESOURCE

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. On the Hawk2 Status screen, go to the Resources list.

3. In the Operations column, click the arrow down icon next to the resource or group you want to modify and select Edit.

   The resource configuration screen opens.
4. To add a new parameter, operation, or meta attribute, select an entry from the empty drop-down box.

5. To edit any values in the Operations category, click the Edit icon of the respective entry, enter a different value for the operation, and click Apply.

6. When you are finished, click the Apply button in the resource configuration screen to confirm your changes to the parameters, operations, or meta attributes. A message at the top of the screen shows if the action has been successful.
7.5.6 Adding STONITH Resources

⚠ Important: No Support Without STONITH

- You must have a node fencing mechanism for your cluster.
- The global cluster options `stonith-enabled` and `startup-fencing` must be set to `true`. When you change them, you lose support.

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. To add a STONITH resource for SBD, for libvirt (KVM/Xen) or for vCenter/ESX Server, the easiest way is to use the Hawk2 wizard (see Section 7.5.2, “Adding Resources with the Wizard”). While STONITH resources are configured similarly to other resources, their behavior is different in some respects. For details refer to Section 10.3, “STONITH Resources and Configuration”.

PROCEDURE 7.8: ADDING A STONITH RESOURCE

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. From the left navigation bar, select Configuration › Add Resource › Primitive.

3. Enter a unique Resource ID.

4. From the Class list, select the resource agent class `stonith`.

5. From the Type list, select the STONITH plug-in to control your STONITH device. A short description for this plug-in is displayed.

6. Hawk2 automatically shows the required Parameters for the resource. Enter values for each parameter.

7. Hawk2 displays the most important resource Operations and proposes default values. If you do not modify any settings here, Hawk2 adds the proposed operations and their default values when you confirm.

8. If there is no reason to change them, keep the default Meta Attributes settings.
9. Confirm your changes to create the STONITH resource.
   A message at the top of the screen shows if the action has been successful.

To complete your fencing configuration, add constraints. For more details, refer to Chapter 10, Fencing and STONITH.

7.5.7 Adding Cluster Resource Groups

Some cluster resources depend on other components or resources. They require that each component or resource starts in a specific order and runs on the same server. To simplify this configuration SUSE Linux Enterprise High Availability Extension supports the concept of groups.

Resource groups contain a set of resources that need to be located together, be started sequentially and stopped in the reverse order. For an example of a resource group and more information about groups and their properties, refer to Section 6.3.5.1, “Groups”.
Note: Empty Groups
Groups must contain at least one resource, otherwise the configuration is not valid. While creating a group, Hawk2 allows you to create more primitives and add them to the group. For details, see Section 7.7.1, “Editing Resources and Groups”.

PROCEDURE 7.9: ADDING A RESOURCE GROUP

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. From the left navigation bar, select Configuration › Add Resource › Group.

3. Enter a unique Group ID.

4. To define the group members, select one or multiple entries in the list of Children. Resort group members by dragging and dropping them into the order you want by using the “handle” icon on the right.

5. If needed, modify or add Meta Attributes.

6. Click Create to finish the configuration. A message at the top of the screen shows if the action has been successful.

![Create Group](image)

**FIGURE 7.6: HAWK2—RESOURCE GROUP**
7.5.8 Adding Clone Resources

If you want certain resources to run simultaneously on multiple nodes in your cluster, configure these resources as clones. An example of a resource that can be configured as a clone is `ocf:pacemaker:controld` for cluster file systems like OCFS2. Any regular resources or resource groups can be cloned. Instances of cloned resources may behave identically. However, they may also be configured differently, depending on which node they are hosted on.

For an overview of the available types of resource clones, refer to Section 6.3.5.2, “Clones”.

Note: Child Resources for Clones

Clones can either contain a primitive or a group as child resources. In Hawk2, child resources cannot be created or modified while creating a clone. Before adding a clone, create child resources and configure them as desired. For details, refer to Section 7.5.3, “Adding Simple Resources” or Section 7.5.7, “Adding Cluster Resource Groups”.

PROCEDURE 7.10: ADDING A CLONE RESOURCE

1. Log in to Hawk2:

   [https://HAWKSERVER:7630/](https://HAWKSERVER:7630/)

2. From the left navigation bar, select Configuration ➔ Add Resource ➔ Clone.

3. Enter a unique Clone ID.

4. From the Child Resource list, select the primitive or group to use as a sub-resource for the clone.

5. If needed, modify or add Meta Attributes.

6. Click Create to finish the configuration. A message at the top of the screen shows if the action has been successful.
7.5.9 Adding Multi-state Resources

Multi-state resources are a specialization of clones. They allow the instances to be in one of two operating modes (called active/passive, primary/secondary, or master/slave). 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, available from http://www.clusterlabs.org/pacemaker/doc/. Refer to section Multi-state - Resources That Have Multiple Modes.

Note: Child Resources for Multi-state Resources

Multi-state resources can either contain a primitive or a group as child resources. In Hawk2, child resources cannot be created or modified while creating a multi-state resource. Before adding a multi-state resource, create child resources and configure them as desired. For details, refer to Section 7.5.3, “Adding Simple Resources” or Section 7.5.7, “Adding Cluster Resource Groups”.

FIGURE 7.7: HAWK2—CLONE RESOURCE
PROCEDURE 7.11: ADDING A MULTI-STATE RESOURCE

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. From the left navigation bar, select Configuration > Add Resource > Multi-state.

3. Enter a unique Multi-state ID.

4. From the Child Resource list, select the primitive or group to use as a sub-resource for the multi-state resource.

5. If needed, modify or add Meta Attributes.

6. Click Create to finish the configuration. A message at the top of the screen shows if the action has been successful.

![FIGURE 7.8: HAWK2—MULTI-STATE RESOURCE](image)

7.5.10 Grouping Resources by Using Tags

Tags are a way to refer to multiple resources at once, without creating any colocation or ordering relationship between them. You can use tags for grouping conceptually related resources. For example, if you have several resources related to a database, you can add all related resources to a tag named database.
All resources belonging to a tag can be started or stopped with a single command.

**PROCEDURE 7.12: ADDING A TAG**

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select **Configuration > Add Resource > Tag**.

3. Enter a unique **Tag ID**.

4. From the **Objects** list, select the resources you want to refer to with the tag.

5. Click **Create** to finish the configuration. A message at the top of the screen shows if the action has been successful.

![Figure 7.9: HAWK2—TAG](image)

### 7.5.11 Configuring Resource Monitoring

The High Availability Extension does not only detect node failures, 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. Usually, resources are only monitored by the cluster while they are running. However, to detect concurrency violations, also configure monitoring for resources...
which are stopped. 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 7.13: ADDING AND MODIFYING AN OPERATION**

1. Log in to Hawk2:

   [https://HAWKSERVER:7630/](https://HAWKSERVER:7630/)

2. Add a resource as described in Procedure 7.5, “Adding a Primitive Resource” or select an existing primitive to edit. Hawk2 automatically shows the most important Operations (start, stop, monitor) and proposes default values. To see the attributes belonging to each proposed value, hover the mouse pointer over the respective value.

   ![](image)

3. To change the suggested `timeout` values for the `start` or `stop` operation:
   a. Click the pen icon next to the operation.
   b. In the dialog that opens, enter a different value for the `timeout` parameter, for example 10, and confirm your change.

4. To change the suggested `interval` value for the `monitor` operation:
   a. Click the pen icon next to the operation.
   b. In the dialog that opens, enter a different value for the monitoring `interval`.
   c. To configure resource monitoring in the case that the resource is stopped:
      i. Select the `role` entry from the empty drop-down box below.
      ii. From the `role` drop-down box, select `Stopped`.

---

114 Configuring Resource Monitoring SLE HA 15 SP1
iii. Click Apply to confirm your changes and to close the dialog for the operation.

5. Confirm your changes in the resource configuration screen. A message at the top of the screen shows if the action has been successful.

For the processes that take place if the resource monitor detects a failure, refer to Section 6.4, “Resource Monitoring”.

To view resource failures, switch to the Status screen in Hawk2 and select the resource you are interested in. In the Operations column click the arrow down icon and select Recent Events. The dialog that opens lists recent actions performed for the resource. Failures are displayed in red. To view the resource details, click the magnifier icon in the Operations column.

---

**FIGURE 7.10: HAWK2—RESOURCE DETAILS**

---

<table>
<thead>
<tr>
<th>Name</th>
<th>Timeout</th>
<th>Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>stop</td>
<td>20s</td>
<td>0</td>
</tr>
<tr>
<td>monitor</td>
<td>20s</td>
<td>10</td>
</tr>
</tbody>
</table>

---

**ID**

---

<table>
<thead>
<tr>
<th>Type</th>
<th>Score</th>
<th>To</th>
</tr>
</thead>
</table>
7.6 Configuring 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 to load resources, and what other resources a specific resource depends on.

For an overview of available types of constraints, refer to Section 6.5.1, “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 6.5.2, “Scores and Infinity”.

7.6.1 Adding Location Constraints

A location constraint determines on which node a resource may be run, is preferably run, or may not be run. An example of a location constraint is to place all resources related to a certain database on the same node.

PROCEDURE 7.14: ADDING A LOCATION CONSTRAINT

1. Log in to Hawk2:

   https://HAWKSReVER:7630/

2. From the left navigation bar, select Configuration > Add Constraint > Location.

3. Enter a unique Constraint ID.

4. From the list of Resources select the resource or resources for which to define the constraint.

5. Enter a Score. The score indicates the value you are assigning to this resource 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. Constraints with higher scores are applied before those with lower scores.
Some often-used values can also be set via the drop-down box:

- To force the resources to run on the node, click the arrow icon and select **Always**. This sets the score to **INFINITY**.

- If you never want the resources to run on the node, click the arrow icon and select **Never**. This sets the score to **-INFINITY**, meaning that the resources must not run on the node.

- To set the score to **0**, click the arrow icon and select **Advisory**. This disables the constraint. This is useful when you want to set resource discovery but do not want to constrain the resources.

6. Select a **Node**.

7. Click **Create** to finish the configuration. A message at the top of the screen shows if the action has been successful.

---

**FIGURE 7.11: HAWK2—LOCATION CONSTRAINT**

### 7.6.2 Adding Colocation Constraints

A colocation constraint tells the cluster which resources may or may not run together on a node. As a colocation constraint defines a dependency between resources, you need at least two resources to create a colocation constraint.
1. Log in to Hawk2:

https://HAWKSERVER:7630/

2. From the left navigation bar, select Configuration > Add Constraint > Colocation.

3. Enter a unique Constraint ID.

4. Enter a Score. The score determines the location relationship between the resources. Positive values indicate that the resources should run on the same node. Negative values indicate that the resources should not run on the same node. The score will be combined with other factors to decide where to put the resource. Some often-used values can also be set via the drop-down box:
   - To force the resources to run on the same node, click the arrow icon and select Always. This sets the score to INFINITY.
   - If you never want the resources to run on the same node, click the arrow icon and select Never. This sets the score to -INFINITY, meaning that the resources must not run on the same node.

5. To define the resources for the constraint:
   a. From the drop-down box in the Resources category, select a resource (or a template). The resource is added and a new empty drop-down box appears beneath.
   b. Repeat this step to add more resources. 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 not allow the dependent resource to run.
   c. To swap the order of resources within the colocation constraint, click the arrow up icon next to a resource to swap it with the entry above.

6. If needed, specify further parameters for each resource (such as Started, Stopped, Master, Slave, Promote, Demote): Click the empty drop-down box next to the resource and select the desired entry.

7. Click Create to finish the configuration. A message at the top of the screen shows if the action has been successful.
7.6.3 Adding Order Constraints

Order constraints define the order in which resources are started and stopped. As an order constraint defines a dependency between resources, you need at least two resources to create an order constraint.

**PROCEDURE 7.16: ADDING AN ORDER CONSTRAINT**

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. From the left navigation bar, select **Configuration > Add Constraint > Order**.

3. Enter a unique **Constraint ID**.

4. Enter a **Score**. If the score is greater than zero, the order constraint is mandatory, otherwise it is optional.
Some often-used values can also be set via the drop-down box:

- If you want to make the order constraint mandatory, click the arrow icon and select **Mandatory**.
- If you want the order constraint to be a suggestion only, click the arrow icon and select **Optional**.
- **Serialize**: To ensure that no two stop/start actions occur concurrently for the resources, click the arrow icon and select **Serialize**. This makes sure that one resource must complete starting before the other can be started. A typical use case are resources that put a high load on the host during start-up.

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

6. To define the resources for the constraint:

   a. From the drop-down box in the **Resources** category, select a resource (or a template). The resource is added and a new empty drop-down box appears beneath.

   b. Repeat this step to add more resources. The topmost resource will start first, then the second, etc. Usually the resources will be stopped in reverse order.

   c. To swap the order of resources within the order constraint, click the arrow up icon next to a resource to swap it with the entry above.

7. If needed, specify further parameters for each resource (like **Started**, **Stopped**, **Master**, **Slave**, **Promote**, **Demote**): Click the empty drop-down box next to the resource and select the desired entry.

8. Confirm your changes to finish the configuration. A message at the top of the screen shows if the action has been successful.
7.6.4 Using Resource Sets for Constraints

As an alternative format for defining constraints, you can use Resource Sets. They have the same ordering semantics as Groups.

PROCEDURE 7.17: USING A RESOURCE SET FOR CONSTRAINTS

1. To use a resource set within a location constraint:

   a. Proceed as outlined in Procedure 7.14, “Adding a Location Constraint”, apart from Step 4: Instead of selecting a single resource, select multiple resources by pressing Ctrl or Shift and mouse click. This creates a resource set within the location constraint.

   b. To remove a resource from the location constraint, press Ctrl and click the resource again to deselect it.

2. To use a resource set within a colocation or order constraint:

   a. Proceed as described in Procedure 7.15, “Adding a Colocation Constraint” or Procedure 7.16, “Adding an Order Constraint”, apart from the step where you define the resources for the constraint (Step 5.a or Step 6.a):

   b. Add multiple resources.
c. To create a resource set, click the chain icon next to a resource to link it to the resource above. A resource set is visualized by a frame around the resources belonging to a set.

d. You can combine multiple resources in a resource set or create multiple resource sets.

e. To unlink a resource from the resource above, click the scissors icon next to the resource.

3. Confirm your changes to finish the constraint configuration.

7.6.5 For More Information

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/pacemaker/doc/:

- Pacemaker Explained, chapter Resource Constraints
- Colocation Explained
- Ordering Explained
7.6.6 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 several 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:

PROCEDURE 7.18: SPECIFYING A FAILOVER NODE

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. Configure a location constraint for the resource as described in Procedure 7.14, “Adding a Location Constraint”.

3. Add the migration-threshold meta attribute to the resource as described in Procedure 7.7: Modifying Parameters, Operations, or Meta Attributes for a Resource, Step 4 and enter a value for the migration-threshold. The value should be positive and less than INFINITY.

4. If you want to automatically expire the failcount for a resource, add the failure-timeout meta attribute to the resource as described in Procedure 7.5: Adding a Primitive Resource, Step 4 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 6.8, "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 7.7.3, "Cleaning Up Resources" for details.

### 7.6.7 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 afterward.

For the implications of different resource stickiness values, refer to Section 6.5.5, "Failback Nodes".

**PROCEDURE 7.19: SPECIFYING RESOURCE STICKINESS**

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. Add the `resource-stickiness` meta attribute to the resource as described in Procedure 7.7: Modifying Parameters, Operations, or Meta Attributes for a Resource, Step 4.

3. Specify a value between `-INFINITY` and `INFINITY` for `resource-stickiness`.

---

124  Specifying Resource Failback Nodes (Resource Stickiness)  SLE HA 15 SP1
7.6.8 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.

For more details and a configuration example, refer to Section 6.5.6, “Placing Resources Based on Their Load Impact”.

Utilization attributes are used to configure both the resource's requirements and the capacity a node provides. You first need to configure a node's capacity before you can configure the capacity a resource requires.
PROCEDURE 7.20: CONFIGURING THE CAPACITY A NODE PROVIDES

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. From the left navigation bar, select Monitoring > Status.

3. On the Nodes tab, select the node whose capacity you want to configure.

4. In the Operations column, click the arrow down icon and select Edit.

   The Edit Node screen opens.

5. Below Utilization, enter a name for a utilization attribute into the empty drop-down box.

   The name can be arbitrary (for example, RAM_in_GB).

6. Click the Add icon to add the attribute.

7. In the empty text box next to the attribute, enter an attribute value. The value must be an integer.

8. Add as many utilization attributes as you need and add values for all of them.

9. Confirm your changes. A message at the top of the screen shows if the action has been successful.
PROCEDURE 7.21: CONFIGURING THE CAPACITY A RESOURCE REQUIRES

Configure the capacity a certain resource requires from a node either when creating a
primitive resource or when editing an existing primitive resource.

Before you can add utilization attributes to a resource, you need to have set utilization
attributes for your cluster nodes as described in Procedure 7.20.

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. To add a utilization attribute to an existing resource: Go to Manage › Status and open the
   resource configuration dialog as described in Section 7.7.1, “Editing Resources and Groups”.
   If you create a new resource: Go to Configuration › Add Resource and proceed as described
   in Section 7.5.3, “Adding Simple Resources”.

3. In the resource configuration dialog, go to the Utilization category.

4. From the empty drop-down box, select one of the utilization attributes that you have
   configured for the nodes in Procedure 7.20.

5. In the empty text box next to the attribute, enter an attribute value. The value must be
   an integer.

6. Add as many utilization attributes as you need and add values for all of them.

7. Confirm your changes. A message at the top of the screen shows if the action has been
   successful.

After you have configured the capacities your nodes provide and the capacities your resources
require, set the placement strategy in the global cluster options. Otherwise the capacity config-
urations 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 6.5.6, “Placing Resources Based on Their Load Impact”.

PROCEDURE 7.22: SETTING THE PLACEMENT STRATEGY

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. From the left navigation bar, select Configuration › Cluster Configuration to open the respec-
tive screen. It shows global cluster options and resource and operation defaults.
3. From the empty drop-down box in the upper part of the screen, select placement-strategy.
   By default, its value is set to Default, which means that utilization attributes and values are not considered.

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

5. Confirm your changes.

### 7.7 Managing Cluster Resources

In addition to configuring your cluster resources, Hawk2 allows you to manage existing resources from the Status screen. For a general overview of the screen refer to Section 7.8.1, “Monitoring a Single Cluster”.

#### 7.7.1 Editing Resources and Groups

In case you need to edit existing resources, go to the Status screen. In the Operations column, click the arrow down icon next to the resource or group you want to modify and select Edit.

The editing screen appears. If you edit a primitive resource, the following operations are available:

**OPERATIONS FOR PRIMITIVES**

- Copying the resource.
- Renaming the resource (changing its ID).
- Deleting the resource.

If you edit a group, the following operations are available:

**OPERATIONS FOR GROUPS**

- Creating a new primitive which will be added to this group.
- Renaming the group (changing its ID).
- Re-sort group members by dragging and dropping them into the order you want using the “handle” icon on the right.
Starting Resources

Before you start a cluster resource, make sure it is set up correctly. For example, if you use an Apache server as a cluster resource, set up the Apache server first. 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 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 the High Availability Extension takes over.

For interventions in resources that are currently managed by the cluster, set the resource to maintenance mode first. For details, see Procedure 17.5, “Putting a Resource into Maintenance Mode with Hawk2”.

When creating a resource with Hawk2, 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 7.23: STARTING A NEW RESOURCE

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. From the left navigation bar, select Monitoring, Status. The list of Resources also shows the Status.

3. Select the resource to start. In its Operations column click the Start icon. To continue, confirm the message that appears.

   When the resource has started, Hawk2 changes the resource's Status to green and shows on which node it is running.
7.7.3 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 it can be reset manually as described below.

**PROCEDURE 7.24: CLEANING UP A RESOURCE**

1. Log in to Hawk2:
   
   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select Status. The list of Resources also shows the Status.

3. Go to the resource to clean up. In the Operations column click the arrow down button and select Cleanup. To continue, confirm the message that appears.
   
   This executes the command `crm resource cleanup` and cleans up the resource on all nodes.

7.7.4 Removing Cluster Resources

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

**PROCEDURE 7.25: REMOVING A CLUSTER RESOURCE**

1. Log in to Hawk2:
   
   ```
   https://HAWKSERVER:7630/
   ```

2. Clean up the resource on all nodes as described in Procedure 7.24, “Cleaning Up A Resource”.

3. Stop the resource:
   
   a. From the left navigation bar, select Monitoring > Status. The list of Resources also shows the Status.
   
   b. In the Operations column click the Stop button next to the resource.
c. To continue, confirm the message that appears.

The Status column will reflect the change when the resource is stopped.

4. Delete the resource:

   a. From the left navigation bar, select Configuration ➤ Edit Configuration.

   b. In the list of Resources, go to the respective resource. From the Operations column click the Delete icon next to the resource.

   c. To continue, confirm the message that appears.

### 7.7.5 Migrating Cluster Resources

As mentioned in Section 7.6.6, “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). You can also manually migrate a resource to another node in the cluster. Or you decide to move the resource away from the current node and let the cluster decide where to put it.

**PROCEDURE 7.26: MANUALLY MIGRATING A RESOURCE**

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. From the left navigation bar, select Monitoring ➤ Status. The list of Resources also shows the Status.

3. In the list of Resources, select the respective resource.

4. In the Operations column click the arrow down button and select Migrate.

5. In the window that opens you have the following choices:

   - **Away from current node**: This creates a location constraint with a -INFINITY score for the current node.

   - Alternatively, you can move the resource to another node. This creates a location constraint with an INFINITY score for the destination node.

6. Confirm your choice.
To allow a resource to move back again, proceed as follows:

**PROCEDURE 7.27: UNMIGRATING A RESOURCE**

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. From the left navigation bar, select **Monitoring > Status**. The list of **Resources** also shows the **Status**.

3. In the list of **Resources**, go to the respective resource.

4. In the **Operations** column click the arrow down button and select **Clear**. To continue, confirm the message that appears.

   Hawk2 uses the `crm_resource --clear` command. The resource can move back to its original location or it may stay where it is (depending on resource stickiness).


### 7.8 Monitoring Clusters

Hawk2 has different screens for monitoring single clusters and multiple clusters: the **Status** and the **Dashboard** screen.

#### 7.8.1 Monitoring a Single Cluster

To monitor a single cluster, use the **Status** screen. After you have logged in to Hawk2, the **Status** screen is displayed by default. An icon in the upper right corner shows the cluster status at a glance. For further details, have a look at the following categories:

**Errors**

If errors have occurred, they are shown at the top of the page.

**Resources**

Shows the configured resources including their **Status**, **Name** (ID), **Location** (node on which they are running), and resource agent **Type**. From the **Operations** column, you can start or stop a resource, trigger several actions, or view details. Actions that can be triggered
include setting the resource to maintenance mode (or removing maintenance mode), migrating it to a different node, cleaning up the resource, showing any recent events, or editing the resource.

**Nodes**

Shows the nodes belonging to the cluster site you are logged in to, including the nodes’ **Status** and **Name**. In the **Maintenance** and **Standby** columns, you can set or remove the maintenance or standby flag for a node. The **Operations** column allows you to view recent events for the node or further details: for example, if a utilization, standby or maintenance attribute is set for the respective node.

**Tickets**

Only shown if tickets have been configured (for use with Geo clustering).

---

**FIGURE 7.15: HAWK2—CLUSTER STATUS**

### 7.8.2 Monitoring Multiple Clusters

To monitor multiple clusters, use the Hawk2 Dashboard. The cluster information displayed in the Dashboard screen is stored on the server side. It is synchronized between the cluster nodes (if passwordless SSH access between the cluster nodes has been configured). For details, see Section D.2, “Configuring a Passwordless SSH Account”. However, the machine running Hawk2 does not even need to be part of any cluster for that purpose—it can be a separate, unrelated system.
In addition to the general *Hawk2 Requirements*, the following prerequisites need to be fulfilled to monitor multiple clusters with Hawk2:

**PREREQUISITES**

- All clusters to be monitored from Hawk2’s *Dashboard* must be running SUSE Linux Enterprise High Availability Extension 15 SP1.

- If you did not replace the self-signed certificate for Hawk2 on every cluster node with your own certificate (or a certificate signed by an official Certificate Authority) yet, do the following: Log in to Hawk2 on *every* node in *every* cluster at least once. Verify the certificate (or add an exception in the browser to bypass the warning). Otherwise Hawk2 cannot connect to the cluster.

**PROCEDURE 7.28: MONITORING MULTIPLE CLUSTERS WITH THE DASHBOARD**

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Monitoring > Dashboard*.

   Hawk2 shows an overview of the resources and nodes on the current cluster site. In addition, it shows any *Tickets* that have been configured for use with a Geo cluster. If you need information about the icons used in this view, click *Legend*. To search for a resource ID, enter the name (ID) into the *Search* text box. To only show specific nodes, click the filter icon and select a filtering option.
3. To add dashboards for multiple clusters:

   a. Click Add Cluster.

   b. Enter the Cluster name with which to identify the cluster in the Dashboard. For example, berlin.

   c. Enter the fully qualified host name of one of the nodes in the second cluster. For example, charlie.
d. Click Add. Hawk2 will display a second tab for the newly added cluster site with an overview of its nodes and resources.

Note: Connection Error
If instead you are prompted to log in to this node by entering a password, you probably did not connect to this node yet and have not replaced the self-signed certificate. In that case, even after entering the password, the connection will fail with the following message: Error connecting to server. Retrying every 5 seconds...

To proceed, see Replacing the Self-Signed Certificate.

4. To view more details for a cluster site or to manage it, switch to the site's tab and click the chain icon.
Hawk2 opens the Status view for this site in a new browser window or tab. From there, you can administer this part of the Geo cluster.

5. To remove a cluster from the dashboard, click the x icon on the right-hand side of the cluster's details.
7.9 Using the Batch Mode

Hawk2 provides a Batch Mode, including a cluster simulator. It can be used for the following:

- Staging changes to the cluster and applying them as a single transaction, instead of having each change take effect immediately.

- Simulating changes and cluster events, for example, to explore potential failure scenarios.

For example, batch mode can be used when creating groups of resources that depend on each other. Using batch mode, you can avoid applying intermediate or incomplete configurations to the cluster.

While batch mode is enabled, you can add or edit resources and constraints or change the cluster configuration. It is also possible to simulate events in the cluster, including nodes going online or offline, resource operations and tickets being granted or revoked. See Procedure 7.30, “Injecting Node, Resource or Ticket Events” for details.

The cluster simulator runs automatically after every change and shows the expected outcome in the user interface. For example, this also means: If you stop a resource while in batch mode, the user interface shows the resource as stopped—while actually, the resource is still running.

**Important: Wizards and Changes to the Live System**

Some wizards include actions beyond mere cluster configuration. When using those wizards in batch mode, any changes that go beyond cluster configuration would be applied to the live system immediately.

Therefore wizards that require root permission cannot be executed in batch mode.

PROCEDURE 7.29: WORKING WITH THE BATCH MODE

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. To activate the batch mode, select Batch from the top-level row.
   An additional bar appears below the top-level row. It indicates that batch mode is active and contains links to actions that you can execute in batch mode.
3. While batch mode is active, perform any changes to your cluster, like adding or editing resources and constraints or editing the cluster configuration. The changes will be simulated and shown in all screens.

4. To view details of the changes you have made, select Show from the batch mode bar. The Batch Mode window opens. For any configuration changes it shows the difference between the live state and the simulated changes in crmsh syntax: Lines starting with a - character represent the current state whereas lines starting with + show the proposed state.

5. To inject events or view even more details, see Procedure 7.30. Otherwise Close the window.

6. Choose to either Discard or Apply the simulated changes and confirm your choice. This also deactivates batch mode and takes you back to normal mode.

When running in batch mode, Hawk2 also allows you to inject Node Events and Resource Events.

**Node Events**

Let you change the state of a node. Available states are online, offline, and unclean.

**Resource Events**

Let you change some properties of a resource. For example, you can set an operation (like start, stop, monitor), the node it applies to, and the expected result to be simulated.

**Ticket Events**

FIGURE 7.17: HAWK2 BATCH MODE ACTIVATED
Let you test the impact of granting and revoking tickets (used for Geo clusters).

**PROCEDURE 7.30: INJECTING NODE, RESOURCE OR TICKET EVENTS**

1. Log in to Hawk2:
   
   `https://HAWKSERVER:7630/`

2. If batch mode is not active yet, click *Batch* at the top-level row to switch to batch mode.

3. In the batch mode bar, click *Show* to open the *Batch Mode* window.

4. To simulate a status change of a node:
   
   a. Click *Inject > Node Event*.
   
   b. Select the *Node* you want to manipulate and select its target *State*.
   
   c. Confirm your changes. Your event is added to the queue of events listed in the *Batch Mode* dialog.

5. To simulate a resource operation:
   
   a. Click *Inject > Resource Event*.
   
   b. Select the *Resource* you want to manipulate and select the *Operation* to simulate.
   
   c. If necessary, define an *Interval*.
   
   d. Select the *Node* on which to run the operation and the targeted *Result*. Your event is added to the queue of events listed in the *Batch Mode* dialog.
   
   e. Confirm your changes.

6. To simulate a ticket action:
   
   a. Click *Inject > Ticket Event*.
   
   b. Select the *Ticket* you want to manipulate and select the *Action* to simulate.
   
   c. Confirm your changes. Your event is added to the queue of events listed in the *Batch Mode* dialog.

7. The *Batch Mode* dialog (Figure 7.18) shows a new line per injected event. Any event listed here is simulated immediately and is reflected on the *Status* screen.
If you have made any configuration changes, too, the difference between the live state and the simulated changes is shown below the injected events.

8. To remove an injected event, click the **Remove** icon next to it. Hawk2 updates the **Status** screen accordingly.

9. To view more details about the simulation run, click **Simulator** and choose one of the following:

   **Summary**
   
   Shows a detailed summary.

   **CIB (in)/CIB (out)**
   
   **CIB (in)** shows the initial CIB state. **CIB (out)** shows what the CIB would look like after the transition.

   **Transition Graph**
   
   Shows a graphical representation of the transition.

   **Transition**
   
   Shows an XML representation of the transition.

10. If you have reviewed the simulated changes, close the **Batch Mode** window.

11. To leave the batch mode, either **Apply** or **Discard** the simulated changes.
7.10 Viewing the Cluster History

Hawk2 provides the following possibilities to view past events on the cluster (on different levels and in varying detail):

- Section 7.10.1, “Viewing Recent Events of Nodes or Resources”
- Section 7.10.2, “Using the History Explorer for Cluster Reports”
- Section 7.10.3, “Viewing Transition Details in the History Explorer”

7.10.1 Viewing Recent Events of Nodes or Resources

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. From the left navigation bar, select Monitoring > Status. It lists Resources and Nodes.

3. To view recent events of a resource:

   a. Click Resources and select the respective resource.

   b. In the Operations column for the resource, click the arrow down button and select Recent events.

   Hawk2 opens a new window and displays a table view of the latest events.

4. To view recent events of a node:

   a. Click Nodes and select the respective node.

   b. In the Operations column for the node, select Recent events.

   Hawk2 opens a new window and displays a table view of the latest events.

<table>
<thead>
<tr>
<th>RC</th>
<th>Resource</th>
<th>Operation</th>
<th>Last Change</th>
<th>State</th>
<th>Call</th>
<th>Exec</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>dummy1</td>
<td>dummy1_start_0</td>
<td>Tue Oct 25 18:10:49 2016</td>
<td>started</td>
<td>18</td>
<td>20ms</td>
<td>✓</td>
</tr>
<tr>
<td>0</td>
<td>dummy1</td>
<td>dummy1_start_0</td>
<td>Tue Oct 25 18:10:49 2016</td>
<td>started</td>
<td>19</td>
<td>20ms</td>
<td>✓</td>
</tr>
<tr>
<td>0</td>
<td>dummy2</td>
<td>dummy2_stop_0</td>
<td>Tue Oct 25 18:10:03 2016</td>
<td>stopped (disabled)</td>
<td>15</td>
<td>23ms</td>
<td>✓</td>
</tr>
<tr>
<td>0</td>
<td>dummy2</td>
<td>dummy2_stop_0</td>
<td>Tue Oct 25 18:09:50 2016</td>
<td>stopped (disabled)</td>
<td>13</td>
<td>19ms</td>
<td>✓</td>
</tr>
</tbody>
</table>
7.10.2 Using the History Explorer for Cluster Reports

From the left navigation bar, select Troubleshooting > History to access the History Explorer. The History Explorer allows you to create detailed cluster reports and view transition information. It provides the following options:

**Generate**

Create a cluster report for a certain time. Hawk2 calls the `crm report` command to generate the report.

**Upload**

Allows you to upload `crm report` archives that have either been created with the `crm` shell directly or even on a different cluster.

After reports have been generated or uploaded, they are shown below Reports. From the list of reports, you can show a report’s details, download or delete the report.

---

**FIGURE 7.19: HAWK2—HISTORY EXPLORER MAIN VIEW**

**PROCEDURE 7.31: GENERATING OR UPLOADING A CLUSTER REPORT**

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. From the left navigation bar, select Troubleshooting > History.

   The History Explorer screen opens in the Generate view. By default, the suggested time frame for a report is the last hour.
3. To create a cluster report:

   a. To immediately start a report, click **Generate**.

   b. To modify the time frame for the report, click anywhere on the suggested time frame and select another option from the drop-down box. You can also enter a **Custom** start date, end date and hour, respectively. To start the report, click **Generate**.

   After the report has finished, it is shown below **Reports**.

4. To upload a cluster report, the **crm report** archive must be located on a file system that you can access with Hawk2. Proceed as follows:

   a. Switch to the **Upload** tab.

   b. **Browse** for the cluster report archive and click **Upload**.

   After the report is uploaded, it is shown below **Reports**.

5. To download or delete a report, click the respective icon next to the report in the **Operations** column.

6. To view **Report Details in History Explorer**, click the report's name or select **Show** from the **Operations** column.

7. Return to the list of reports by clicking the **Reports** button.
REPORT DETAILS IN HISTORY EXPLORER

- Name of the report.
- Start time of the report.
- End time of the report.
- Number of transitions plus time line of all transitions in the cluster that are covered by the report. To learn how to view more details for a transition, see Section 7.10.3.
- Node events.
- Resource events.

7.10.3 Viewing Transition Details in the History Explorer

For each transition, the cluster saves a copy of the state which it provides as input to pacemaker-schedulerc. The path to this archive is logged. All pe-* files are generated on the Designated Coordinator (DC). As the DC can change in a cluster, there may be pe-* files from several nodes. Any pe-* files are saved snapshots of the CIB, used as input of calculations by pacemaker-schedulerc.

In Hawk2, you can display the name of each pe-* file plus the time and node on which it was created. In addition, the History Explorer can visualize the following details, based on the respective pe-* file:

TRANSITION DETAILS IN THE HISTORY EXPLORER

Details
Shows snippets of logging data that belongs to the transition. Displays the output of the following command (including the resource agents' log messages):

```bash
crm history transition peinput
```

Configuration
Shows the cluster configuration at the time that the pe-* file was created.

Diff
Shows the differences of configuration and status between the selected pe-* file and the following one.
Log

Shows snippets of logging data that belongs to the transition. Displays the output of the following command:

```
crm history transition log peinput
```

This includes details from the following daemons: pacemaker-schedulercd, pacemaker-controld, and pacemaker-execd.

Graph

Shows a graphical representation of the transition. If you click Graph, the calculation is simulated (exactly as done by pacemaker-schedulercd) and a graphical visualization is generated.

PROCEDURE 7.32: VIEWING TRANSITION DETAILS

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. From the left navigation bar, select Troubleshooting ➔ History.
   If reports have already been generated or uploaded, they are shown in the list of Reports. Otherwise generate or upload a report as described in Procedure 7.31.

3. Click the report's name or select Show from the Operations column to open the Report Details in History Explorer.

4. To access the transition details, you need to select a transition point in the transition time line that is shown below. Use the Previous and Next icons and the Zoom In and Zoom Out icons to find the transition that you are interested in.

5. To display the name of a pe-input* file plus the time and node on which it was created, hover the mouse pointer over a transition point in the time line.

6. To view the Transition Details in the History Explorer, click the transition point for which you want to know more.

7. To show Details, Configuration, Diff, Logs or Graph, click the respective buttons to show the content described in Transition Details in the History Explorer.

8. To return to the list of reports, click the Reports button.
7.11 Verifying Cluster Health

Hawk2 provides a wizard which checks and detects issues with your cluster. After the analysis is complete, Hawk2 creates a cluster report with further details. To verify cluster health and generate the report, Hawk2 requires passwordless SSH access between the nodes. Otherwise it can only collect data from the current node. If you have set up your cluster with the bootstrap scripts, provided by the ha-cluster-bootstrap package, passwordless SSH access is already configured. In case you need to configure it manually, see Section D.2, “Configuring a Passwordless SSH Account”.

1. Log in to Hawk2:

   https://HAWKSERVER:7630/

2. From the left navigation bar, select Configuration › Wizards.

3. Expand the Basic category.

4. Select the Verify health and configuration wizard.

5. Confirm with Verify.

6. Enter the root password for your cluster and click Apply. Hawk2 will generate the report.
To configure and manage cluster resources, either use the crm shell (crmsh) command line utility or Hawk2, 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 need 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.

### 8.1 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: Interactive crm Prompt

By using crm without arguments (or with only one sublevel as argument), the crm shell enters the interactive mode. This mode is indicated by the following prompt:

crm(live/HOSTNAME)

For readability reasons, we omit the host name in the interactive crm prompts in our documentation. We only include the host name if you need to run the interactive shell on a specific node, like alice for example:

crm(live/alice)

8.1.1 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 the `group` subcommand of `configure`:

  root # crm configure help group

This is the same:

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 -I` will complete the word like in the interactive shell.

8.1.2 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).
  It is possible to abbreviate subcommands as long as they are unique. For example, you can shorten `status` as `st` and `crmsh` will know what you have meant.
  Another feature is to shorten parameters. Usually, you add parameters through the `params` keyword. You can leave out the `params` section if it is the first and only section. For example, this line:

  ```
  root # crm primitive ipaddr IPaddr2 params ip=192.168.0.55
  ```

  is equivalent to this line:

  ```
  root # crm primitive ipaddr IPaddr2 ip=192.168.0.55
  ```

- **As crm Shell Script:** Crm shell scripts contain subcommands of `crm`. For more information, see Section 8.1.4, “Using crmsh's Shell Scripts”.

- **As crmsh Cluster Scripts:** These are a collection of metadata, references to RPM packages, configuration files, and crmsh subcommands bundled under a single, yet descriptive name. They are managed through the `crm script` command.
Do not confuse them with crmsh shell scripts: although both share some common objectives, the crm shell scripts only contain subcommands whereas cluster scripts incorporate much more than a simple enumeration of commands. For more information, see Section 8.1.5, “Using crmsh's Cluster Scripts”.

- **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 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#`. To leave the internal shell, use the command `quit`. 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 press `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 of some important aspects of the `crm` tool.
Displaying Information about OCF Resource Agents

As you need to deal with resource agents in your cluster configuration all the time, the `crm` tool contains the `ra` command. Use it to show information about resource agents and to manage them (for additional information, see also Section 6.3.2, “Supported Resource Agent Classes”):

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

The command `classes` lists all classes and providers:

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

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

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

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

```
crm(live)ra# info ocf:linbit:drbd
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):

<table>
<thead>
<tr>
<th>Operation</th>
<th>timeout</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>240</td>
</tr>
<tr>
<td>promote</td>
<td>90</td>
</tr>
<tr>
<td>demote</td>
<td>90</td>
</tr>
<tr>
<td>notify</td>
<td>90</td>
</tr>
<tr>
<td>stop</td>
<td>100</td>
</tr>
<tr>
<td>monitor_Slave_0</td>
<td>interval=20 timeout=20 start-delay=1m</td>
</tr>
<tr>
<td>monitor_Master_0</td>
<td>interval=10 timeout=20 start-delay=1m</td>
</tr>
</tbody>
</table>

Leave the viewer by pressing Q.

Tip: Use crm Directly

In the former example we used the internal shell of the crm command. However, you do not necessarily need 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.

8.1.4 Using crmsh's Shell Scripts

The crmsh shell scripts provide a convenient way to enumerate crmsh subcommands into a file. This makes it easy to comment specific lines or to replay them later. Keep in mind that a crmsh shell script can contain only crmsh subcommands. Any other commands are not allowed.

Before you can use a crmsh shell script, create a file with specific commands. For example, the following file prints the status of the cluster and gives a list of all nodes:

```
EXAMPLE 8.1: A SIMPLE CRMSH SHELL SCRIPT

# A small example file with some crm subcommands
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 is recommended to indent lines that belong to a certain subcommand to improve readability.
To use this script, use one of the following methods:

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

### 8.1.5 Using crmsh's Cluster Scripts

Collecting information from all cluster nodes and deploying any changes is a key cluster administration task. Instead of performing the same procedures manually on different nodes (which is error-prone), you can use the crmsh cluster scripts.

Do not confuse them with the *crmsh shell scripts*, which are explained in Section 8.1.4, “Using crmsh’s Shell Scripts”.

In contrast to crmsh shell scripts, cluster scripts performs additional tasks like:

- Installing software that is required for a specific task.
- Creating or modifying any configuration files.
- Collecting information and reporting potential problems with the cluster.
- Deploying the changes to all nodes.

Crmsh cluster scripts do not replace other tools for managing clusters—they provide an integrated way to perform the above tasks across the cluster. Find detailed information at [http://crmsh.github.io/scripts/](http://crmsh.github.io/scripts/).

#### 8.1.5.1 Usage

To get a list of all available cluster scripts, run:

```
root # crm script list
```

To view the components of a script, use the `show` command and the name of the cluster script, for example:

```
root # crm script show mailto
mailto (Basic)
MailTo

This is a resource agent for MailTo. It sends email to a sysadmin whenever a takeover occurs.
```
1. Notifies recipients by email in the event of resource takeover

- id (required) (unique)
  - Identifier for the cluster resource
- email (required)
  - Email address
- subject
  - Subject

The output of `show` contains a title, a short description, and a procedure. Each procedure is divided into a series of steps, performed in the given order.

Each step contains a list of required and optional parameters, along with a short description and its default value.

Each cluster script understands a set of common parameters. These parameters can be passed to any script:

**Table 8.1: Common Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>INDEX</td>
<td>If set, only execute a single action (index, as returned by verify)</td>
</tr>
<tr>
<td>dry_run</td>
<td>BOOL</td>
<td>If set, simulate execution only (default: no)</td>
</tr>
<tr>
<td>nodes</td>
<td>LIST</td>
<td>List of nodes to execute the script for</td>
</tr>
<tr>
<td>port</td>
<td>NUMBER</td>
<td>Port to connect to</td>
</tr>
<tr>
<td>statefile</td>
<td>FILE</td>
<td>When single-stepping, the state is saved in the given file</td>
</tr>
<tr>
<td>sudo</td>
<td>BOOL</td>
<td>If set, crm will prompt for a sudo password and use sudo where appropriate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(default: no)</td>
</tr>
<tr>
<td>timeout</td>
<td>NUMBER</td>
<td>Execution timeout in seconds (default: 600)</td>
</tr>
</tbody>
</table>
### 8.1.5.2 Verifying and Running a Cluster Script

Before running a cluster script, review the actions that it will perform and verify its parameters to avoid problems. A cluster script can potentially perform a series of actions and may fail for various reasons. Thus, verifying your parameters before running it helps to avoid problems.

For example, the `mailto` resource agent requires a unique identifier and an e-mail address. To verify these parameters, run:

```
root # crm script verify mailto id=sysadmin email=tux@example.org
1. Ensure mail package is installed
   mailx
2. Configure cluster resources
   primitive sysadmin MailTo
   email="tux@example.org"
   op start timeout="10"
   op stop timeout="10"
   op monitor interval="10" timeout="10"
   clone c-sysadmin sysadmin
```

The `verify` prints the steps and replaces any placeholders with your given parameters. If `verify` finds any problems, it will report it. If everything is ok, replace the `verify` command with `run`:

```
root # crm script run mailto id=sysadmin email=tux@example.org
INFO: MailTo
INFO: Nodes: alice, bob
OK: Ensure mail package is installed
OK: Configure cluster resources
```

Check whether your resource is integrated into your cluster with `crm status`:

```
root # crm status
[...] status
Clone Set: c-sysadmin [sysadmin]
```
8.1.6 Using Configuration Templates

Note: Deprecation Notice

The use of configuration templates is deprecated and will be removed in the future. Configuration templates will be replaced by cluster scripts, see Section 8.1.5, “Using crmsh’s Cluster Scripts”.

Configuration templates are ready-made cluster configurations for crmsh. Do not confuse them with the resource templates (as described in Section 8.3.3, “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 select 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 configuration you have created:

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

b. Display the minimum required changes that need 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 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
```

6. Submit your changes to the CIB:

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

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

```bash
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.

### 8.1.7 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:
   ```bash
   root # crm configure
   ```

2. Create a new shadow configuration:
   ```bash
   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. To copy the current live configuration into your shadow configuration, use the following command, otherwise skip this step:
   ```bash
   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:
   ```bash
   crm(myNewConfig)# commit
   ```

5. If you need the live cluster configuration again, switch back with the following command:
   ```bash
   crm(myNewConfig)configure# cib use live
   crm(live)#
   ```
8.1.8 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
```

8.1.9 Cluster Diagram

To output a 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
```

8.2 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
```
The `diff` command is very helpful: 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](http://crmsh.nongnu.org/crm.8.html#cmdhelp_corosync).

### 8.3 Configuring Cluster Resources

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

For an overview of resource types you can create, refer to Section 6.3.3, “Types of Resources”.
8.3.1 Loading Cluster Resources from a File

Parts or all of the configuration can be loaded from a local file or a network URL. Three different methods can be defined:

replace
This option replaces the current configuration with the new source configuration.

update
This option tries to import the source configuration. It adds new items or updates existing items to the current configuration.

push
This option imports the content from the source into the current configuration (same as update). However, it removes objects that are not available in the new configuration.

To load the new configuration from the file mycluster-config.txt use the following syntax:

```
root #crm configure load push mycluster-config.txt
```

8.3.2 Creating Cluster Resources

There are three types of RAs (Resource Agents) available with the cluster (for background information, see Section 6.3.2, “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 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

8.3.3 Creating Resource Templates

If you want to create several resources with similar configurations, a resource template simplifies the task. See also Section 6.5.3, “Resource Templates and Constraints” for some basic background information. Do not confuse them with the “normal” templates from Section 8.1.6, “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 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 example, the following new primitive MyVM2 doubles the timeout for monitor operations but leaves others untouched:

```
crm(live)configure# primitive MyVM2 @BigVM 
   params xmemfile="/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.

### 8.3.4 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/ibmrsla
   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
```

8.3.5 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 6.5, “Resource Constraints”.

8.3.5.1 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 ping:

```
crm(live)configure# primitive ping ping \
    params name=ping dampen=5s multiplier=100 host_list="r1 r2"
crm(live)configure# clone cl-ping ping meta interleave=true
crm(live)configure# location loc-node_pref internal_www \
    rule 50: #uname eq alice \
    rule ping: defined ping
```

The parameter `host_list` is a space-separated list of hosts to ping and count. 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 needed to be duplicated several times in the configuration, making it unnecessarily complex.

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

```
crm(live)configure# primitive vip1 IPaddr2 params ip=192.168.1.5
crm(live)configure# primitive vip2 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
```

### 8.3.5.2 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.

### 8.3.5.3 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.

```bash
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.

### 8.3.5.4 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:

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

### 8.3.5.5 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.

```bash
crm(live)configure# colocation filesystem_on_master inf:
```
The NFS server and 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 and 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 Mandatory:  
  drbd_resource:promote filesystem_resource:start
```

### 8.3.6 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 rsc1-alice rsc1 100: alice
```

Normally, rsc1 prefers to run on alice. If it fails there, migration-threshold is checked and compared to the failcount. If failcount $\geq$ migration-threshold then it is migrated to the node with the next best preference.

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

For an overview, refer to Section 6.5.4, “Failover Nodes”.

### 8.3.7 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. To prevent a resource from failing back to the node that it was running on, or to specify a different node for the resource to fail back to, change its resource stickiness value. You can either specify resource stickiness when you are creating a resource or afterward.

For an overview, refer to Section 6.5.5, “Failback Nodes”.

---

**filesystem_resource drbd_resource:Master**
8.3.8 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 6.5.6, “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. You now additionally want to configure the capacities a certain node provides and the capacity a certain resource requires.

PROCEDURE 8.1: 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 Xen ... \
       utilization memory=4096 cpu=4
   ```
This would make the resource consume 4096 of those memory units from \textit{NODE\_1}, and 4 of the CPU units.

4. Configure the placement strategy with the \texttt{property} command:

\begin{verbatim}
crm(live)configure# property ...
\end{verbatim}

The following values are available:

\begin{description}
\item[default (default value)] Utilization values are not considered. Resources are allocated according to location scoring. If scores are equal, resources are evenly distributed across nodes.
\item[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.
\item[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 (to achieve power savings on the remaining nodes).
\item[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.
\end{description}

\begin{itemize}
\item \textbf{Note: Configuring Resource Priorities}
\end{itemize}

The available placement strategies are best-effort—they do not yet use complex heuristic solvers to always reach optimum allocation results. Ensure that resource priorities are properly set so that your most important resources are scheduled first.

5. Commit your changes before leaving \texttt{crmsh}:

\begin{verbatim}
crm(live)configure# commit
\end{verbatim}
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 charlie utilization memory="4000"
crm(live)configure# primitive xenA Xen 
    utilization hv_memory="3500" meta priority="10" 
    params xmfile="/etc/xen/shared-vm/vm1"
crm(live)configure# primitive xenB Xen 
    utilization hv_memory="2000" meta priority="1" 
    params xmfile="/etc/xen/shared-vm/vm2"
crm(live)configure# primitive xenC Xen 
    utilization hv_memory="2000" meta priority="1" 
    params xmfile="/etc/xen/shared-vm/vm3"
crm(live)configure# primitive xenD Xen 
    utilization hv_memory="1000" meta priority="5" 
    params xmfile="/etc/xen/shared-vm/vm4"
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.

### 8.3.9 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 6.4, “Resource Monitoring”](#).
8.3.10 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:heartbeat:IPaddr \ 
        params ip=1.2.3.4 id= Public-IP
    crm(live)configure# primitive Email systemd:postfix \ 
        params id=Email
    ```

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

    ```
    crm(live)configure# group g-mailsvc 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-mailsvc add Email before Public-IP
    ```

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

    ```
    crm(live)configure# modgroup g-mailsvc remove Email
    ```

For an overview, refer to Section 6.3.5.1, “Groups”.

8.3.11 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 be useful for several 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 6.3.5.2, “Clones”.
8.3.11.1 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 apache
   ```

3. Clone the primitive:

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

8.3.11.2 Creating Promotable Clone Resources

Promotable clone resources (formerly known as multi-state resources) are a specialization of clones. This type allows the instances to be in one of two operating modes, be it active/passive, primary/secondary, or master/slave.

To create a promotable clone resource, first create a primitive resource and then the promotable clone resource. The promotable clone 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 \n   op monitor interval=61 role=Master
   ```

3. Create the promotable clone resource:

   ```
  crm(live)configure# ms ms-rsc my-rsc
   ```
8.4 Managing Cluster Resources

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

8.4.1 Showing Cluster Resources

When administering a cluster the command _crm configure show_ lists the current CIB objects like cluster configuration, global options, primitives, and others:

```bash
root # crm configure show
node 178326192: alice
node 178326448: bob
primitive admin_addr IPaddr2 \
    params ip=192.168.2.1 \
    op monitor interval=10 timeout=20
primitive stonith-sbd stonith:external/sbd \
    params pcmk_delay_max=30
property cib-bootstrap-options: \
    have-watchdog=true \
    dc-version=1.1.15-17.1-e174ec8 \
    cluster-infrastructure=corosync \
    cluster-name=hacluster \
    stonith-enabled=true \
    placement-strategy=balanced \
    standby-mode=true
rsc_defaults rsc-options: \
    resource-stickiness=1 \
    migration-threshold=3
op_defaults op-options: \
    timeout=600 \
    record-pending=true
```

In case you have lots of resources, the output of _show_ is too verbose. To restrict the output, use the name of the resource. For example, to list the properties of the primitive _admin_addr_ only, append the resource name to _show_:

```bash
root # crm configure show admin_addr
primitive admin_addr IPaddr2 \
    params ip=192.168.2.1 \
    op monitor interval=10 timeout=20
```
However, in some cases, you want to limit the output of specific resources even more. This can be achieved with filters. Filters limit the output to specific components. For example, to list the nodes only, use `type:node`:

```
root # crm configure show type:node
node 178326192: alice
node 178326448: bob
```

In case you are also interested in primitives, use the `or` operator:

```
root # crm configure show type:node or type:primitive
node 178326192: alice
node 178326448: bob
primitive admin_addr IPaddr2
  params ip=192.168.2.1
  op monitor interval=10 timeout=20
primitive stonith-sbd stonith:external/sbd
  params pcmk_delay_max=30
```

Furthermore, to search for an object that starts with a certain string, use this notation:

```
root # crm configure show type:primitive and and 'admin*'  
primitive admin_addr IPaddr2
  params ip=192.168.2.1
  op monitor interval=10 timeout=20
```

To list all available types, enter `crm configure show type:` and press the `→|` key. The Bash completion will give you a list of all types.

### 8.4.2 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 ID
   ```
8.4.3 Stopping a Cluster Resource

To stop one or more existing cluster resources you need the respective identifier(s). 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. Stop the resource with **stop** and press the →| key to show all known resources:

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

   It’s possible to stop multiple resources at once:

   ```
   crm(live)resource#stop ID1 ID2 ...
   ```

8.4.4 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 when 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:heartbeat:lvmlockd) Started

3. To clean up the resource **dlm**, for example:

   ```
   root #crm resource cleanup dlm
   ```
8.4.5 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
   ```

8.4.6 Migrating a Cluster Resource

Although resources are configured to automatically fail over (or migrate) to other nodes of the cluster if a hardware or software failure occurs, you can also manually move a resource to another node using either Hawk2 or the command line.

Use the move 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# move ipaddress1 bob
```
8.4.7 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 several resources related to a database, create a tag called `databases` and add all resources related to the database to this tag:

```
root # crm configure tag databases: db1 db2 db3
```

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

```
root # crm resource start databases
```

Similarly, you can stop them all too:

```
root # crm resource stop databases
```

8.4.8 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. However, for this subsection, we focus on how to get 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
```

If you prefer to run only one step from the suite, the `describe` command lists 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
statefile='health.json'
```

It is also possible to run the above commands with `crm cluster health`. 
For additional information regarding scripts, see http://crmsh.github.io/scripts/.

8.5 Setting Passwords Independent of cib.xml

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

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

```
root #crm configure show
primitive mydb mysql \
  params replication_user=admin ...
```

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.

8.6 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`.

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
```
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](http://labix.org/python-dateutil).

The **info** subcommand shows all the parameters which are covered by the **crm report**:

```text
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:

```text
detail 1
```

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:

```text
transition -1
```

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:

```text
transition -1 nograph
```
8.7  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.
9 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.

9.1 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.

⚠️ 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. For more information on STONITH devices and resource agents, see Chapter 10, Fencing and STONITH.

As of yet there is no documentation about writing STONITH agents. If you want to write new STONITH agents, consult the examples available in the source of the `cluster-glue` package.

9.2 Writing OCF Resource Agents

All OCF resource agents (RAs) are available in `/usr/lib/ocf/resource.d/`, see Section 6.3.2, “Supported Resource Agent Classes” for more information. Each resource agent must support the following operations to control it:

`start`
start or enable the resource

stop

stop or disable the resource

status

returns the status of the resource

monitor

similar to status, but checks also for unexpected states

validate

validate the resource's configuration

meta-data

returns information about the resource agent in XML

The general procedure of how to create an 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`:

   ```
   root #crm configure primitive coffee_1 ocf:coffee_machine:kitchen ...
   ```

3. Implement the different shell functions and save your file under a different name.

More details about writing OCF resource agents can be found at [http://linux-ha.org/wiki/Resources_Agents](http://linux-ha.org/wiki/Resources_Agents). Find special information about several concepts at *Chapter 1, Product Overview.*
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>
<thead>
<tr>
<th>Recovery Type</th>
<th>Description</th>
<th>Action Taken by the Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>soft</td>
<td>A transient error occurred.</td>
<td>Restart the resource or move it to a new location.</td>
</tr>
<tr>
<td>hard</td>
<td>A non-transient error occurred. The error may be specific to the current node.</td>
<td>Move the resource elsewhere and prevent it from being retried on the current node.</td>
</tr>
<tr>
<td>fatal</td>
<td>A non-transient error occurred that will be common to all cluster nodes. This means a bad configuration was specified.</td>
<td>Stop the resource and prevent it from being started on any cluster node.</td>
</tr>
</tbody>
</table>

Assuming an action is considered to have failed, the following table outlines the different OCF return codes. It also shows the type of recovery the cluster will initiate when the respective error code is received.

<table>
<thead>
<tr>
<th>OCF Return Code</th>
<th>OCF Alias</th>
<th>Description</th>
<th>Recovery Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OCF_SUCCESS</td>
<td>Success. The command completed successfully. This is the expected result for all start, stop, promote and demote commands.</td>
<td>soft</td>
</tr>
<tr>
<td>1</td>
<td>OCF_ERR_GENERIC</td>
<td>Generic “there was a problem” error code.</td>
<td>soft</td>
</tr>
<tr>
<td>OCF Return Code</td>
<td>OCF Alias</td>
<td>Description</td>
<td>Recovery Type</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>2</td>
<td>OCF_ERR_ARGS</td>
<td>The resource's configuration is not valid on this machine (for example, it refers to a location/tool not found on the node).</td>
<td>hard</td>
</tr>
<tr>
<td>3</td>
<td>OCF_ERR_UNIMPLEMENTED</td>
<td>The requested action is not implemented.</td>
<td>hard</td>
</tr>
<tr>
<td>4</td>
<td>OCF_ERR_PERM</td>
<td>The resource agent does not have sufficient privileges to complete the task.</td>
<td>hard</td>
</tr>
<tr>
<td>5</td>
<td>OCF_ERR_INSTALLED</td>
<td>The tools required by the resource are not installed on this machine.</td>
<td>hard</td>
</tr>
<tr>
<td>6</td>
<td>OCF_ERR_CONFIGURED</td>
<td>The resource's configuration is invalid (for example, required parameters are missing).</td>
<td>fatal</td>
</tr>
<tr>
<td>7</td>
<td>OCF_NOT_RUNNING</td>
<td>The resource is not running. The cluster will not attempt to stop a resource that returns this for any action. This OCF return code may or may not require resource recovery—it depends on what is the expected resource status. If unexpected, then soft recovery.</td>
<td>N/A</td>
</tr>
<tr>
<td>8</td>
<td>OCF_RUNNING_MASTER</td>
<td>The resource is running in Master mode.</td>
<td>soft</td>
</tr>
<tr>
<td>9</td>
<td>OCF_FAILED_MASTER</td>
<td>The resource is in Master mode but has failed. The resource will be demoted, stopped and then started (and possibly promoted) again.</td>
<td>soft</td>
</tr>
<tr>
<td>other</td>
<td>N/A</td>
<td>Custom error code.</td>
<td>soft</td>
</tr>
</tbody>
</table>
10 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 the 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.

10.1 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**

Resource level fencing ensures exclusive access to a given resource. Common examples of this are changing the zoning of the node from a SAN fiber channel switch (thus locking the node out of access to its disks) or methods like SCSI reserve. For examples, refer to Section 11.10, “Additional Mechanisms for Storage Protection”.

**Node Level Fencing**

Node level fencing prevents a failed node from accessing shared resources entirely. This is usually done in a simple and abrupt way: reset or power off the node.
FIGURE 10.1: CLASSES OF FENCING
10.2 Node Level Fencing

In a Pacemaker cluster, the implementation of node level fencing is STONITH (Shoot The Other Node in the Head). 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.

10.2.1 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 one of the following commands on any of the nodes:

```
root # stonith -L
```
or
```
root # crm ra list stonith
```

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 if a utility power failure occurs.

**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 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. Before 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.

10.2.2 STONITH Implementation

The STONITH implementation of SUSE® Linux Enterprise High Availability Extension consists of two components:

pacemaker-fenced

pacemaker-fenced 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 pacemaker-fenced daemon runs on every node in the High Availability cluster. The pacemaker-fenced instance running on the DC node receives a fencing request from the pacemaker-controld. It is up to this and other pacemaker-fenced 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. The STONITH plug-ins contained in the cluster-glue package reside in /usr/lib64/stonith/plugins on each node. (If you installed the fence-agents package, too, the plug-ins contained there are installed in /usr/sbin/fence_*) All STONITH plug-ins look the same to pacemaker-fenced, 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.
10.3 STONITH Resources and Configuration

To set up fencing, you need to configure one or more STONITH resources—the pacemaker-fenced daemon requires no configuration. All configuration is stored in the CIB. A STONITH resource is a resource of class stonith (see Section 6.3.2, “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, 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 like any other resource. For details how to do so with your preferred cluster management tool:

- **Hawk2**: Section 7.5.6, “Adding STONITH Resources”
- **crmsh**: Section 8.3.4, “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:

```bash
stonith -t stonith-device-type -n
```

For example, to view the parameters for the ibmhmc device type, enter the following:

```bash
stonith -t ibmhmc -n
```

To get a short help text for the device, use the -h option:

```bash
stonith -t stonith-device-type -h
```

### 10.3.1 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:

```bash
root # crm < sample.txt
```
For more information about configuring resources with the `crm` command line tool, refer to Chapter 8, Configuring and Managing Cluster Resources (Command Line).

**EXAMPLE 10.1: CONFIGURATION OF AN IBM RSA LIGHTS-OUT DEVICE**

An IBM RSA lights-out device might be configured like this:

```plaintext
configure
primitive st-ibmrsa-1 stonith:external/ibmrsa-telnet \
params nodename=alice ip_address=192.168.0.101 \
username=USERNAME password=PASSW0RD
primitive st-ibmrsa-2 stonith:external/ibmrsa-telnet \
params nodename=bob ip_address=192.168.0.102 \
username=USERNAME password=PASSW0RD
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, `pacemaker-fenced` refuses to terminate its host.

**EXAMPLE 10.2: CONFIGURATION OF A 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, usually 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:

```plaintext
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

For Config info [-p] syntax, give each of the above parameters in order as the -p value. Arguments are separated by white space.
Config file [-F] syntax is the same as -p, except # at the start of a line denotes a comment.

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 APC UPS product lines.

EXAMPLE 10.3: CONFIGURATION OF A KDUMP DEVICE

Kdump belongs to the Special Fencing Devices and is in fact the opposite of a fencing device. The 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 Kdump plug-in must be used in concert with another, real STONITH device, for example, external/ipmi. For the fencing mechanism to work properly, you must specify that Kdump is checked before a real STONITH device is triggered. Use crm configure fencing_topology to specify the order of the fencing devices as shown in the following procedure.
1. Use the `stonith:fence_kdump` resource agent (provided by the package `fence-agents`) to monitor all nodes with the Kdump function enabled. Find a configuration example for the resource below:

```bash
configure
primitive st-kdump stonith:fence_kdump
    params nodename="alice "
    pmk_host_check="static-list" 
    pmk_reboot_action="off" 
    pmk_monitor_action="metadata" 
    pmk_reboot_retries="1" 
    timeout="60"
commit
```

1. Name of the node to be monitored. If you need to monitor more than one node, configure more STONITH resources. To prevent a specific node from using a fencing device, add location constraints.

The fencing action will be started after the timeout of the resource.

2. In `/etc/sysconfig/kdump` on each node, configure `KDUMP_POSTSCRIPT` to send a notification to all nodes when the Kdump process is finished. For example:

```
/usr/lib/fence_kdump_send -i INTERVAL -p PORT -c 1 alice bob charlie [...] 
```

The node that does a Kdump will restart automatically after Kdump has finished.

3. Write a new `initrd` to include the library `fence_kdump_send` with network enabled. Use the `-f` option to overwrite the existing file, so the new file will be used for the next boot process:

```
root # dracut -f -a kdump
```

4. Open a port in the firewall for the `fence_kdump` resource. The default port is `7410`.

5. To achieve that Kdump is checked before triggering a real fencing mechanism (like `external/ipmi`), use a configuration similar to the following:

```bash
fencing_topology \\
alice: kdump-node1 ipmi-node1 \\
bob: kdump-node2 ipmi-node2
```
For more details on `fencing_topology`:

```
crm configure help fencing_topology
```

10.4 Monitoring Fencing Devices

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 usually be enough. 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 8.3.9, "Configuring Resource Monitoring" for the command line approach.

10.5 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 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**

---

**fence_kdump**

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 configuration details, see Example 10.3, “Configuration of a Kdump Device”.

**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 11, Procedure 11.7, “Configuring the Cluster to Use SBD”. 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.
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.

**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.

**Diskless SBD**

This configuration is useful if you want a fencing mechanism without shared storage. In this diskless mode, SBD fences nodes by using the hardware watchdog without relying on any shared device. However, diskless SBD cannot handle a split brain scenario for a two-node cluster. Use this option only for clusters with more than two nodes.

suicide is the only exception to the “I do not shoot my host” rule.

### 10.6 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 take 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 6.3.9, “Timeout Values”.
• Use appropriate fencing devices for your setup. For details, also refer to Section 10.5, “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 start-up, the node will be fenced once to clarify its status.

10.7 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 High Availability Linux Project.

http://www.clusterlabs.org/pacemaker/doc/

• Pacemaker Explained: Explains the concepts used to configure Pacemaker. Contains comprehensive and very detailed information for reference.


Article explaining the concepts of split brain, quorum and fencing in HA clusters.
11 Storage Protection and SBD

SBD (STONITH Block Device) provides a node fencing mechanism for Pacemaker-based clusters through the exchange of messages via shared block storage (SAN, iSCSI, FCoE, etc.). This isolates the fencing mechanism from changes in firmware version or dependencies on specific firmware controllers. SBD needs a watchdog on each node to ensure that misbehaving nodes are really stopped. Under certain conditions, it is also possible to use SBD without shared storage, by running it in diskless mode.

The ha-cluster-bootstrap scripts provide an automated way to set up a cluster with the option of using SBD as fencing mechanism. For details, see the Article “Installation and Setup Quick Start”. However, manually setting up SBD provides you with more options regarding the individual settings.

This chapter explains the concepts behind SBD. It guides you through configuring the components needed by SBD to protect your cluster from potential data corruption in case of a split brain scenario.

In addition to node level fencing, you can use additional mechanisms for storage protection, such as LVM2 exclusive activation or OCFS2 file locking support (resource level fencing). They protect your system against administrative or application faults.

11.1 Conceptual Overview

SBD expands to Storage-Based Death or STONITH Block Device.

The highest priority of the High Availability cluster stack is to protect the integrity of data. This is achieved by preventing uncoordinated concurrent access to data storage. The cluster stack takes care of this using several control mechanisms.

However, network partitioning or software malfunction could potentially cause scenarios where several DCs are elected in a cluster. If this so-called split brain scenario were allowed to unfold, data corruption might occur.
Node fencing via STONITH is the primary mechanism to prevent this. Using SBD as a node fencing mechanism is one way of shutting down nodes without using an external power off device in case of a split brain scenario.

SBD COMPONENTS AND MECHANISMS

SBD Partition

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 (for example, 1 MB for standard SCSI disks with 512 byte block size or 4 MB for DASD disks with 4 kB block size). The initialization process creates a message layout on the device with slots for up to 255 nodes.

SBD Daemon

After the respective SBD 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.

Messages

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.

Also, 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 workload will terminate anyway if the storage connectivity has been lost.

Watchdog

Whenever SBD is used, a correctly working watchdog is crucial. Modern systems support a hardware watchdog that needs to be “tickled” or “fed” by a software component. The software component (in this case, the SBD daemon) “feeds” the watchdog by regularly writing 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 I/O error.
If Pacemaker integration is activated, SBD will not self-fence if device majority is lost. For example, your cluster contains three nodes: A, B, and C. Because of 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 because of 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 nodes B and C would continue to run.

11.2 Overview of Manually Setting Up SBD

The following steps are necessary to manually set up storage-based protection. They must be executed as root. Before you start, check Section 11.3, “Requirements”.

1. Setting Up the Watchdog

2. Depending on your scenario, either use SBD with one to three devices or in diskless mode. For an outline, see Section 11.4, “Number of SBD Devices”. The detailed setup is described in:
   - Setting Up SBD with Devices
   - Setting Up Diskless SBD

3. Testing SBD and Fencing

11.3 Requirements

- You can use up to three SBD devices for storage-based fencing. When using one to three devices, the shared storage must be accessible from all nodes.

- The path to the shared storage device must be persistent and consistent across all nodes in the cluster. Use stable device names such as /dev/disk/by-id/dm-uuid-part1-mpath-abcedf12345.

- The shared storage can be connected via Fibre Channel (FC), Fibre Channel over Ethernet (FCoE), or even iSCSI.

- The shared storage segment must not use host-based RAID, LVM2, or DRBD*. DRBD can be split, which is problematic for SBD, as there cannot be two states in SBD. Cluster multi-device (Cluster MD) cannot be used for SBD.
• However, using storage-based RAID and multipathing is recommended for increased reliability.

• An SBD device can be shared between different clusters, as long as no more than 255 nodes share the device.

• For clusters with more than two nodes, you can also use SBD in diskless mode.

11.4 Number of SBD Devices

SBD supports the use of up to three 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 because of failures or maintenance. SBD will terminate itself only if more than one device is lost and if required, depending on the status of the cluster partition or node. If at least two devices are still accessible, fencing messages can be successfully transmitted. 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.

Diskless
This configuration is useful if you want a fencing mechanism without shared storage. In this diskless mode, SBD fences nodes by using the hardware watchdog without relying on any shared device. However, diskless SBD cannot handle a split brain scenario for a two-node cluster. Use this option only for clusters with more than two nodes.
11.5 Calculation of Timeouts

When using SBD as a fencing mechanism, it is vital to consider the timeouts of all components, because they depend on each other.

**Watchdog Timeout**

This timeout is set during initialization of the SBD device. It depends mostly on your storage latency. The majority of devices must be successfully read within this time. Otherwise, the node might self-fence.

**Note: Multipath or iSCSI Setup**

If your SBD device(s) reside on a multipath setup or iSCSI, the timeout should be set to the time required to detect a path failure and switch to the next path. This also means that in `/etc/multipath.conf` the value of `max_polling_interval` must be less than `watchdog` timeout.

**msgwait Timeout**

This timeout is set during initialization of the SBD device. It defines the time after which a message written to a node's slot on the SBD device is considered delivered. The timeout should be long enough for the node to detect that it needs to self-fence. However, if the `msgwait` timeout is relatively long, a fenced cluster node might rejoin before the fencing action returns. This can be mitigated by setting the `SBD_DELAY_START` parameter in the SBD configuration, as described in Procedure 11.4 in Step 4.

**stonith-timeout in the CIB**

This timeout is set in the CIB as a global cluster property. It defines how long to wait for the STONITH action (reboot, on, off) to complete.

**stonith-watchdog-timeout in the CIB**

This timeout is set in the CIB as a global cluster property. If not set explicitly, it defaults to 0, which is appropriate for using SBD with one to three devices. For use of SBD in diskless mode, see Procedure 11.8, “Configuring Diskless SBD” for more details.

If you change the watchdog timeout, you need to adjust the other two timeouts as well. The following “formula” expresses the relationship between these three values:

**EXAMPLE 11.1: FORMULA FOR TIMEOUT CALCULATION**

```
Timeout (msgwait) >= (Timeout (watchdog) * 2)
```
For example, if you set the watchdog timeout to 120, set the msgwait timeout to 240 and the stonith-timeout to 288.

If you use the ha-cluster-bootstrap scripts to set up a cluster and to initialize the SBD device, the relationship between these timeouts is automatically considered.

11.6 Setting Up the Watchdog

SUSE Linux Enterprise High Availability Extension ships with several kernel modules that provide hardware-specific watchdog drivers. For a list of the most commonly used ones, see Commonly Used Watchdog Drivers.

For clusters in production environments we recommend to use a hardware-specific watchdog driver. However, if no watchdog matches your hardware, softdog can be used as kernel watchdog module.

The High Availability Extension uses the SBD daemon as the software component that “feeds” the watchdog.

11.6.1 Using a Hardware Watchdog

Finding the right watchdog kernel module for a given system is not trivial. Automatic probing fails very often. As a result, lots of modules are already loaded before the right one gets a chance. Table 11.1 lists the most commonly used watchdog drivers. If your hardware is not listed there, the directories /lib/modules/KERNEL_VERSION/kernel/drivers/watchdog or /lib/modules/KERNEL_VERSION/kernel/drivers/ipmi give you a list of choices, too. Alternatively, ask your hardware or system vendor for details on system specific watchdog configuration.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Driver</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP</td>
<td>hpwdt</td>
</tr>
<tr>
<td>Dell, Lenovo (Intel TCO)</td>
<td>iTCO_wdt</td>
</tr>
<tr>
<td>Fujitsu</td>
<td>ipmi_watchdog</td>
</tr>
<tr>
<td>Hardware</td>
<td>Driver</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>VM on z/VM on IBM mainframe</td>
<td>vmwatchdog</td>
</tr>
<tr>
<td>Xen VM (DomU)</td>
<td>xen_xdt</td>
</tr>
<tr>
<td>Generic</td>
<td>softdog</td>
</tr>
</tbody>
</table>

Important: Accessing the Watchdog Timer
Some hardware vendors ship systems management software that uses the watchdog for system resets (for example, HP ASR daemon). If the watchdog is used by SBD, disable such software. No other software must access the watchdog timer.

PROCEDURE 11.1: LOADING THE CORRECT KERNEL MODULE
To make sure the correct watchdog module is loaded, proceed as follows:

1. List the drivers that have been installed with your kernel version:

   ```
   root # rpm -ql kernel-VERSION | grep watchdog
   ```

2. List any watchdog modules that are currently loaded in the kernel:

   ```
   root # lsmod | egrep "(wd|dog)"
   ```

3. If you get a result, unload the wrong module:

   ```
   root # rmmod WRONG_MODULE
   ```

4. Enable the watchdog module that matches your hardware:

   ```
   root # echo WATCHDOG_MODULE > /etc/modules-load.d/watchdog.conf
   root # systemctl restart systemd-modules-load
   ```

5. Test whether the watchdog module is loaded correctly:

   ```
   root # lsmod | grep dog
   ```

6. Verify if the watchdog device is available and works:

   ```
   root # ls -l /dev/watchdog*
   root # sbd query-watchdog
   ```
If your watchdog device is not available, stop here and check the module name and options. Maybe use another driver.

7. Verify if the watchdog device works:

```bash
root # sbd -w WATCHDOG_DEVICE test-watchdog
```

8. Reboot your machine to make sure there are no conflicting kernel modules. For example, if you find the message `cannot register ...` in your log, this would indicate such conflicting modules. To ignore such modules, refer to https://documentation.suse.com/sles/html/SLES-all/cha-mod.html#sec-mod-modprobe-blacklist.

11.6.2 Using the Software Watchdog (softdog)

For clusters in production environments we recommend to use a hardware-specific watchdog driver. However, if no watchdog matches your hardware, `softdog` can be used as kernel watchdog module.

⚠️ Important: Softdog Limitations

The softdog driver assumes that at least one CPU is still running. If all CPUs are stuck, the code in the softdog driver that should reboot the system will never be executed. In contrast, hardware watchdogs keep working even if all CPUs are stuck.

PROCEDURE 11.2: LOADING THE SOFTDOG KERNEL MODULE

1. Enable the softdog driver:

```bash
root # echo softdog > /etc/modules-load.d/watchdog.conf
```

2. Add the `softdog` module in `/etc/modules-load.d/watchdog.conf` and restart a service:

```bash
root # echo softdog > /etc/modules-load.d/watchdog.conf
root # systemctl restart systemd-modules-load
```

3. Test whether the softdog watchdog module is loaded correctly:

```bash
root # lsmod | grep softdog
```
11.7 Setting Up SBD with Devices

The following steps are necessary for setup:

1. **Initializing the SBD Devices**
2. **Editing the SBD Configuration File**
3. **Enabling and Starting the SBD Service**
4. **Testing the SBD Devices**
5. **Configuring the Cluster to Use SBD**

Before you start, make sure the block device or devices you want to use for SBD meet the requirements specified in Section 11.3.

When setting up the SBD devices, you need to take several timeout values into account. For details, see Section 11.5, “Calculation of Timeouts”.

The node will terminate itself if the SBD daemon running on it has not updated the watchdog timer fast enough. After having set the timeouts, test them in your specific environment.

**PROCEDURE 11.3: INITIALIZING THE SBD DEVICES**

To use SBD with shared storage, you must first create the messaging layout on one to three block devices. The `sbd create` command will write a metadata header to the specified device or devices. It will also initialize the messaging slots for up to 255 nodes. If executed without any further options, the command will use the default timeout settings.

**Warning: Overwriting Existing Data**

Make sure the device or devices you want to use for SBD do not hold any important data. When you execute the `sbd create` command, roughly the first megabyte of the specified block devices will be overwritten without further requests or backup.

1. Decide which block device or block devices to use for SBD.

2. Initialize the SBD device with the following command:

   ```
   root # sbd -d /dev/SBD create
   ```

   (Replace `/dev/SBD` with your actual path name, for example: `/dev/disk/by-id/scsi-ST2000DM001-0123456_Wabcdefg`.)
To use more than one device for SBD, specify the \texttt{-d} option multiple times, for example:

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

3. If your SBD device resides on a multipath group, use the \texttt{-1} and \texttt{-4} options to adjust the timeouts to use for SBD. For details, see \textit{Section 11.5, “Calculation of Timeouts”}. All timeouts are given in seconds:

```
root # sbd -d /dev/SBD -4 180 -1 90 create
```

\begin{itemize}
\item The \texttt{-4} option is used to specify the \texttt{msgwait} timeout. In the example above, it is set to 180 seconds.
\item The \texttt{-1} option is used to specify the \texttt{watchdog} timeout. In the example above, it is set to 90 seconds. The minimum allowed value for the emulated watchdog is 15 seconds.
\end{itemize}

4. Check what has been written to the device:

```
root # sbd -d /dev/SBD dump
Header version : 2.1
UUID            : 619127f4-0e06-434c-84a0-ea82036e144c
Number of slots : 255
Sector size     : 512
Timeout (watchdog) : 5
Timeout (allocate) : 2
Timeout (loop)  : 1
Timeout (msgwait) : 10
==Header on disk /dev/SBD is dumped
```

As you can see, the timeouts are also stored in the header, to ensure that all participating nodes agree on them.

After you have initialized the SBD devices, edit the SBD configuration file, then enable and start the respective services for the changes to take effect.

\textbf{PROCEDURE 11.4: EDITING THE SBD CONFIGURATION FILE}

1. Open the file \texttt{/etc/sysconfig/sbd} and use the following entries:

```
SBD_PACEMAKER=yes
SBD_STARTMODE=always
SBD_DELAY_START=no
SBD_WATCHDOG_DEV=/dev/watchdog
SBD_WATCHDOG_TIMEOUT=5
```
The `SBD_DEVICE` entry is not needed as no shared disk is used. When this parameter is missing, the `sbd` service does not start any watcher process for SBD devices.

2. Search for the following parameter: `SBD DEVICE`.
   It specifies the devices to monitor and to use for exchanging SBD messages.

3. Edit this line by replacing `SBD` with your SBD device:

   ```
   SBD_DEVICE="/dev/SBD"
   ```

   If you need to specify multiple devices in the first line, separate them with semicolons (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 cluster start-up.

4. Search for the following parameter: `SBD_DELAY_START`.
   Enables or disables a delay. Set `SBD_DELAY_START` to `yes` if `msgwait` is relatively long, but your cluster nodes boot very fast. Setting this parameter to `yes` delays the start of SBD on boot. This is sometimes necessary with virtual machines.

After you have added your SBD devices to the SBD configuration file, enable the SBD daemon. The SBD daemon is a critical piece of the cluster stack. It needs to be running when the cluster stack is running. Thus, the `sbd` service is started as a dependency whenever the cluster services are started.

PROCEDURE 11.5: ENABLING AND STARTING THE SBD SERVICE

1. On each node, enable the SBD service:

   ```
   root # systemctl enable sbd
   ```

   It will be started together with the Corosync service whenever the cluster services are started.

2. Restart the cluster services on each node:

   ```
   root #crm cluster restart
   ```

   This automatically triggers the start of the SBD daemon.

As a next step, test the SBD devices as described in Procedure 11.6.
PROCEDURE 11.6: TESTING THE SBD DEVICES

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. For example, if you have a two-node cluster, the message slot should show `clear` for both nodes:

```
0 alice clear
1 bob clear
```

2. Try sending a test message to one of the nodes:

```
root # sbd -d /dev/SBD message alice test
```

3. The node will acknowledge the receipt of the message in the system log files:

```
May 03 16:08:31 alice sbd[66139]: /dev/SBD: notice: servant: Received command test from bob on disk /dev/SBD
```

This confirms that SBD is indeed up and running on the node and that it is ready to receive messages.

As a final step, you need to adjust the cluster configuration as described in Procedure 11.7.

PROCEDURE 11.7: CONFIGURING THE CLUSTER TO USE SBD

To configure the use of SBD in the cluster, you need to do the following in the cluster configuration:

- Set the `stonith-timeout` parameter to a value that matches your setting.
- Configure the SBD STONITH resource.

For the calculation of the `stonith-timeout` refer to Section 11.5, “Calculation of Timeouts”.

1. Start a shell and log in as `root` or equivalent.

2. Run `crm configure`.

3. Enter the following:

```
crm(live)configure# property stonith-enabled="true" ①
crm(live)configure# property stonith-watchdog-timeout=0 ②
```
This is the default configuration, because clusters without STONITH are not supported. But in case STONITH has been deactivated for testing purposes, make sure this parameter is set to `true` again.

If not explicitly set, this value defaults to `0`, which is appropriate for use of SBD with one to three devices.

A `stonith-timeout` value of `40` would be appropriate if the `msgwait` timeout value for SBD was set to `30` seconds.

4. For a two-node cluster, in case of split-brain, there will be fencing issued from each node to the other as expected. To prevent both nodes from being reset at practically the same time, it is recommended to apply the following fencing delays to help one of the nodes, or even the preferred node, win the fencing match. For clusters with more than two nodes, you do not need to apply these delays.

**Priority fencing delay**

The `priority-fencing-delay` cluster property is disabled by default. By configuring a delay value, if the other node is lost and it has the higher total resource priority, the fencing targeting it will be delayed for the specified amount of time. This means that in case of split-brain, the more important node wins the fencing match.

Resources that matter can be configured with priority meta attribute. On calculation, the priority values of the resources or instances that are running on each node are summed up to be accounted. A promoted resource instance takes the configured base priority plus one, so that it receives a higher value than any unpromoted instance.

```
root #crm configure property priority-fencing-delay=30
```

Even if `priority-fencing-delay` is used, we still recommend also using `pcmk_delay_base` or `pcmk_delay_max` as described below to address any situations where the nodes happen to have equal priority. The value of `priority-fencing-delay` should be significantly greater than the maximum of `pcmk_delay_base /pcmk_delay_max`, and preferably twice the maximum.

**Predictable static delays**

This parameter adds a static delay before executing STONITH actions. To prevent the nodes from getting reset at the same time under split-brain of a two-node cluster, configure separate fencing resources with different delay values. The preferred node
can be marked with the parameter to be targeted with a longer fencing delay, so that it wins any fencing match. To make this succeed, it is essential to create two primitive STONITH devices for each node. In the following configuration, alice will win and survive in case of a split brain scenario:

```
crm(live)configure# primitive st-sbd-alice stonith:external/sbd params 
    pcmk_host_list=alice pcmk_delay_base=20

crm(live)configure# primitive st-sbd-bob stonith:external/sbd params 
    pcmk_host_list=bob pcmk_delay_base=0
```

**Dynamic Random Delays**

This parameter adds a random delay for STONITH actions on the fencing device. Rather than a static delay targeting a specific node, the parameter `pcmk_delay_max` adds a random delay for any fencing with the fencing resource to prevent double reset. Unlike `pcmk_delay_base`, this parameter can be specified for an unified fencing resource targeting multiple nodes.

```
crm(live)configure# primitive stonith_sbd stonith:external/sbd params pcmk_delay_max=30
```

5. Review your changes with `show`.

6. Submit your changes with `commit` and leave the crm live configuration with `quit`.

After the resource has started, your cluster is successfully configured for use of SBD. It will use this method in case a node needs to be fenced.

### 11.8 Setting Up Diskless SBD

SBD can be operated in a diskless mode. In this mode, a watchdog device will be used to reset the node in the following cases: if it loses quorum, if any monitored daemon is lost and not recovered, or if Pacemaker decides that the node requires fencing. Diskless SBD is based on “self-fencing” of a node, depending on the status of the cluster, the quorum and some reasonable assumptions. No STONITH SBD resource primitive is needed in the CIB.
Important: Number of Cluster Nodes

Do not use diskless SBD as a fencing mechanism for two-node clusters. Use diskless SBD only for clusters with three or more nodes. SBD in diskless mode cannot handle split brain scenarios for two-node clusters. If you want to use diskless SBD for two-node clusters, use QDevice as described in Chapter 12, QDevice and QNetd.

PROCEDURE 11.8: CONFIGURING DISKLESS SBD

1. Open the file /etc/sysconfig/sbd and use the following entries:

   SBD_PACEMAKER=yes
   SBD_STARTMODE(always
   SBD_DELAY_START=no
   SBD_WATCHDOG_DEV=/dev/watchdog
   SBD_WATCHDOG_TIMEOUT=5

   The SBD_DEVICE entry is not needed as no shared disk is used. When this parameter is missing, the sbd service does not start any watcher process for SBD devices.

2. On each node, enable the SBD service:

   root # systemctl enable sbd

   It will be started together with the Corosync service whenever the cluster services are started.

3. Restart the cluster services on each node:

   root # crm cluster restart

   This automatically triggers the start of the SBD daemon.

4. Check if the parameter have-watchdog=true has been automatically set:

   root # crm configure show | grep have-watchdog
   have-watchdog=true

5. Run crm configure and set the following cluster properties on the crm shell:

   crm(live)configure# property stonith-enabled="true" 1
   crm(live)configure# property stonith-watchdog-timeout=10 2
This is the default configuration, because clusters without STONITH are not supported. But in case STONITH has been deactivated for testing purposes, make sure this parameter is set to true again.

For diskless SBD, this parameter must not equal zero. It defines after how long it is assumed that the fencing target has already self-fenced. Therefore its value needs to be >= the value of SBD_WATCHDOG_TIMEOUT in /etc/sysconfig/sbd. Starting with SUSE Linux Enterprise High Availability Extension 15, if you set stonith-watchdog-timeout to a negative value, Pacemaker will automatically calculate this timeout and set it to twice the value of SBD_WATCHDOG_TIMEOUT.

6. Review your changes with show.

7. Submit your changes with commit and leave the crm live configuration with quit.

11.9 Testing SBD and Fencing

To test whether SBD works as expected for node fencing purposes, use one or all of the following methods:

Manually Triggering Fencing of a Node

To trigger a fencing action for node NODENAME:

```
root # crm node fence NODENAME
```

Check if the node is fenced and if the other nodes consider the node as fenced after the stonith-watchdog-timeout.

Simulating an SBD Failure

1. Identify the process ID of the SBD inquisitor:

```
root # systemctl status sbd
● sbd.service - Shared-storage based fencing daemon
    Loaded: loaded (/usr/lib/systemd/system/sbd.service; enabled; vendor preset: disabled)
    Active: active (running) since Tue 2018-04-17 15:24:51 CEST; 6 days ago
    Docs: man:sbd(8)
    Process: 1844 ExecStart=/usr/sbin/sbd $SBD_OPTS -p /var/run/sbd.pid watch (code=exited, status=0/SUCCESS)
    Main PID: 1859 (sbd)
    Tasks: 4 (limit: 4915)
```
2. Simulate an SBD failure by terminating the SBD inquisitor process. In our example, the process ID of the SBD inquisitor is 1859:

```
root # kill -9 1859
```

The node proactively self-fences. The other nodes notice the loss of the node and consider it has self-fenced after the `stonith-watchdog-timeout`.

### Triggering Fencing through a Monitor Operation Failure

With a normal configuration, a failure of a resource *stop operation* will trigger fencing. To trigger fencing manually, you can produce a failure of a resource stop operation. Alternatively, you can temporarily change the configuration of a resource *monitor operation* and produce a monitor failure as described below:

1. Configure an `on-fail=fence` property for a resource monitor operation:

```
op monitor interval=10 on-fail=fence
```

2. Let the monitoring operation fail (for example, by terminating the respective daemon, if the resource relates to a service). This failure triggers a fencing action.

### 11.10 Additional Mechanisms for Storage Protection

Apart from node fencing via STONITH there are other methods to achieve storage protection at a resource level. For example, SCSI-3 and SCSI-4 use persistent reservations whereas `sfex` provides a locking mechanism. Both methods are explained in the following subsections.

#### 11.10.1 Configuring an sg_persist Resource

The SCSI specifications 3 and 4 define *persistent reservations*. These are SCSI protocol features and can be used for I/O fencing and failover. This feature is implemented in the `sg_persist` Linux command.
Note: SCSI Disk Compatibility

Any backing disks for \texttt{sg\_persist} must be SCSI disk compatible. \texttt{sg\_persist} only works for devices like SCSI disks or iSCSI LUNs. Do \textit{not} use it for IDE, SATA, or any block devices which do not support the SCSI protocol.

Before you proceed, check if your disk supports persistent reservations. Use the following command (replace \texttt{DISK} with your device name):

\begin{verbatim}
root # sg\_persist -n --in --read-reservation -d /dev/DISK
\end{verbatim}

The result shows whether your disk supports persistent reservations:

- Supported disk:
  
  PR generation=0x0, there is NO reservation held

- Unsupported disk:
  
  PR in (Read reservation): command not supported
  Illegal request, Invalid opcode

If you get an error message (like the one above), replace the old disk with an SCSI compatible disk. Otherwise proceed as follows:

1. To create the primitive resource \texttt{sg\_persist}, run the following commands as \texttt{root}:

\begin{verbatim}
root # crm configure
crm(live)configure\# primitive sg sg\_persist \ 
  params devs="/dev/sdc" reservation_type=3 \ 
  op monitor interval=60 timeout=60
\end{verbatim}

2. Add the \texttt{sg\_persist} primitive to a master-slave group:

\begin{verbatim}
crm(live)configure\# ms ms-sg sg \ 
  meta master-max=1 notify=true
\end{verbatim}

3. Do some tests. When the resource is in master/slave status, you can mount and write on \texttt{/dev/sdc1} on the cluster node where the master instance is running, while you cannot write on the cluster node where the slave instance is running.
4. Add a file system primitive for Ext4:

```bash
crm(live)configure# primitive ext4 Filesystem \ 
   params device="/dev/sdc1" directory="/mnt/ext4" ftype=ext4
```

5. Add the following order relationship plus a collocation between the `sg_persist` master and the file system resource:

```bash
crm(live)configure# order o-ms-sg-before-ext4 Mandatory: ms-sg:promote ext4:start

crm(live)configure# colocation col-ext4-with-sg-persist inf: ext4 ms-sg:Master
```

6. Check all your changes with the `show changed` command.

7. Commit your changes.

For more information, refer to the `sg_persist` man page.

---

11.10.2 Ensuring Exclusive Storage Activation with sfex

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. As sfex requires shared storage, it is recommended that the SBD node fencing mechanism described above is used on another partition of the storage.

By design, sfex cannot be used with workloads that require concurrency (such as OCFS2). It serves as a layer of protection for classic failover style workloads. This is similar to an SCSI-2 reservation in effect, but more general.

#### 11.10.2.1 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. 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.
11.10.2.2 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, and needs 1 KB of storage space allocated per lock. By default, `sfex_init` creates one lock on the partition.

### Important: Requirements

- The shared partition for sfex should be on the same logical unit as the data you want to protect.
- The shared sfex partition must not use host-based RAID, nor DRBD.
- Using an LVM2 logical volume is possible.

#### PROCEDURE 11.9: 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 metadata with the following command:

   ```
   root # sfex_init -n 1 /dev/sfex
   ```

3. Verify that the metadata has been created correctly:

   ```
   root # sfex_stat -i 1 /dev/sfex ; echo $?
   ```

   This should return 2, since the lock is not currently held.

#### PROCEDURE 11.10: 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 an sfex lock, create mandatory ordering and placement constraints between the resources to protect the sfex resource. If the resource to be protected has the ID `filesystem1`:

```
crm(live)configure# order order-sfex-1 Mandatory: sfex_1 filesystem1
crm(live)configure# colocation col-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
```

11.11 For More Information

- `man sbd`
- `http://www.linux-ha.org/wiki/SBD_Fencing`
12  QDevice and QNetd

QDevice and QNetd participate in quorum decisions. With assistance from the arbitrator `corosync-qnetd`, `corosync-qdevice` provides a configurable number of votes, allowing a cluster to sustain more node failures than the standard quorum rules allow. We recommend deploying `corosync-qnetd` and `corosync-qdevice` for clusters with an even number of nodes, and especially for two-node clusters.

12.1  Conceptual Overview

In comparison to calculating quora among cluster nodes, the QDevice-and-QNetd approach has the following benefits:

- It provides better sustainability in case of node failures.
- You can write your own heuristics scripts to affect votes. This is especially useful for complex setups, such as SAP applications.
- It enables you to configure a QNetd server to provide votes for multiple clusters.
- Allows using diskless SBD for two-node clusters.
- It helps with quorum decisions for clusters with an even number of nodes under split-brain situations, especially for two-node clusters.

A setup with QDevice/QNetd consists of the following components and mechanisms:

**QDEVICE/QNETD COMPONENTS AND MECHANISMS**

**QNetd (`corosync-qnetd`)**

A systemd service (a daemon, the “QNetd server”) which is not part of the cluster. The systemd service provides a vote to the `corosync-qdevice` daemon. To improve security, `corosync-qnetd` can work with TLS for client certificate checking.

**QDevice (`corosync-qdevice`)**

A systemd service (a daemon) on each cluster node running together with Corosync. This is the client of `corosync-qnetd`. Its primary use is to allow a cluster to sustain more node failures than standard quorum rules allow.

QDevice is designed to work with different arbitrators. However, currently, only QNetd is supported.
Algorithms

QDevice supports different algorithms which determine the behaviour how votes are assigned. Currently, the following exist:

- **FFSplit** (“fifty-fifty split” is the default. It is used for clusters with an even number of nodes. If the cluster splits into two similar partitions, this algorithm provides one vote to one of the partitions, based on the results of heuristics checks and other factors.

- **LMS** (“last man standing”) allows the only remaining node that can see the QNetd server to get the votes. So this algorithm is useful when a cluster with only one active node should remain quorate.

Heuristics

QDevice supports a set of commands (“heuristics”). The commands are executed locally on startup of cluster services, cluster membership change, successful connect to `corosync-qnetd`, or optionally, at regular times. The heuristics can be set with the `quorum.device.heuristics` key (in the `corosync.conf` file) or with the `--qdevice-heuristics-mode` option. Both know the values `off` (default), `sync`, and `on`. The difference between `sync` and `on` is you can additionally execute the above commands regularly. Only if all commands executed successfully are the heuristics considered to have passed; otherwise, they failed. The heuristics' result is sent to `corosync-qnetd` where it is used in calculations to determine which partition should be quorate.

Tiebreaker

This is used as a fallback if the cluster partitions are completely equal even with the same heuristics results. It can be configured to be the lowest, the highest, or a specific node ID.

12.2 Requirements and Prerequisites

Before setting up QDevice and QNetd, you need to prepare the environment as the following:

- In addition to the cluster nodes, you have a separate machine which will become the QNetd server. See Section 12.3, “Setting Up the QNetd Server”.

- A different physical network than the one that Corosync uses. It is recommended for QDevice to reach the QNetd server. Ideally, the QNetd server should be in a separate rack than the main cluster, or at least on a separate PSU and not in the same network segment as the corosync ring or rings.
12.3 Setting Up the QNetd Server

The QNetd server is not part of the cluster stack, it is also not a real member of your cluster. As such, you cannot move resources to this server.

The QNetd server is almost “state free”. Usually, you do not need to change anything in the configuration file `/etc/sysconfig/corosync-qnetd`. By default, the `corosync-qnetd` service runs the daemon as user `coroqnetd` in the group `coroqnetd`. This avoids running the daemon as `root`.

To create a QNetd server, proceed as follows:

1. On the machine that will become the QNetd server, install SUSE Linux Enterprise Server 15 SP1.
2. Enable the SUSE Linux Enterprise High Availability Extension using the command listed in `SUSEConnect --list-extensions`.
3. Install the `corosync-qnetd` package:

   ```
   root # zypper install corosync-qnetd
   ```

   You do not need to manually start the `corosync-qnetd` service. It will be started automatically when you configure QDevice on the cluster.

Your QNetd server is ready to accept connections from a QDevice client `corosync-qdevice`. Further configuration is not needed.

12.4 Connecting QDevice Clients to the QNetd Server

After you set up your QNetd server, you can set up and run the clients. You can connect the clients to the QNetd server during the installation of your cluster, or you can add them later. This procedure documents how to add them later.

1. On all nodes, install the `corosync-qdevice` package:

   ```
   root # zypper install corosync-qdevice
   ```

2. On one of the nodes, run the following command to configure QDevice:

   ```
   root #crm cluster init qdevice
   ```

   Do you want to configure QDevice (y/n)? y

   HOST or IP of the QNetd server to be used [/]QNETD_SERVER
12.5 Setting Up a QDevice with Heuristics

If you need additional control over how votes are determined, use heuristics. Heuristics are a set of commands which are executed in parallel.

For this purpose, the command `crm cluster init qdevice` provides the option `--qdevice-heuristics`. You can pass one or more commands (separated by semicolon) with absolute paths.

For example, if your own command for heuristic checks is located at `/usr/sbin/my-script.sh` you can run it on one of your cluster nodes as follows:

```
root # crm cluster init qdevice --qdevice-hostname=charlie \ 
    --qdevice-heuristics=/usr/sbin/my-script.sh \ 
    --qdevice-heuristics-mode=on
```

The command or commands can be written in any language such as Shell, Python, or Ruby. If they succeed, they return 0 (zero), otherwise they return an error code.

You can also pass a set of commands. Only when all commands finish successfully (return code is zero), the heuristics have passed.

The `--qdevice-heuristics-mode=on` option lets the heuristics commands run regularly.

12.6 Checking and Showing Quorum Status

You can query the quorum status on one of your cluster nodes as shown in Example 12.1, "Status of QDevice". It shows the status of your QDevice nodes.

Example 12.1: Status of QDevice

```
root # corosync-quorumtool
```
Quorum information
---------------------
Date:             ...
Quorum provider:  corosync_votequorum
Nodes:            2
Node ID:          3232235777
Ring ID:          3232235777/8
Quorate:          Yes

Votequorum information
----------------------
Expected votes:   3
Highest expected: 3
Total votes:      3
Quorum:           2
Flags:            Quorate Qdevice

Membership information
----------------------
<table>
<thead>
<tr>
<th>Nodeid</th>
<th>Votes</th>
<th>Qdevice Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>3232235777</td>
<td>1</td>
<td>A,V,NMW 192.168.1.1 (local)</td>
</tr>
<tr>
<td>3232235778</td>
<td>1</td>
<td>A,V,NMW 192.168.1.2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Qdevice</td>
</tr>
</tbody>
</table>

As an alternative with an identical result, you can also use the `crm corosync status quorum` command.

The number of expected nodes we are expecting. In this example, it is a two-node cluster.

As the node ID is not explicitly specified in `corosync.conf` this ID is a 32-bit integer representation of the IP address. In this example, the value `3232235777` stands for the IP address `192.168.1.1`.

The quorum status. In this case, the cluster has quorum.

The status for each cluster node means:

**A** (Alive) or **NA** (not alive)

Shows the connectivity status between QDevice and Corosync. If there is a heartbeat between QDevice and Corosync, it is shown as alive (A).

**V** (Vote) or **NV** (non vote)

Shows if the quorum device has given a vote (letter V) to the node. A letter V means that both nodes can communicate with each other. In a split-brain situation, one node would be set to V and the other node would be set to NV.
MW (Master wins) or NMW (not master wins)

Shows if the quorum device master_wins flag is set. By default, the flag is not set, so you see NMW (not master wins) See the man page votequrorum_qdevice_master_wins(3) for more information.

NR (not registered)

Shows that the cluster is not using a quorum device.

If you want to query the status of the QNetd server, you get a similar output as shown in Example 12.2, “Status of QNetd Server”:

EXAMPLE 12.2: STATUS OF QNETD SERVER

```
root # corosync-qnetd-tool
Cluster "hacluster":
  Algorithm:      Fifty-Fifty split
  Tie-breaker:    Node with lowest node ID
Node ID 3232235777:
  Client address: ::ffff:192.168.1.1:54732
  HB interval:    8000ms
  Configured node list: 3232235777, 3232235778
  Ring ID:        aa10ab0.8
  Membership node list: 3232235777, 3232235778
  Heuristics:     Undefined (membership: Undefined, regular: Undefined)
  TLS active:     Yes (client certificate verified)
  Vote:           ACK (ACK)
Node ID 3232235778:
  Client address: ::ffff:192.168.1.2:43016
  HB interval:    8000ms
  Configured node list: 3232235777, 3232235778
  Ring ID:        aa10ab0.8
  Membership node list: 3232235777, 3232235778
  Heuristics:     Undefined (membership: Undefined, regular: Undefined)
  TLS active:     Yes (client certificate verified)
  Vote:           No change (ACK)
```

1 As an alternative with an identical result, you can also use the `crm corosync status qnetd` command.

2 The name of your cluster as set in the configuration file `/etc/corosync/corosync.conf` in the `totem.cluster_name` section.

3 The algorithm currently used. In this example, it is `FFSplit`.

4 This is the entry for the node with the IP address `192.168.1.1`. TLS is active.
12.7 For More Information

For additional information about QDevice and QNtd Man pages of corosync-qdevice(8), corosync-qnetd(8).
13 Access Control Lists

The cluster administration tools like crm shell (crmsh) or Hawk2 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 use 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.

Note: CIB Syntax Validation Version and ACL Differences

This ACL documentation only applies if your CIB is validated with the CIB syntax version pacemaker-2.0 or higher. For details on how to check this and upgrade the CIB version, see Note: Upgrading the CIB Syntax Version.

13.1 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.

## 13.2 Conceptual Overview

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 consists of the following:

- an access right like **read**, **write**, or **deny**
- a specification where to apply the rule. This specification can be a type, an ID reference, or an XPath expression. XPath is a language for selecting nodes in an XML document. Refer to [http://en.wikipedia.org/wiki/XPath](http://en.wikipedia.org/wiki/XPath).

Usually, it is convenient to bundle ACLs into roles and assign a specific role to system users (ACL targets). There are these methods to create ACL roles:

- **Section 13.7, “Setting ACL Rules via XPath Expressions”**. You need to know the structure of the underlying XML to create ACL rules.
- **Section 13.8, “Setting ACL Rules via Abbreviations”**. Create a shorthand syntax and ACL rules to apply to the matched objects.

## 13.3 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 Hawk2 as described in Procedure 13.1, “Enabling Use of ACLs with Hawk2”.

---

**PROCEDURE 13.1: ENABLING USE OF ACLS WITH HAWK2**

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. In the left navigation bar, select Cluster Configuration to display the global cluster options and their current values.

3. Below Cluster Configuration click the empty drop-down box and select enable-acl to add the parameter. It is added with its default value No.

4. Set its value to Yes and apply your changes.

---

### 13.4 Creating a Read-Only Monitor Role

The following subsections describe how to configure read-only access by defining a monitor role either in Hawk2 or crm shell.

---

#### 13.4.1 Creating a Read-Only Monitor Role with Hawk2

The following procedures show how to configure 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 13.4, “Adding a Monitor Role and Assigning a User with crmsh”.

**PROCEDURE 13.2: ADDING A MONITOR ROLE WITH HAWK2**

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. In the left navigation bar, select Roles.

3. Click Create.

4. Enter a unique Role ID, for example, monitor.

5. As access Right, select Read.
6. As Xpath, enter the XPath expression /cib.

7. Click Create.
   This creates a new role with the name monitor, sets the read rights and applies this to all elements in the CIB by using the XPath expression /cib.

8. If necessary, add more rules by clicking the plus icon and specifying the respective parameters.

9. Sort the individual rules by using the arrow up or down buttons.

PROCEDURE 13.3: ASSIGNING A ROLE TO A TARGET WITH HAWK2

To assign the role we created in Procedure 13.2 to a system user (target), proceed as follows:

1. Log in to Hawk2:
   
   https://HAWKSERVER:7630/

2. In the left navigation bar, select Targets.

3. To create a system user (ACL Target), click Create and enter a unique Target ID, for example, tux. Make sure this user belongs to the haclient group.

4. To assign a role to the target, select one or multiple Roles.
   In our example, select the monitor role you created in Procedure 13.2.
5. Confirm your choice.

To configure access rights for resources or constraints, you can also use the abbreviated syntax as explained in Section 13.8, “Setting ACL Rules via Abbreviations”.

### 13.4.2 Creating a Read-Only Monitor Role 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 13.4: 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 the 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 ACL targets, which are the corresponding system users. Make sure they belong to the `haclient` group.

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

5. Check your changes:

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

6. Commit your changes:

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

To configure access rights for resources or constraints, you can also use the abbreviated syntax as explained in Section 13.8, “Setting ACL Rules via Abbreviations”.

13.5 Removing a User

The following subsections describe how to remove an existing user from ACL either in Hawk2 or crmsh.

13.5.1 Removing a User with Hawk2

To remove a user from ACL, proceed as follows:

1. Log in to Hawk2:

   ```plaintext
   https://HAWKSERVER:7630/
   ```

2. In the left navigation bar, select `Targets`.

3. To remove a system user (ACL target), click the garbage bin icon under the `Operations` column.
4. Confirm the dialog box.

### 13.5.2 Removing a User with crmsh

To remove a user from ACL, replace the placeholder `USER` with the name of the user:

```
root #crm configure delete USERNAME
```

As an alternative, you can use the `edit` subcommand:

```
root #crm configure edit USERNAME
```

### 13.6 Removing an Existing Role

The following subsections describe how to remove an existing role in either Hawk2 or crmsh.

#### Note: Deleting Roles with Referenced Users

Keep in mind, no user should belong to this role. If there is still a reference to a user in the role, the role cannot be deleted. Delete the references to the users first before you delete the role.

### 13.6.1 Removing an Existing Role with Hawk2

To remove a role, proceed as follows:

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. In the left navigation bar, select `Roles`.

3. To remove a role, click the garbage bin icon under the `Operations` column.

4. Confirm the dialog box. If an error message appears, make sure your role is “empty” and does not reference users.
13.6.2 Removing an Existing Role with crmsh

To remove an existing role, replace the placeholder `ROLE` with the name of the role:

```
root #crm configure delete ROLE
```

13.7 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 13.1):

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

**EXAMPLE 13.1: EXCERPT OF A CLUSTER CONFIGURATION IN XML**

```
<cib>
  <!-- ... -->
  <configuration>
    <crm_config>
      <cluster_property_set id="cib-bootstrap-options">
        <nvpair name="stonith-enabled" value="true" id="cib-bootstrap-options-stonith-enabled"/>
      </cluster_property_set>
      [...]
    </crm_config>
  </configuration>
  <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 13.1, “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 cannot reconfigure any resources (for example, change parameters or operations), nor change the configuration of colocation or ordering constraints.

### TABLE 13.1: OPERATOR ROLE—ACCESS TYPES AND XPATH EXPRESSIONS

<table>
<thead>
<tr>
<th>Type</th>
<th>XPath/Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write</td>
<td>//crm_config//nvpair[@name='maintenance-mode']</td>
</tr>
<tr>
<td></td>
<td>Turn cluster maintenance mode on or off.</td>
</tr>
<tr>
<td>Write</td>
<td>//op_defaults//nvpair[@name='record-pending']</td>
</tr>
<tr>
<td></td>
<td>Choose whether pending operations are recorded.</td>
</tr>
<tr>
<td>Write</td>
<td>//nodes/node//nvpair[@name='standby']</td>
</tr>
<tr>
<td></td>
<td>Set node in online or standby mode.</td>
</tr>
<tr>
<td>Write</td>
<td>//resources//nvpair[@name='target-role']</td>
</tr>
<tr>
<td></td>
<td>Start, stop, promote or demote any resource.</td>
</tr>
<tr>
<td>Write</td>
<td>//resources//nvpair[@name='maintenance']</td>
</tr>
<tr>
<td></td>
<td>Select if a resource should be put to maintenance mode or not.</td>
</tr>
<tr>
<td>Write</td>
<td>//constraints/rsc_location</td>
</tr>
<tr>
<td></td>
<td>Migrate/move resources from one node to another.</td>
</tr>
<tr>
<td>Read</td>
<td>/cib</td>
</tr>
<tr>
<td></td>
<td>View the status of the cluster.</td>
</tr>
</tbody>
</table>
13.8 Setting ACL Rules via Abbreviations

For users who do not want to deal with the XML structure there is an easier method. For example, consider the following XPath:

```
//*[@id="rsc1"]
```

which locates all the XML nodes with the ID _rsc1_. The abbreviated syntax is written like this:

```
ref:"rsc1"
```

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

```
//constraints/rsc_location
```

The abbreviated syntax is written like this:

```
type:"rsc_location"
```

The abbreviated syntax can be used in crmsh and Hawk2. The CIB daemon knows how to apply the ACL rules to the matching objects.
14 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 device will become active if the primary device 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.

14.1 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 System > 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).
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. To select the Ethernet devices that you want to include into the bond, activate the check box in front of the respective devices.
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 lose 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`.

14.2 Hotplugging devices into a bond

Sometimes it is necessary to replace an interface in a bond with another one, for example, if the respective network device constantly fails. The solution is to set up hotplugging. It is also necessary to change the `udev` rules 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 14.1: HOTPLUGGING DEVICES INTO A BOND WITH YAST

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

1. Start YaST as `root` and select System → Network Settings.

2. In the Network Settings, switch to the Overview tab, which shows the already configured devices. If devices are already configured in a bond, 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 devices, but for the bond to become ready, which needs at least one available device. When one of the 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.

14.3 For More Information

All modes and many options are explained in detail in the Linux Ethernet Bonding Driver HOWTO. The file can be found at `/usr/src/linux/Documentation/networking/bonding.txt` after you have installed the package `kernel-source`.

For High Availability setups, the following options described therein are especially important: `miimon` and `use_carrier`.

15 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.

15.1 Conceptual Overview

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 fewer 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.

On the other hand, Linux Virtual Server includes two different software: IPVS (IP Virtual Server) and KTCPVS (Kernel TCP Virtual Server). IPVS provides layer 4 load balancing whereas KTCPVS provides layer 7 load balancing.

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

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 host name. With DNS round robin, the DNS server will return all of them in a rotating order. Thus different clients will see different addresses.

- **Selecting the “best” server.** Although this has several drawbacks, 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.

- **Geo Location.** It is 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 is not adequate and vice versa:

- **SSL termination.** The front-end load balancers can handle the SSL layer. 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 failover that is transparent to the client and server.
15.2 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.

15.2.1 Director

The main component of LVS is the ip_vs (or IPVS) Kernel code. It is part of the default Kernel and 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).

15.2.2 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 (see below) specifies the virtual services and their associated real servers and tells ldirectord how to configure the server as an 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.
15.2.3 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. They 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. Thus, 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 need to be in the same physical network segment.

15.2.4 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.

More detailed description about Linux Virtual Server scheduling algorithms can be found at [http://kb.linuxvirtualserver.org/wiki/IPVS](http://kb.linuxvirtualserver.org/wiki/IPVS). Furthermore, search for `--scheduler` in the `ipvsadm` man page.
Related load balancing strategies for HAPerxy can be found at http://www.haproxy.org/download/1.6/doc/configuration.txt.

15.2.5 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 default installation does not include the configuration file /etc/ha.d/ldirectord.cf. This file is created by the YaST module. 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 15.1, “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 15.1: 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 host name 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 e-mail 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 whether 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 log files in *Log File*. By default, ldirectord writes its log files to /var/log/ldirectord.log.

---

**FIGURE 15.1: YAST IP LOAD BALANCING—GLOBAL PARAMETERS**

---
PROCEDURE 15.2: 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 host name 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 host names) of the servers, the ports (or service names) and the forwarding method. The forwarding method must either be gate, ipip or masq, see Section 15.2.3, “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 box.

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`.

![YAST IP Load Balancing—Virtual Services](image)

**FIGURE 15.2: YAST IP LOAD BALANCING—VIRTUAL SERVICES**

**EXAMPLE 15.1: SIMPLE LDIRECTORD CONFIGURATION**

The values shown in Figure 15.1, “YaST IP Load Balancing—Global Parameters” and Figure 15.2, “YaST IP Load Balancing—Virtual Services”, would lead to the following configuration, defined in `/etc/ha.d/ldirectord.cf`:

```plaintext
autoreload = yes  
    checkinterval = 5  
    checktimeout = 3  
    quiescent = yes  
    virtual = 192.168.0.200:80  
    checktype = negotiate  
    fallback = 127.0.0.1:80  
    protocol = tcp  
    real = 192.168.0.110:80 gate  
    real = 192.168.0.120:80 gate  
    receive = “still alive”
```
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. Instead 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.
15.2.6 Further Setup

Apart from the configuration of \texttt{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 \texttt{ldirectord}, so that \texttt{ldirectord} can fail over to other servers in case of hardware failure.
- As the backup of the load balancer also needs the \texttt{ldirectord} configuration file to fulfill its task, make sure the \texttt{/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 4.5, “Transferring the Configuration to All Nodes”.

15.3 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 back-end servers. It is configured as active/passive, meaning if one server fails, the passive server becomes active. In such a scenario, the user will not notice any interruption.

In this section, we will use the following setup:

- A load balancer, with the IP address 192.168.1.99.
- A virtual, floating IP address 192.168.1.99.
- Our servers (usually for Web content) \texttt{www.example1.com} (IP: 192.168.1.200) and \texttt{www.example2.com} (IP: 192.168.1.201)

To configure HAProxy, use the following procedure:

1. Install the \texttt{haproxy} package.
2. Create the file `/etc/haproxy/haproxy.cfg` with the following contents:

```plaintext
global
  maxconn 256
  daemon

defaults
  log     global
  mode    http
  option  httplog
  option  dontlognull
  retries 3
  option redispatch
  maxconn 2000
  timeout connect 5000
  timeout client 50s
  timeout server 50000

frontend LB
  bind 192.168.1.99:80
  reqadd X-Forwarded-Proto:\ http
  default_backend LB

backend LB
  mode http
  stats enable
  stats hide-version
  stats uri /stats
  stats realm Haproxy\ Statistics
  stats auth haproxy:password
  balance roundrobin
  option httpclose
  option forwardfor
  cookie LB insert
  option httpchk GET /robots.txt HTTP/1.0
  server web1-srv 192.168.1.200:80 cookie web1-srv check
  server web2-srv 192.168.1.201:80 cookie web2-srv check
```

1. 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.
Section which sets default parameters for all other sections following its declaration. Some important lines:

```
redispatch
   Enables or disables 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 that 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.
```

3. The maximum time to wait for a connection attempt to a server to succeed.

4. The maximum time of inactivity on the client side.

5. The maximum time of inactivity on the server side.

6. Section which combines front-end and back-end 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.
```

7. Credentials for HAProxy Statistic report page.

8. Load balancing will work in a round-robin process.

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 the 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`. For details, see the Installation and Setup Quick Start.

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 start timeout=120 interval=0 \ 
    op stop timeout=120 interval=0 \ 
    op monitor timeout=100 interval=5s \ 
    meta target-role=Started
crm(haproxy-config)# primitive vip IPaddr2 \ 
    params ip=192.168.1.99 nic=eth0 cidr_netmask=23 broadcast=192.168.1.255 \ 
    op monitor interval=5s timeout=120 on-fail=restart
crm(haproxy-config)# group g-haproxy vip 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
```
15.4 For More Information

- [http://wwwhaproxy.org](http://wwwhaproxy.org)


- For more information about `ldirectord`, refer to its comprehensive man page.

- LVS Knowledge Base: [http://kb.linuxvirtualserver.org/wiki/Main_Page](http://kb.linuxvirtualserver.org/wiki/Main_Page)
Apart from local clusters and metro area clusters, SUSE® Linux Enterprise High Availability Extension 15 SP1 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**. For details on how to use and set up Geo clusters, refer to Article “Geo Clustering Quick Start” and Book “Geo Clustering Guide”.
17 Executing Maintenance Tasks

To perform maintenance tasks on the cluster nodes, you might need to stop the resources running on that node, to move them, or to shut down or reboot the node. It might also be necessary to temporarily take over the control of resources from the cluster, or even to stop the cluster service while resources remain running.

This chapter explains how to manually take down a cluster node without negative side-effects. It also gives an overview of different options the cluster stack provides for executing maintenance tasks.

17.1 Implications of Taking Down a Cluster Node

In SUSE Linux Enterprise Server High Availability Extension 15 SP2, the way cluster services were started and stopped has changed. SUSE recommends using the crm shell. The old commands with `systemctl` are still available, but are recommended for experienced users only. For details, refer to the SUSE blog article [https://www.suse.com/c/suse-high-availability-cluster-services-how-to-stop-start-or-view-the-status/](https://www.suse.com/c/suse-high-availability-cluster-services-how-to-stop-start-or-view-the-status/).

The recommended way to start, stop, or view the status is:

```bash
crm cluster start
Start cluster services on one node

crm cluster stop
Stop cluster services on one node

crm cluster restart
Restart cluster services on one node

crm cluster status
View status of the cluster stack on one node
```

Execute the above commands as user `root` or as a user who has the required privileges.
When you shut down or reboot a cluster node (or stop the cluster services on a node), the following processes will be triggered:

- The resources that are running on the node will be stopped or moved off the node.
- If stopping the resources should fail or time out, the STONITH mechanism will fence the node and shut it down.

**PROCEDURE 17.1: MANUALLY REBOOTING A CLUSTER NODE**

If your aim is to move the services off the node in an orderly fashion before shutting down or rebooting the node, proceed as follows:

1. On the node you want to reboot or shut down, log in as `root` or equivalent.
2. Put the node into **standby** mode:
   
   ```shell
   root # crm -w node standby
   ```
   
   That way, services can migrate off the node without being limited by the shutdown timeout of the cluster services.
3. Check the cluster status with:
   
   ```shell
   root # crm status
   ```
   
   It shows the respective node in **standby** mode:
   
   ```
   [...] 
   Node bob: standby 
   [...] 
   ```
4. Stop the cluster services on that node:
   
   ```shell
   root # crm cluster stop
   ```
5. Reboot the node.

To check if the node joins the cluster again:

1. Log in to the node as `root` or equivalent.
2. Check if the cluster services has started:
   
   ```shell
   root # crm cluster status
   ```
3. If not, start it:

```
root # crm cluster start
```

4. Check the cluster status with:

```
root # crm status
```

It should show the node coming online again.

17.2 Different Options for Maintenance Tasks

Pacemaker offers a variety of options for performing system maintenance:

**Putting the Cluster in Maintenance Mode**

The global cluster property `maintenance-mode` puts all resources into maintenance state at once. The cluster stops monitoring them and becomes oblivious to their status. Note that only the resource management by Pacemaker is disabled. Corosync and SBD are still functional. Use maintenance mode for any tasks involving cluster resources. For any tasks involving infrastructure such as storage or networking, the safest method is to stop the cluster services completely. See Section 17.1, “Implications of Taking Down a Cluster Node”.

**Putting a Node in Maintenance Mode**

This option allows you to put all resources running on a specific node into maintenance state at once. The cluster will cease monitoring them and thus become oblivious to their status.

**Putting a Node in Standby Mode**

A node that is in standby mode can no longer run resources. Any resources running on the node will be moved away or stopped (in case no other node is eligible to run the resource). Also, all monitoring operations will be stopped on the node (except for those with `role="Stopped"`).

You can use this option if you need to stop a node in a cluster while continuing to provide the services running on another node.

**Putting a Resource into Maintenance Mode**

When this mode is enabled for a resource, no monitoring operations will be triggered for the resource.
Use this option if you need to manually touch the service that is managed by this resource and do not want the cluster to run any monitoring operations for the resource during that time.

**Putting a Resource into Unmanaged Mode**

The `is-managed` meta attribute allows you to temporarily “release” a resource from being managed by the cluster stack. This means you can manually touch the service that is managed by this resource (for example, to adjust any components). However, the cluster will continue to monitor the resource and to report any failures.

If you want the cluster to also cease monitoring the resource, use the per-resource maintenance mode instead (see *Putting a Resource into Maintenance Mode*).

### 17.3 Preparing and Finishing Maintenance Work

**Warning: Risk of Data Loss**

If you need to do testing or maintenance work, follow the general steps below. Otherwise you risk unwanted side effects, like resources not starting in an orderly fashion, unsynchronized CIBs across the cluster nodes, or even data loss.

1. Before you start, choose which of the options outlined in Section 17.2 is appropriate for your situation.

2. Apply this option with Hawk2 or crmsh.

3. Execute your maintenance task or tests.

4. After you have finished, put the resource, node or cluster back to “normal” operation.
17.4 Putting the Cluster in Maintenance Mode

**Warning: Maintenance mode only disables Pacemaker**

When putting a cluster into maintenance mode, only the resource management by Pacemaker is disabled. Corosync and SBD are still functional. Depending on your maintenance tasks, this might lead to fence operations.

Use maintenance mode for any tasks involving cluster resources. For any tasks involving infrastructure such as storage or networking, the safest method is to stop the cluster services completely. See Section 17.1, “Implications of Taking Down a Cluster Node”.

To put the cluster in maintenance mode on the crm shell, use the following command:

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

To put the cluster back to normal mode after your maintenance work is done, use the following command:

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

**PROCEDURE 17.2: PUTTING THE CLUSTER IN MAINTENANCE MODE WITH HAWK2**

1. Start a Web browser and log in to the cluster as described in Section 7.2, “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, activate the check box next to maintenance-mode and confirm your changes.

5. After you have finished the maintenance task for the whole cluster, deactivate the check box next to the maintenance-mode attribute.

   From this point on, High Availability Extension will take over cluster management again.
17.5 Putting a Node in Maintenance Mode

To put a node in maintenance mode on the crm shell, use the following command:

```
root # crm node maintenance NODENAME
```

To put the node back to normal mode after your maintenance work is done, use the following command:

```
root # crm node ready NODENAME
```

PROCEDURE 17.3: PUTTING A NODE IN MAINTENANCE MODE WITH HAWK2

1. Start a Web browser and log in to the cluster as described in Section 7.2, “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.
4. After you have finished your maintenance task, click the wrench icon next to the node and select Ready.

17.6 Putting a Node in Standby Mode

To put a node in standby mode on the crm shell, use the following command:

```
root # crm node standby NODENAME
```

To bring the node back online after your maintenance work is done, use the following command:

```
root # crm node online NODENAME
```

PROCEDURE 17.4: PUTTING A NODE IN STANDBY MODE WITH HAWK2

1. Start a Web browser and log in to the cluster as described in Section 7.2, “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 Standby.
4. Finish the maintenance task for the node.

5. To deactivate the standby mode, click the wrench icon next to the node and select *Ready*.

### 17.7 Putting a Resource into Maintenance Mode

To put a resource into maintenance mode on the crm shell, use the following command:

```bash
root # crm resource maintenance RESOURCE_ID true
```

To put the resource back into normal mode after your maintenance work is done, use the following command:

```bash
root # crm resource maintenance RESOURCE_ID false
```

**PROCEDURE 17.5: PUTTING A RESOURCE INTO MAINTENANCE MODE WITH HAWK2**

1. Start a Web browser and log in to the cluster as described in *Section 7.2, “Logging In”*.

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

3. 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*.

4. Open the *Meta Attributes* category.

5. From the empty drop-down list, select the *maintenance* attribute and click the plus icon to add it.

6. Activate the check box next to *maintenance* to set the maintenance attribute to *yes*.

7. Confirm your changes.

8. After you have finished the maintenance task for that resource, deactivate the check box next to the *maintenance* attribute for that resource.

   From this point on, the resource will be managed by the High Availability Extension software again.
17.8 Putting a Resource into Unmanaged Mode

To put a resource into unmanaged mode on the crm shell, use the following command:

```
root # crm resource unmanage RESOURCE_ID
```

To put it into managed mode again after your maintenance work is done, use the following command:

```
root # crm resource manage RESOURCE_ID
```

PROCEDURE 17.6: PUTTING A RESOURCE INTO UNMANAGED MODE WITH HAWK2

1. Start a Web browser and log in to the cluster as described in Section 7.2, “Logging In”.

2. From the left navigation bar, select Status and go to the Resources list.

3. In the Operations column, click the arrow down icon next to the resource you want to modify and select Edit.
   The resource configuration screen opens.

4. Below Meta Attributes, select the is-managed entry from the empty drop-down box.

5. Set its value to No and click Apply.

6. After you have finished your maintenance task, set is-managed to Yes (which is the default value) and apply your changes.
   From this point on, the resource will be managed by the High Availability Extension software again.

17.9 Rebooting a Cluster Node While in Maintenance Mode

Note: Implications

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

If you want to take down a node while either the cluster or the node is in maintenance mode, proceed as follows:

1. On the node you want to reboot or shut down, log in as `root` or equivalent.

2. If you have a DLM resource (or other resources depending on DLM), make sure to explicitly stop those resources before stopping the cluster services:

   ```bash
   crm(live)resource# stop RESOURCE_ID
   ```

   The reason is that stopping Pacemaker also stops the Corosync service, on whose membership and messaging services DLM depends. If Corosync stops, the DLM resource will assume a split brain scenario and trigger a fencing operation.

3. Stop the cluster services on that node:

   ```bash
   root # crm cluster stop
   ```

4. Shut down or reboot the node.
III Storage and Data Replication

18 Distributed Lock Manager (DLM) 265
19 OCFS2 267
20 GFS2 277
21 DRBD 282
22 Cluster Logical Volume Manager (Cluster LVM) 301
23 Cluster Multi-device (Cluster MD) 316
24 Samba Clustering 321
25 Disaster Recovery with Rear (Relax-and-Recover) 330
18 Distributed Lock Manager (DLM)

The Distributed Lock Manager (DLM) in the kernel is the base component used by OCFS2, GFS2, Cluster MD, and Cluster LVM (lvmlockd) to provide active-active storage at each respective layer.

18.1 Protocols for DLM Communication

To avoid single points of failure, redundant communication paths are important for High Availability clusters. This is also true for DLM communication. If network bonding (Link Aggregation Control Protocol, LACP) cannot be used for any reason, we highly recommend to define a redundant communication channel (a second ring) in Corosync. For details, see Procedure 4.3, “Defining a Redundant Communication Channel”.

Depending on the configuration in `/etc/corosync/corosync.conf`, DLM then decides whether to use the TCP or SCTP protocol for its communication:

- If `rrp_mode` is set to `none` (which means redundant ring configuration is disabled), DLM automatically uses TCP. However, without a redundant communication channel, DLM communication will fail if the TCP link is down.

- If `rrp_mode` is set to `passive` (which is the typical setting), and a second communication ring in `/etc/corosync/corosync.conf` is configured correctly, DLM automatically uses SCTP. In this case, DLM messaging has the redundancy capability provided by SCTP.

18.2 Configuring DLM Cluster Resources

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

Note: DLM Resource for Several Solutions

As OCFS2, GFS2, Cluster MD, and Cluster LVM (lvmlockd) all use DLM, it is enough to configure one resource for DLM. As the DLM resource runs on all nodes in the cluster it is configured as a clone resource.
If you have a setup that includes both OCFS2 and Cluster LVM, configuring one DLM resource for both OCFS2 and Cluster LVM is enough.

**PROCEDURE 18.1: CONFIGURING A BASE GROUP FOR DLM**

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 afterward (for both OCFS2 and Cluster LVM, 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 simplifies the overall setup as you do not need to specify several individual groups, clones and their dependencies.

Follow the steps below on 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 and further storage-related resources:

   ```
  crm(live)configure# group g-storage dlm
   ```
5. Clone the `g-storage` group so that it runs on all nodes:

   ```
  crm(live)configure# clone cl-storage g-storage \ 
   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 `quit`.

**Note: Failure When Disabling STONITH**

Clusters without STONITH are not supported. If you set the global cluster option `stonith-enabled` to `false` for testing or troubleshooting purposes, the DLM resource and all services depending on it (such as Cluster LVM, GFS2, and OCFS2) will fail to start.
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).

19.1 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.

Note: Support for OCFS2
OCFS2 is only supported by SUSE when used with the pcmk (Pacemaker) stack, as provided by SUSE Linux Enterprise High Availability Extension. SUSE does not provide support for OCFS2 in combination with the o2cb stack.

19.2 OCFS2 Packages and Management Utilities

The OCFS2 Kernel module (ocfs2) is installed automatically in the High Availability Extension on SUSE® Linux Enterprise Server 15 SP1. To use OCFS2, make sure the following packages are installed on each node in the cluster: ocfs2-tools and the matching ocfs2-kmp-* packages for your Kernel.

The ocfs2-tools package provides the following utilities for management of OFS2 volumes. For syntax information, see their man pages.

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

<table>
<thead>
<tr>
<th>OCFS2 Utility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mounted.ocfs2</td>
<td>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.</td>
</tr>
<tr>
<td>tunefs.ocfs2</td>
<td>Changes OCFS2 file system parameters, including the volume label, number of node slots, journal size for all node slots, and volume size.</td>
</tr>
</tbody>
</table>

### 19.3 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.

The following procedure uses the `crm` shell to configure the cluster resources. Alternatively, you can also use Hawk2 to configure the resources as described in Section 19.6, “Configuring OCFS2 Resources With Hawk2”.

**PROCEDURE 19.1: CONFIGURING A STONITH RESOURCE**

danger: 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 Procedure 11.3, “Initializing the SBD Devices”.
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:

```bash
crm(live)configure# primitive sbd_stonith stonith:external/sbd
```
5. Review your changes with `show`.

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

For details on configuring the resource group for DLM, see Procedure 18.1, “Configuring a Base Group for DLM”.

### 19.4 Creating OCFS2 Volumes

After you have configured a DLM cluster resource as described in Section 19.3, “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 19.2, “Creating and Formatting an OCFS2 Volume”. The most important parameters for the command are listed in Table 19.2, “Important OCFS2 Parameters”. For more information and the command syntax, refer to the `mkfs.ocfs2` man page.

<table>
<thead>
<tr>
<th>OCFS2 Parameter</th>
<th>Description and Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume Label (-L)</td>
<td>A descriptive name for the volume to make it uniquely identifiable when it is mounted on different nodes. Use the <code>tunefs.ocfs2</code> utility to modify the label as needed.</td>
</tr>
<tr>
<td>OCFS2 Parameter</td>
<td>Description and Recommendation</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Cluster Size ( -C )</td>
<td>Cluster size is the smallest unit of space allocated to a file to hold the data. For the available options and recommendations, refer to the <code>mkfs.ocfs2</code> man page.</td>
</tr>
<tr>
<td>Number of Node Slots ( -N )</td>
<td>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. Nodes that access the volume can be a combination of little-endian architectures (such as AMD64/Intel 64) and big-endian architectures (such as S/390x). Node-specific files are called local files. A node slot number is appended to the local file. For example: <code>journal:0000</code> belongs to whatever node is assigned to slot number 0. Set each volume's maximum number of node slots when you create it, according to how many nodes that you expect to concurrently mount the volume. Use the <code>tunefs.ocfs2</code> utility to increase the number of node slots as needed. Note that the value cannot be decreased, and one node slot will consume about 100 MiB disk space. In case the -N parameter is not specified, the number of node slots is decided based on the size of the file system. For the default value, refer to the <code>mkfs.ocfs2</code> man page.</td>
</tr>
<tr>
<td>OCFS2 Parameter</td>
<td>Description and Recommendation</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Block Size ( -b )</td>
<td>The smallest unit of space addressable by the file system. Specify the block size when you create the volume. For the available options and recommendations, refer to the <code>mkfs.ocfs2</code> man page.</td>
</tr>
<tr>
<td>Specific Features On/Off (--fs-features)</td>
<td>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.</td>
</tr>
<tr>
<td>Pre-Defined Features (--fs-feature-level)</td>
<td>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.</td>
</tr>
</tbody>
</table>

If you do not specify any 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 19.2: 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
   ```
19.5 Mounting OCFS2 Volumes

You can either mount an OCFS2 volume manually or with the cluster manager, as described in Procedure 19.4, “Mounting an OCFS2 Volume with the Cluster Resource Manager”.

PROCEDURE 19.3: 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 19.4: 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 Hawk2 to configure the resources as described in Section 19.6, “Configuring OCFS2 Resources With Hawk2”.

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" \ 
   op start timeout="60" op stop timeout="60" \ 
   meta target-role="Started"
   ```

4. Add the ocfs2-1 primitive to the g-storage group you created in Procedure 18.1, “Configuring a Base Group for DLM”.

   ```
   crm(live)configure# modgroup g-storage add ocfs2-1
   ```
The **add** subcommand appends the new group member by default. Because of the base group’s internal colocation and ordering, Pacemaker will only start the `ocfs2-1` resource on nodes that also have a `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 `quit`.

### 19.6 Configuring OCFS2 Resources With Hawk2

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 Hawk2’s *Setup Wizard*.

**Important: Differences Between Manual Configuration and Hawk2**

The OCFS2 template in the *Setup Wizard* does not include the configuration of 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 19.1, “Configuring a STONITH Resource”*.

Using the OCFS2 template in the Hawk2 *Setup Wizard* also leads to a slightly different resource configuration than the manual configuration described in *Procedure 18.1, “Configuring a Base Group for DLM”* and *Procedure 19.4, “Mounting an OCFS2 Volume with the Cluster Resource Manager”*.

**PROCEDURE 19.5: CONFIGURING OCFS2 RESOURCES WITH HAWK2’S WIZARD**

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

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

3. Expand the *File System* category and select *OCFS2 File System*.

4. Follow the instructions on the screen. If you need information about an option, click it to display a short help text in Hawk2. After the last configuration step, Verify the values you have entered.
The wizard displays the configuration snippet that will be applied to the CIB and any additional changes, if required.

5. Check the proposed changes. If everything is according to your wishes, apply the changes. A message on the screen shows if the action has been successful.

19.7 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 `tunefs.ocfs2`.

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

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

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

For more information about OCFS2, see the following links:

https://ocfs2.wiki.kernel.org/ — The OCFS2 project home page.


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.

### 20.1 GFS2 Packages and Management Utilities

To use GFS2, make sure `gfs2-utils` and a 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 20.1: GFS2 UTILITIES**

<table>
<thead>
<tr>
<th>GFS2 Utility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fsck.gfs2</td>
<td>Checks the file system for errors and optionally repairs errors.</td>
</tr>
<tr>
<td>gfs2_jadd</td>
<td>Adds additional journals to a GFS2 file system.</td>
</tr>
<tr>
<td>gfs2_grow</td>
<td>Grow a GFS2 file system.</td>
</tr>
<tr>
<td>mkfs.gfs2</td>
<td>Create a GFS2 file system on a device, usually a shared device or partition.</td>
</tr>
</tbody>
</table>
### GFS2 Utility

<table>
<thead>
<tr>
<th>GFS2 Utility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tunegfs2</td>
<td>Allows viewing and manipulating the GFS2 file system parameters such as UUID, label, lockproto and locktable.</td>
</tr>
</tbody>
</table>

### 20.2 Configuring GFS2 Services and a STONITH Resource

Before you can create GFS2 volumes, you must configure DLM and a STONITH resource.

**PROCEDURE 20.1: 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 *Procedure 11.3, “Initializing the SBD Devices”*.
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 \
   params pcmk_delay_max=30 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 `quit`.

For details on configuring the resource group for DLM, see *Procedure 18.1, “Configuring a Base Group for DLM”*. 
20.3 Creating GFS2 Volumes

After you have configured DLM as cluster resources as described in Section 20.2, “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 20.2, “Creating and Formatting a GFS2 Volume”. The most important parameters for the command are listed in Table 20.2, “Important GFS2 Parameters”. For more information and the command syntax, refer to the `mkfs.gfs2` man page.

<table>
<thead>
<tr>
<th>GFS2 Parameter</th>
<th>Description and Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock Protocol Name (-p)</td>
<td>The name of the locking protocol to use. Acceptable locking protocols are lock_dlm (for shared storage) or if you are using GFS2 as a local file system (1 node only), you can specify the lock_nolock protocol. If this option is not specified, lock_dlm protocol will be assumed.</td>
</tr>
<tr>
<td>Lock Table Name (-t)</td>
<td>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).</td>
</tr>
<tr>
<td>Number of Journals (-j)</td>
<td>The number of journals for gfs2_mkfs 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.</td>
</tr>
</tbody>
</table>
PROCEDURE 20.2: CREATING AND FORMATTING A 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:

   ```bash
   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).

PROCEDURE 20.4: MOUNTING A GFS2 VOLUME WITH THE CLUSTER MANAGER

To mount a 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 Hawk2 to configure the resources.

**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.
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" \ 
   op start timeout="60" op stop timeout="60" \ 
   meta target-role="Stopped"
   ```

4. Create a base group that consists of the dlm primitive you created in Procedure 18.1, “Configuring a Base Group for DLM” and the gfs2-1 primitive. Clone the group:

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

   Because of the base group's internal colocation and ordering, Pacemaker will only start the gfs2-1 resource on nodes that also have a 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 quit.
21 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.

21.1 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.

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 21.1, “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`. 
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 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 \texttt{/dev/drbdN} 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 symbolic links in the form \texttt{/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 KB hidden and reserved for the metadata. Any attempt to access the remaining kilobytes via raw disks fails because it is not available for user data.

### 21.2 Installing DRBD Services

Install the High Availability Extension on both SUSE Linux Enterprise Server machines in your networked cluster as described in \textit{Part I, “Installation, Setup and Upgrade”}. Installing High Availability Extension also installs the DRBD program files.

If you do not need the complete cluster stack but only want to use DRBD, install the packages \texttt{drbd}, \texttt{drbd-kmp-FLAVOR}, \texttt{drbd-utils}, and \texttt{yast2-drbd}.

To simplify the work with \texttt{drbdadm}, 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 \texttt{root}, create, or extend a file \texttt{/root/.bashrc} and insert the previous line.

### 21.3 Setting Up DRBD Service

\textbf{Note: Adjustments Needed}

The following procedure uses the server names alice and bob, and the cluster resource name \texttt{r0}. It sets up alice as the primary node and \texttt{/dev/sda1} for storage. Make sure to modify the instructions to use your own nodes and file names.
The following sections assumes you have two nodes, alice and bob, and that they should use the TCP port 7788. Make sure this port is open in your firewall.

1. Prepare your system:
   
   a. Make sure the block devices in your Linux nodes are ready and partitioned (if needed).
   
   b. If your disk already contains a file system that you do not need anymore, destroy the file system structure with the following command:

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

   If you have more file systems to destroy, repeat this step on all devices you want to include into your DRBD setup.

   c. If the cluster is already using DRBD, put your cluster in maintenance mode:

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

   As an alternative, you can also use `drbdadm -c FILE` to test a configuration file.

2. Configure DRBD by choosing your method:

   - Section 21.3.1, “Configuring DRBD Manually”
   - Section 21.3.2, “Configuring DRBD with YaST”

3. If you have configured Csync2 (which should be the default), the DRBD configuration files are already included in the list of files 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 bob:/etc/
   root # scp /etc/drbd.d/* bob:/etc/drbd.d/
   ```

4. Perform the initial synchronization (see Section 21.3.3, “Initializing and Formatting DRBD Resource”).
5. Reset the cluster’s maintenance mode flag:

   ```bash
   root #crm configure property maintenance-mode=false
   ```

### 21.3.1 Configuring DRBD Manually

**Note: Restricted Support of “Auto Promote” Feature**

The DRBD9 feature “auto-promote” can automatically promote a resource to the primary role when one of its devices is mounted or opened for writing.

The auto promote feature has currently restricted support. With DRBD 9, SUSE supports the same use cases that were also supported with DRBD 8. Use cases beyond that, such as setups with more than two nodes, are not supported.

To set up DRBD manually, proceed as follows:

**PROCEDURE 21.1: MANUALLY CONFIGURING DRBD**

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

1. Open the file `/etc/drbd.d/global_common.conf`. It contains already some global, predefined 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 https://docs.linbit.com/docs/users-guide-9.0/#ch-configure for more details.

2. Create the file `/etc/drbd.d/r0.res`. Change the lines according to your situation and save it:

   ```
   resource r0 {
     device /dev/drbd0;
     disk /dev/sda1;
     meta-disk internal;
     on alice {
   ```
address 192.168.1.10:7788; ①
node-id 0; ②
}
on bob { ③
  address 192.168.1.11:7788; ④
  node-id 1; ⑤
}disk { ⑥
  resync-rate 10M; ⑦
}connection-mesh { ⑧
  hosts alice bob;
}

① DRBD resource name that allows some association to the service that needs them. For example, nfs, http, mysql_0, postgres_wal, etc. Here a more general name r0 is used.
② 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 symbolic link /dev/drbd/by-res/nfs/0. Alternatively, omit the device node name in the configuration and use the following line instead: drbd0 minor 0 (/dev/ is optional) or /dev/drbd0
③ 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.
④ The meta-disk parameter usually contains the value internal, but it is possible to specify an explicit device to hold the meta data. See https://docs.linbit.com/docs/users-guide-9.0/#s-metadata for more information.
⑤ The on section states which host this configuration statement applies to.
⑥ The IP address and port number of the respective node. Each resource needs an individual port, usually starting with 7788. Both ports must be the same for a DRBD resource.
⑦ The node ID is required when configuring more than two nodes. It is a unique, non-negative integer to distinguish the different nodes.
⑧ 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.
Defines all nodes of a mesh. The `hosts` parameter contains all host names that share the same DRBD setup.

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

```
root # drbdadm dump all
```

4. Continue with Section 21.3.3, “Initializing and Formatting DRBD Resource”.

### 21.3.2 Configuring DRBD with YaST

YaST can be used to start with an initial setup of DRBD. After you have created your DRBD setup, you can fine-tune the generated files manually.

However, when you have changed the configuration files, do not use the YaST DRBD module anymore. The DRBD module supports only a limited set of basic configuration. If you use it again, it is very likely that the module will not show your changes.

To set up DRBD with YaST, proceed as follows:

**PROCEDURE 21.2: 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`.

2. Leave the booting flag in `Start-up Configuration > Booting` as it is (by default it is `off`); do not change that as Pacemaker manages this service.

3. If you have a firewall running, enable `Open Port in Firewall`.

4. Go to the `Resource Configuration` entry. Press `Add` to create a new resource (see Figure 21.2, “Resource Configuration”).
The following parameters need to be set:

**Resource Name**
- The name of the DRBD resource (mandatory)

**Name**
- The host name of the relevant node

**Address:Port**
- The IP address and port number (default 7788) for the respective node

**Device**
- The block device path that is used to access the replicated data. If the device contains a minor number, the associated block device is usually named /dev/drbdX, where X is the device minor number. If the device does not contain a minor number, make sure to add minor 0 after the device name.

**Disk**
- The raw device that is replicated between both nodes. If you use LVM, insert your LVM device name.

**Meta-disk**
- The Meta-disk is either set to the value internal or specifies an explicit device extended by an index to hold the meta data needed by DRBD.
A real device may also be used for multiple drbd resources. For example, if your Meta-Disk is `/dev/sda6[0]` for the first resource, you may use `/dev/sda6[1]` for the second resource. However, there must be at least 128 MB space for each resource available on this disk. The fixed metadata size limits the maximum data size that you can 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)`.

5. Click `Save`.

6. Click `Add` to enter the second DRBD resource and finish with `Save`.

7. Close the resource configuration with `Ok` and `Finish`.

8. If you use LVM with DRBD, it is necessary to change some options in the LVM configuration file (see the LVM Configuration entry). This change can be done by the YaST DRBD module automatically.

   The disk name of localhost for the DRBD resource and the default filter will be rejected in the LVM filter. Only `/dev/drbd` can be scanned for an LVM device.

   For example, if `/dev/sda1` is used as a DRBD disk, the device name will be inserted as the first entry in the LVM filter. To change the filter manually, click the Modify LVM Device Filter Automatically check box.

9. Save your changes with `Finish`.

10. Continue with Section 21.3.3, “Initializing and Formatting DRBD Resource”.

### 21.3.3 Initializing and Formatting DRBD Resource

After you have prepared your system and configured DRBD, initialize your disk for the first time:

1. On both nodes (alice and bob), initialize the meta data storage:

   ```
   root # drbdadm create-md r0
   root # drbdadm up r0
   ```
2. To shorten the initial resynchronization of your DRBD resource check the following:

- If the DRBD devices on all nodes have the same data (for example, by destroying the file system structure with the `dd` command as shown in Section 21.3, “Setting Up DRBD Service”), then skip the initial resynchronization with the following command (on both nodes):

  ```
  root # drbdadm new-current-uuid --clear-bitmap r0/0
  ```

  The state will be `Secondary/Secondary UpToDate/UpToDate`

- Otherwise, proceed with the next step.

3. On the primary node alice, start the resynchronization process:

  ```
  root # drbdadm primary --force r0
  ```

4. Check the status with:

  ```
  root # drbdadm status r0
  r0 role:Primary
disk:UpToDate
  bob role:Secondary
  peer-disk:UpToDate
  ```

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

  ```
  root # mkfs.ext3 /dev/drbd0
  ```

6. Mount the file system and use it:

  ```
  root # mount /dev/drbd0 /mnt/
  ```

### 21.4 Migrating from DRBD 8 to DRBD 9

Between DRBD 8 (shipped with SUSE Linux Enterprise High Availability Extension 12 SP1) and DRBD 9 (shipped with SUSE Linux Enterprise High Availability Extension 12 SP2), the metadata format has changed. DRBD 9 does not automatically convert previous metadata files to the new format.

After migrating to 12 SP2 and before starting DRBD, convert the DRBD metadata to the version 9 format manually. To do so, use `drbdadm create-md`. No configuration needs to be changed.
Note: Restricted Support

With DRBD 9, SUSE supports the same use cases that were also supported with DRBD 8. Use cases beyond that, such as setups with more than two nodes, are not supported.

DRBD 9 will fall back to be compatible with version 8. For three nodes and more, you need to re-create the metadata to use DRBD version 9 specific options.

If you have a stacked DRBD resource, refer also to Section 21.5, “Creating a Stacked DRBD Device” for more information.

To keep your data and allow to add new nodes without re-creating new resources, do the following:

1. Set one node in standby mode.
2. Update all the DRBD packages on all of your nodes, see Section 21.2, “Installing DRBD Services”.
3. Add the new node information to your resource configuration:
   - `node-id` on every `on` section.
   - `connection-mesh` section contains all host names in the `hosts` parameter.

   See the example configuration in Procedure 21.1, “Manually Configuring DRBD”.
4. Enlarge the space of your DRBD disks when using `internal` as `meta-disk` key. Use a device that supports enlarging the space like LVM. As an alternative, change to an external disk for metadata and use `meta-disk DEVICE;`.
5. Re-create the metadata based on the new configuration:

   ```
   root # drbdadm create-md RESOURCE
   ```
6. Cancel the standby mode.

21.5 Creating a Stacked DRBD Device

A stacked DRBD device contains two other devices of which at least one device is also a DRBD resource. In other words, DRBD adds an additional node on top of an already existing DRBD resource (see Figure 21.3, “Resource Stacking”). Such a replication setup can be used for backup and disaster recovery purposes.
Three-way replication uses asynchronous (DRBD protocol A) and synchronous replication (DRBD protocol C). The asynchronous part is used for the stacked resource whereas the synchronous part is used for the backup.

Your production environment uses the stacked device. For example, if you have a DRBD device `/dev/drbd0` and a stacked device `/dev/drbd10` on top, the file system will be created on `/dev/drbd10`, see Example 21.1, “Configuration of a Three-Node Stacked DRBD Resource” for more details.

**EXAMPLE 21.1: CONFIGURATION OF A THREE-NODE STACKED DRBD RESOURCE**

```bash
# /etc/drbd.d/r0.res
resource r0 {
    protocol C;
    device /dev/drbd0;
    disk /dev/sda1;
    meta-disk internal;

    on amsterdam-alice {
        address 192.168.1.1:7900;
    }

    on amsterdam-bob {
        address 192.168.1.2:7900;
    }
}

resource r0-U {
    protocol A;
```
21.6 Using Resource-Level Fencing with STONITH

When a DRBD replication link becomes interrupted, Pacemaker tries to promote the DRBD resource to another node. To prevent Pacemaker from starting a service with outdated data, enable resource-level fencing in the DRBD configuration file.

The fencing policy can have different values (see man page `drbdsetup` and the `--fencing` option). As a SUSE Linux Enterprise High Availability Extension cluster is normally used with a STONITH device, the value `resource-and-stonith` is used in Example 21.2, “Configuration of DRBD with Resource-Level Fencing Using the Cluster Information Base (CIB)”.

**EXAMPLE 21.2: CONFIGURATION OF DRBD WITH RESOURCE-LEVEL FENCING USING THE CLUSTER INFORMATION BASE (CIB)**

```plaintext
resource RESOURCE {
  net {
    fencing resource-and-stonith;
    # ...
  }
  handlers {
    fence-peer "usr/lib/drbd/crm-fence-peer.9.sh";
    after-resync-target "usr/lib/drbd/crm-unfence-peer.9.sh";
    # ...
  }
  ...
}
```

If the DRBD replication link becomes disconnected, DRBD does the following:

1. DRBD calls the `crm-fence-peer.9.sh` script.
2. The script contacts the cluster manager.
3. The script determines the Pacemaker resource associated with this DRBD resource.

4. The script ensures that the DRBD resource no longer gets promoted to any other node. It stays on the currently active one.

5. If the replication link becomes connected again and DRBD completes its synchronization process, then the constraint is removed. The cluster manager is now free to promote the resource.

21.7 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 alice.
   a. Open a terminal console, then log in as root.
   b. Create a mount point on alice, 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_alice
   ```

e. Unmount the disk on alice:

   ```
   root # umount /srv/r0
   ```

f. Downgrade the DRBD service on alice by typing the following command on alice:

   ```
   root # drbdadm secondary r0
   ```

2. Test the DRBD service on bob.

   a. Open a terminal console, then log in as root on bob.
b. On bob, promote the DRBD service to primary:

```
root # drbdadm primary r0
```

c. On bob, check to see if bob is primary:

```
root # drbdadm status r0
```

d. On bob, create a mount point such as `/srv/r0`:

```
root # mkdir /srv/r0
```

e. On bob, mount the DRBD device:

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

f. Verify that the file you created on alice exists:

```
root # ls /srv/r0/from_alice
```

The `/srv/r0/from_alice` file should be listed.

3. If the service is working on both nodes, the DRBD setup is complete.

4. Set up alice as the primary again.

   a. Dismount the disk on bob by typing the following command on bob:

```
root # umount /srv/r0
```

   b. Downgrade the DRBD service on bob by typing the following command on bob:

```
root # drbdadm secondary r0
```

   c. On alice, promote the DRBD service to primary:

```
root # drbdadm primary r0
```

   d. On alice, check to see if alice is primary:

```
root # drbdadm status r0
```
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 15 SP1 see Part II, “Configuration and Administration”.

21.8 Monitoring DRBD Devices

DRBD comes with the utility `drbdmon` which offers realtime monitoring. It shows all the configured resources and their problems.

![FIGURE 21.4: SHOWING A GOOD CONNECTION BY drbdmon](image)

In case of problems, `drbdadm` shows an error message:

![FIGURE 21.5: SHOWING A BAD CONNECTION BY drbdmon](image)

21.9 Tuning DRBD

There are several ways to tune DRBD:

1. Use an external disk for your metadata. This might help, at the cost of maintenance ease.

2. Tune your network connection, by changing the receive and send buffer settings via `sysctl`.
3. Change the `max-buffers`, `max-epoch-size` or both in the DRBD configuration.

4. Increase the `al-extents` value, depending on your IO patterns.

5. 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`.


### 21.10 Troubleshooting DRBD

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

#### 21.10.1 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.

21.10.2 Host Names

For DRBD, host names are case-sensitive (Node0 would be a different host than node0), and compared to the host name 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 host name will likely not resolve to the used IP address. In this case, use the parameter `disable-ip-verification`.

21.10.3 TCP Port 7788

If your system cannot 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.

21.10.4 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 commands on the node which has data to be discarded:

```
root # drbdadm secondary r0
```

If the state is in `WFconnection`, disconnect first:

```
root # drbdadm disconnect r0
```

On the node which has the latest data enter the following:

```
root # drbdadm connect --discard-my-data r0
```

That resolves the issue by overwriting one node's data with the peer's data, therefore getting a consistent view on both nodes.
21.11 For More Information

The following open source resources are available for DRBD:


- See Article “Highly Available NFS Storage with DRBD and Pacemaker”.

- [http://clusterlabs.org/wiki/DRBD_HowTo_1.0](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-meta(8), drbdsetup(8), drbdadm(8), drbd.conf(5)`.

- Find a commented example configuration for DRBD at `/usr/share/doc/packages/drbd-utils/drbd.conf.example`.

- Furthermore, for easier storage administration across your cluster, see the recent announcement about the **DRBD-Manager** at [https://www.linbit.com/en/drbd-manager/](https://www.linbit.com/en/drbd-manager/).
22  Cluster Logical Volume Manager (Cluster LVM)

When managing shared storage on a cluster, every node must be informed about changes that are done to the storage subsystem. The Logical 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. Volume groups shared among multiple hosts can be managed using the same commands as local storage.

22.1  Conceptual Overview

Cluster LVM is coordinated with different tools:

**Distributed Lock Manager (DLM)**

Coordinates access to shared resources among multiple hosts through cluster-wide locking.

**Logical Volume Manager (LVM2)**

LVM2 provides a virtual pool of disk space and enables flexible distribution of one logical volume over several disks.

**Cluster Logical Volume Manager (Cluster LVM)**

The term **Cluster LVM** indicates that LVM2 is being used in a cluster environment. This needs some configuration adjustments to protect the LVM2 metadata on shared storage. From SUSE Linux Enterprise 15 onward, the cluster extension uses lvmlockd, which replaces the well-known clvmd. For more information about lvmlockd, see the man page of the **lvmlockd** command (**man 8 lvmlockd**).

**Volume Group and Logical Volume**

Volume groups (VGs) and logical volumes (LVs) are basic concepts of LVM2. A volume group is a storage pool of multiple physical disks. A logical volume belongs to a volume group, and can be seen as an elastic volume on which you can create a file system. In a cluster environment, there is a concept of shared VGs, which consist of shared storage and can be used concurrently by multiple hosts.
22.2  Configuration of Cluster LVM

Make sure the following requirements are fulfilled:

- A shared storage device is available, provided by a Fibre Channel, FCoE, SCSI, iSCSI SAN, or DRBD*, for example.
- Make sure the following packages have been installed: `lvm2` and `lvm2-lockd`.
- From SUSE Linux Enterprise 15 onward, we use lvmlockd as the LVM2 cluster extension, rather than clvmd. Make sure the clvmd daemon is not running, otherwise lvmlockd will fail to start.

22.2.1  Creating the Cluster Resources

Perform the following basic steps on one node to configure a shared VG in the cluster:

- Creating a DLM Resource
- Creating an lvmlockd Resource
- Creating a Shared VG and LV
- Creating an LVM-activate Resource

PROCEDURE 22.1: CREATING A DLM RESOURCE

1. Start a shell and log in as `root`.
2. Check the current configuration of the cluster resources:

```
root # crm configure show
```

3. If you have already configured a DLM resource (and a corresponding base group and base clone), continue with Procedure 22.2, “Creating an lvmlockd Resource”.

Otherwise, configure a DLM resource and a corresponding base group and base clone as described in Procedure 18.1, “Configuring a Base Group for DLM”.

PROCEDURE 22.2: CREATING AN LVMLOCKD RESOURCE

1. Start a shell and log in as `root`.
2. Run the following command to see the usage of this resource:

```
root # crm configure ra info lvmlockd
```
3. Configure a lvmlockd resource as follows:

   root # crm configure primitive lvmlockd lvmlockd \
   op start timeout="90" \
   op stop timeout="100" \
   op monitor interval="30" timeout="90"

4. To ensure the lvmlockd resource is started on every node, add the primitive resource to the base group for storage you have created in Procedure 22.1, “Creating a DLM Resource”:

   root # crm configure modgroup g-storage add lvmlockd

5. Review your changes:

   root # crm configure show

6. Check if the resources are running well:

   root # crm status full

PROCEDURE 22.3: CREATING A SHARED VG AND LV

1. Start a shell and log in as root.

2. Assuming you already have two shared disks, create a shared VG with them:

   root # vgcreate --shared vg1 /dev/sda /dev/sdb

3. Create an LV and do not activate it initially:

   root # lvcreate -an -L10G -n lv1 vg1

PROCEDURE 22.4: CREATING AN LVM-ACTIVATE RESOURCE

1. Start a shell and log in as root.

2. Run the following command to see the usage of this resource:

   root # crm configure ra info LVM-activate

   This resource manages the activation of a VG. In a shared VG, LV activation has two different modes: exclusive and shared mode. The exclusive mode is the default and should be used normally, when a local file system like ext4 uses the LV. The shared mode should only be used for cluster file systems like OCFS2.
3. Configure a resource to manage the activation of your VG. Choose one of the following options according to your scenario:

- Use exclusive activation mode for local file system usage:

```bash
root # crm configure primitive vg1 LVM-activate \
    params vgname=vg1 vg_access_mode=lvmlockd \
    op start timeout=90s interval=0 \
    op stop timeout=90s interval=0 \
    op monitor interval=30s timeout=90s
```

- Use shared activation mode for OCFS2 and add it to the cloned `g-storage` group:

```bash
root # crm configure primitive vg1 LVM-activate \
    params vgname=vg1 vg_access_mode=lvmlockd activation_mode=shared \
    op start timeout=90s interval=0 \
    op stop timeout=90s interval=0 \
    op monitor interval=30s timeout=90s

root # crm configure modgroup g-storage add vg1
```

4. Check if the resources are running well:

```bash
root # crm status full
```

### 22.2.2 Scenario: Cluster LVM 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 22.1, “Setup of a Shared Disk with Cluster LVM”.

```bash
```
Warning: Data Loss
The following procedures will destroy any data on your disks!

Configure only one SAN box first. Each SAN box needs to export its own iSCSI target. Proceed as follows:

PROCEDURE 22.5: CONFIGURING ISCSI TARGETS (SAN)

1. Run YaST and click Network Services > iSCSI LIO 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 needs to be formatted like this:
      
      \[iqn.DATE.DOMAIN\]
      
      For more information about the format, refer to Section 3.2.6.3.1. Type "irqn." (iSCSI Qualified Name) at 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 \(/etc/iscsi/iscsid.conf\) and change the parameter node.startup to automatic.

Now set up your iSCSI initiators as follows:

PROCEDURE 22.6: 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 the IP address and the port of your iSCSI target (see Procedure 22.5, “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 user name 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 -p 192.168.3.100 -T iqn.2010-03.de.jupiter:san1
   Logging in to [iface: default, target: iqn.2010-03.de.jupiter:san1, portal: 192.168.3.100,3260]
   Login to [iface: default, target: iqn.2010-03.de.jupiter:san1, portal: 192.168.3.100,3260]: successful
   ```

   See the device names with `lsscsi`:

   ```
   ...
   [4:0:0:2] disk IET ... 0 /dev/sdd
   [5:0:0:1] disk IET ... 0 /dev/sde
   ```

   Look for entries with IET in their third column. In this case, the devices are `/dev/sdd` and `/dev/sde`.

PROCEDURE 22.7: CREATING THE SHARED VOLUME GROUPS

1. Open a root shell on one of the nodes you have run the iSCSI initiator from Procedure 22.6, “Configuring iSCSI Initiators”.

2. Create the shared volume group on disks /dev/sdd and /dev/sde:

   ```
   root # vgcreate --shared testvg /dev/sdd /dev/sde
   ```

3. Create logical volumes as needed:

   ```
   root # lvcreate --name lv1 --size 500M testvg
   ```

4. Check the volume group with `vgdisplay`:

   ```
   --- Volume group ---
   VG Name       testvg
   System ID
   Format        lvml2
   ```
5. Check the shared state of the volume group with the command `vgs`:

```
root # vgs
VG   #PV #LV #SN Attr  VSize  VFree
vgshared 1 1 0 wz--ns 1016.00m 1016.00m
```

The `Attr` column shows the volume attributes. In this example, the volume group is writable (w), resizeable (z), the allocation policy is normal (n), and it is a shared resource (s). See the man page of `vgs` for details.

After you have created the volumes and started your resources you should have new device names under `/dev/testvg`, for example `/dev/testvg/lv1`. This indicates the LV has been activated for use.

### 22.2.3 Scenario: Cluster LVM with DRBD

The following scenarios can be used if you have data centers located in different parts of your city, country, or continent.

**PROCEDURE 22.8: 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 21.1, “Manually Configuring DRBD”*. Make sure the disk state is up-to-date on both nodes. Check this with `drbdadm status`. 

308  Scenario: Cluster LVM with DRBD SLE HA 15 SP1
b. Add the following options to your configuration file (usually something like \texttt{/etc/drbd.d/r0.res}):

```bash
resource r0 {
    net {
        allow-two-primaries;
    }
    ...
}
```

c. Copy the changed configuration file to the other node, for example:

```bash
root # scp /etc/drbd.d/r0.res venus:/etc/drbd.d/
```

d. Run the following commands on both nodes:

```bash
root # drbdadm disconnect r0
root # drbdadm connect r0
root # drbdadm primary r0
```

e. Check the status of your nodes:

```bash
root # drbdadm status r0
```

2. Include the lvmlockd resource as a clone in the pacemaker configuration, and make it depend on the DLM clone resource. See Procedure 22.1, “Creating a DLM Resource” for detailed instructions. Before proceeding, confirm that these resources have started successfully on your cluster. Use \texttt{crm status} or the Web interface to check the running services.

3. Prepare the physical volume for LVM with the command \texttt{pvcreate}. For example, on the device \texttt{/dev/drbd_r0} the command would look like this:

```bash
root # pvcreate /dev/drbd_r0
```

4. Create a shared volume group:

```bash
root # vgcreate --shared testvg /dev/drbd_r0
```

5. Create logical volumes as needed. You probably want to change the size of the logical volume. For example, create a 4 GB logical volume with the following command:

```bash
root # lvcreate --name lv1 -L 4G testvg
```
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 like on any stand-alone workstation.

### 22.3 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), we recommend 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. This 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.
22.4 Online Migration from Mirror LV to Cluster MD

Starting with SUSE Linux Enterprise High Availability Extension 15, cmirrod in cluster LVM is deprecated. We highly recommend to migrate the mirror logical volumes in your cluster to cluster MD. Cluster MD stands for cluster multi-device and is a software-based RAID storage solution for a cluster.

22.4.1 Example Setup Before Migration

Let us assume you have the following example setup:

- You have a two-node cluster consisting of the nodes alice and bob.
- A mirror logical volume named test-lv was created from a volume group named cluster-vg2.
- The volume group cluster-vg2 is composed of the disks /dev/vdb and /dev/vdc.

```bash
root # lsblk
NAME                                 MAJ:MIN  RM   SIZE RO TYPE MOUNTPOINT
vda                                   253:0   0    40G  0 disk
├─vda1                                253:1   0    4G   0 part [SWAP]
└─vda2                                253:2   0    36G  0 part /
vdb                                   253:16  0    20G  0 disk
├─cluster--vg2-test--lv_mlog_mimage_0 254:0   0    4M  0 lvm
│ └─cluster--vg2-test--lv_mlog         254:2   0    4M  0 lvm
│   └─cluster--vg2-test--lv            254:5   0    12G 0 lvm
└─cluster--vg2-test--lv_mimage_0      254:3   0    12G 0 lvm
└─cluster--vg2-test--lv               254:5   0    12G 0 lvm
vdc                                   253:32  0    20G  0 disk
├─cluster--vg2-test--lv_mlog_mimage_1 254:1   0    4M  0 lvm
│ └─cluster--vg2-test--lv_mlog         254:2   0    4M  0 lvm
│   └─cluster--vg2-test--lv            254:5   0    12G 0 lvm
└─cluster--vg2-test--lv_mimage_1      254:4   0    12G 0 lvm
└─cluster--vg2-test--lv               254:5   0    12G 0 lvm
```
Important: Avoiding Migration Failures

Before you start the migration procedure, check the capacity and degree of utilization of your logical and physical volumes. If the logical volume uses 100% of the physical volume capacity, the migration might fail with an insufficient free space error on the target volume. How to prevent this migration failure depends on the options used for mirror log:

- Is the mirror log itself mirrored (mirrored option) and allocated on the same device as the mirror log? (For example, this might be the case if you have created the logical volume for a cmirrord setup on SUSE Linux Enterprise High Availability Extension 11 or 12 as described in the Administration Guide for those versions (https://documentation.suse.com/sle-ha/12-SP5/html/SLE-HA-all/cha-ha-clvm.html#sec-ha-clvm-config-cmirrord).)

By default, mdadm reserves a certain amount of space between the start of a device and the start of array data. During migration, you can check for the unused padding space and reduce it with the data-offset option as shown in Step 1.d and following.

The data-offset must leave enough space on the device for cluster MD to write its metadata to it. On the other hand, the offset must be small enough for the remaining capacity of the device to accommodate all physical volume extents of the migrated volume. Because the volume may have spanned the complete device minus the mirror log, the offset must be smaller than the size of the mirror log.

We recommend to set the data-offset to 128 kB. If no value is specified for the offset, its default value is 1 kB (1024 bytes).

- Is the mirror log written to a different device (disk option) or kept in memory (core option)? Before starting the migration, either enlarge the size of the physical volume or reduce the size of the logical volume (to free more space for the physical volume).

22.4.2 Migrating a Mirror LV to Cluster MD

The following procedure is based on Section 22.4.1, “Example Setup Before Migration”. Adjust the instructions to match your setup and replace the names for the LVs, VGs, disks and the cluster MD device accordingly.
The migration does not involve any downtime. The file system can still be mounted during the migration procedure.

1. On node alice, execute the following steps:

   a. Convert the mirror logical volume test-lv to a linear logical volume:

   ```
   root # lvconvert -m0 cluster-vg2/test-lv /dev/vdc
   ```

   b. Remove the physical volume /dev/vdc from the volume group cluster-vg2:

   ```
   root # vgreduce cluster-vg2 /dev/vdc
   ```

   c. Remove this physical volume from LVM:

   ```
   root # pvremove /dev/vdc
   ```

   When you run `lsblk` now, you get:

<table>
<thead>
<tr>
<th>NAME</th>
<th>MAJ:MIN</th>
<th>RM</th>
<th>SIZE</th>
<th>RO</th>
<th>TYPE</th>
<th>MOUNTPOINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>vda</td>
<td>253:0</td>
<td>0</td>
<td>40G</td>
<td>0</td>
<td>disk</td>
<td></td>
</tr>
<tr>
<td>└─vda1</td>
<td>253:1</td>
<td>0</td>
<td>4G</td>
<td>0</td>
<td>part</td>
<td>[SWAP]</td>
</tr>
<tr>
<td>└─vda2</td>
<td>253:2</td>
<td>0</td>
<td>36G</td>
<td>0</td>
<td>part</td>
<td>/</td>
</tr>
<tr>
<td>vdb</td>
<td>253:16</td>
<td>0</td>
<td>20G</td>
<td>0</td>
<td>disk</td>
<td></td>
</tr>
<tr>
<td>└─cluster--vg2-test--lv</td>
<td>254:5</td>
<td>0</td>
<td>12G</td>
<td>0</td>
<td>lvm</td>
<td></td>
</tr>
<tr>
<td>vdc</td>
<td>253:32</td>
<td>0</td>
<td>20G</td>
<td>0</td>
<td>disk</td>
<td></td>
</tr>
</tbody>
</table>

   d. Create a cluster MD device /dev/md0 with the disk /dev/vdc:

   ```
   root # mdadm --create /dev/md0 --bitmap=clustered \
            --metadata=1.2 --raid-devices=1 --force --level=mirror \
            /dev/vdc --data-offset=128
   ```

   For details on why to use the `data-offset` option, see Important: Avoiding Migration Failures.

2. On node bob, assemble this MD device:

   ```
   root # mdadm --assemble md0 /dev/vdc
   ```

   If your cluster consists of more than two nodes, execute this step on all remaining nodes in your cluster.
3. Back on node alice:

a. Initialize the MD device /dev/md0 as physical volume for use with LVM:

```
root # pvcreate /dev/md0
```

b. Add the MD device /dev/md0 to the volume group cluster-vg2:

```
root # vgextend cluster-vg2 /dev/md0
```

c. Move the data from the disk /dev/vdb to the /dev/md0 device:

```
root # pvmove /dev/vdb /dev/md0
```

d. Remove the physical volume /dev/vdb from the volume group cluster-vg2:

```
root # vgreduce cluster-vg2 /dev/vdb
```

e. Remove the label from the device so that LVM no longer recognizes it as physical volume:

```
root # pvremove /dev/vdb
```

f. Add /dev/vdb to the MD device /dev/md0:

```
root # mdadm --grow /dev/md0 --raid-devices=2 --add /dev/vdb
```

22.4.3 Example Setup After Migration

When you run `lsblk` now, you get:

<table>
<thead>
<tr>
<th>NAME</th>
<th>MAJ:MIN</th>
<th>RM</th>
<th>SIZE</th>
<th>RO</th>
<th>TYPE</th>
<th>MOUNTPOINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>vda</td>
<td>253:0</td>
<td>0</td>
<td>40G</td>
<td>0</td>
<td>disk</td>
<td></td>
</tr>
<tr>
<td>├─vda1</td>
<td>253:1</td>
<td>0</td>
<td>4G</td>
<td>0</td>
<td>part</td>
<td>[SWAP]</td>
</tr>
<tr>
<td>│─vda2</td>
<td>253:2</td>
<td>0</td>
<td>36G</td>
<td>0</td>
<td>part</td>
<td>/</td>
</tr>
<tr>
<td>vdb</td>
<td>253:16</td>
<td>0</td>
<td>20G</td>
<td>0</td>
<td>disk</td>
<td></td>
</tr>
<tr>
<td>├─md0</td>
<td>9:0</td>
<td>0</td>
<td>20G</td>
<td>0</td>
<td>raid1</td>
<td></td>
</tr>
<tr>
<td>│─cluster-vg2-test-1v 254:5</td>
<td>0</td>
<td>12G</td>
<td>0</td>
<td>lvm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vdc</td>
<td>253:32</td>
<td>0</td>
<td>20G</td>
<td>0</td>
<td>disk</td>
<td></td>
</tr>
<tr>
<td>├─md0</td>
<td>9:0</td>
<td>0</td>
<td>20G</td>
<td>0</td>
<td>raid1</td>
<td></td>
</tr>
<tr>
<td>│─cluster-vg2-test-1v 254:5</td>
<td>0</td>
<td>12G</td>
<td>0</td>
<td>lvm</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
22.5 For More Information

For more information about lvmlockd, see the man page of the `lvmlockd` command (\texttt{man 8 lvmlockd}).

Thorough information is available from the pacemaker mailing list, available at \url{http://www.clusterlabs.org/wiki/Help:Contents}.
23  Cluster Multi-device (Cluster MD)

The cluster multi-device (Cluster MD) is a software based RAID storage solution for a cluster. Currently, Cluster MD provides the redundancy of RAID1 mirroring to the cluster. With SUSE Linux Enterprise High Availability Extension 15 SP1, RAID10 is included as a technology preview. If you want to try RAID10, replace `mirror` with `10` in the related `mdadm` command. This chapter shows you how to create and use Cluster MD.

23.1  Conceptual Overview

The Cluster MD provides support for use of RAID1 across a cluster environment. The disks or devices used by Cluster MD are accessed by each node. If one device of the Cluster MD fails, it can be replaced at runtime by another device and it is re-synced to provide the same amount of redundancy. The Cluster MD requires Corosync and Distributed Lock Manager (DLM) for coordination and messaging.

A Cluster MD device is not automatically started on boot like the rest of the regular MD devices. A clustered device needs to be started using resource agents to ensure the DLM resource has been started.

23.2  Creating a Clustered MD RAID Device

**REQUIREMENTS**

- A running cluster with pacemaker.
- A resource agent for DLM (see Procedure 18.1, “Configuring a Base Group for DLM” on how to configure DLM).
- At least two shared disk devices. You can use an additional device as a spare which will fail over automatically in case of device failure.
- An installed package `cluster-md-kmp-default`. 
1. Make sure the DLM resource is up and running on every node of the cluster and check the resource status with the command:

```
root # crm_resource -r dlm -W
```

2. Create the Cluster MD device:

- If you do not have an existing normal RAID device, create the Cluster MD device on the node running the DLM resource with the following command:

```
root # mdadm --create /dev/md0 --bitmap=clustered \
--metadata=1.2 --raid-devices=2 --level=mirror /dev/sda /dev/sdb
```

As Cluster MD only works with version 1.2 of the metadata, it is recommended to specify the version using the `--metadata` option. For other useful options, refer to the man page of `mdadm`. Monitor the progress of the re-sync in `/proc/mdstat`.

- If you already have an existing normal RAID, first clear the existing bitmap and then create the clustered bitmap:

```
root # mdadm --grow /dev/mdX --bitmap=none
root # mdadm --grow /dev/mdX --bitmap=clustered
```

- Optionally, to create a Cluster MD device with a spare device for automatic failover, run the following command on one cluster node:

```
root # mdadm --create /dev/md0 --bitmap=clustered --raid-devices=2 \
--level=mirror --spare-devices=1 /dev/sda /dev/sdb /dev/sdc --metadata=1.2
```

3. Get the UUID and the related md path:

```
root # mdadm --detail --scan
```

The UUID must match the UUID stored in the superblock. For details on the UUID, refer to the `mdadm.conf` man page.

4. Open `/etc/mdadm.conf` and add the md device name and the devices associated with it. Use the UUID from the previous step:

```
DEVICE /dev/sda /dev/sdb
ARRAY /dev/md0 UUID=1d70f103:49740ef1:af2afce5:fcf6a489
```
5. Open Csync2’s configuration file `/etc/csync2/csync2.cfg` and add `/etc/mdadm.conf`:

```
{ 
    # ... list of files pruned ... 
    include /etc/mdadm.conf 
}
```

### 23.3 Configuring a Resource Agent

Configure a CRM resource as follows:

1. Create a **Raid1** primitive:

   ```
   primitive raider Raid1 
   "params raidconf="/etc/mdadm.conf" raiddev=/dev/md0 " 
   "force_clones=true" 
   "op monitor timeout=20s interval=10" 
   "op start timeout=20s interval=0" 
   "op stop timeout=20s interval=0"
   ```

2. Add the **raider** resource to the base group for storage that you have created for DLM:

   ```
   modgroup g-storage add raider
   ```

   The **add** sub-command appends the new group member by default.
   If not already done, clone the `g-storage` group so that it runs on all nodes:

   ```
   clone cl-storage g-storage 
   "meta interleave=true target-role=Started"
   ```

3. Review your changes with **show**.

4. If everything seems correct, submit your changes with **commit**.

### 23.4 Adding a Device

To add a device to an existing, active Cluster MD device, first ensure that the device is “visible” on each node with the command `cat /proc/mdstat`. If the device is not visible, the command will fail.
Use the following command on one cluster node:

```bash
root # mdadm --manage /dev/md0 --add /dev/sdc
```

The behavior of the new device added depends on the state of the Cluster MD device:

- If only one of the mirrored devices is active, the new device becomes the second device of the mirrored devices and a recovery is initiated.
- If both devices of the Cluster MD device are active, the new added device becomes a spare device.

### 23.5 Re-adding a Temporarily Failed Device

Quite often the failures are transient and limited to a single node. If any of the nodes encounters a failure during an I/O operation, the device will be marked as failed for the entire cluster. This could happen, for example, because of a cable failure on one of the nodes. After correcting the problem, you can re-add the device. Only the outdated parts will be synchronized as opposed to synchronizing the entire device by adding a new one.

To re-add the device, run the following command on one cluster node:

```bash
root # mdadm --manage /dev/md0 --re-add /dev/sdb
```

### 23.6 Removing a Device

Before removing a device at runtime for replacement, do the following:

1. Make sure the device is failed by introspecting `/proc/mdstat`. Look for an (F) before the device.

2. Run the following command on one cluster node to make a device fail:

   ```bash
   root # mdadm --manage /dev/md0 --fail /dev/sda
   ```

3. Remove the failed device using the command on one cluster node:

   ```bash
   root # mdadm --manage /dev/md0 --remove /dev/sda
   ```
23.7 Assembling Cluster MD as normal RAID at the disaster recovery site

In the event of disaster recovery, you might face the situation that you do not have a Pacemaker cluster stack in the infrastructure on the disaster recovery site, but applications still need to access the data on the existing Cluster MD disks, or from the backups.

You can convert a Cluster MD RAID to a normal RAID by using the \texttt{--assemble} operation with the \texttt{-U no-bitmap} option to change the metadata of the RAID disks accordingly.

Find an example below of how to assemble all arrays on the data recovery site:

```bash
while read i; do
    NAME=`echo $i | sed 's/.*name=//'|awk '{print $1}'|sed 's/.*://'
    UUID=`echo $i | sed 's/.*UUID=//'|awk '{print $1}'`
    mdadm -AR "/dev/md/$NAME" -u $UUID -U no-bitmap
    echo "NAME =" $NAME ", UUID =" $UUID ", assembled."
done < <(mdadm -Es)
```
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.

### 24.1 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 24.1, “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 x86 with AMD64.
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 needs 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.

### 24.2 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 24.1: SETTING UP A BASIC CLUSTERED SAMBA SERVER

1. Prepare your cluster:
   a. Make sure the following packages are installed before you proceed: ctdb, tdb-tools, 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 /srv/clusterfs. See Chapter 19, OCFS2 for more information.
   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 # tunefs.ocfs2 --fs-feature=xattr DEVICE
        ```
        Make sure the acl option is specified in the file system resource. Use the crm shell as follows:
        
        ```
        crm(live)configure# primitive ocfs2-3 ocf:heartbeat:Filesystem params options="acl" ...
        ```
   e. Make sure the services ctdb, smb, and nmb are disabled:

        ```
        root # systemctl disable ctdb
        root # systemctl disable smb
        root # systemctl disable nmb
        ```
   f. Open 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 /srv/clusterfs/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 host name of your choice in place of “CTDB-SERVER” (all nodes in the cluster will appear as one big node with this name, effectively):

```
[globals]
  # ...
  # settings applicable for all CTDB deployments
  netbios name = CTDB-SERVER
  clustering = yes
  idmap config * : backend = tdb2
  passdb backend = tdbsam
  ctdb_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 4.6, “Synchronizing the Configuration Files with Csync2”.

6. Add a CTDB resource to the cluster:

```
root # crm configure
crm(live)configure# primitive ctdb CTDB params \
  ctdb_manages_winbind="false" \n  ctdb_manages_samba="false" \n  ctdb_recovery_lock="/srv/clusterfs/samba/ctdb.lock" \n  ctdb_socket="/var/lib/ctdb/ctdb.socket" \n  op monitor interval="10" timeout="20" \n  op start interval="0" timeout="90" \n  op stop interval="0" timeout="100"

crm(live)configure# primitive nmb systemd:nmb \
  op start timeout="60" interval="0" \n  op stop timeout="60" interval="0" \n  op monitor timeout="60" interval="60"

crm(live)configure# primitive smb systemd:smb \
  op start timeout="60" interval="0" \
```
7. Add a clustered IP address:

```bash
crm(live)configure# primitive ip IPaddr2 params ip=192.168.2.222 \
    unique_clone_address="true" \ 
    op monitor interval="60" \ 
    meta resource-stickiness="0"
```

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 15.2, “Configuring Load Balancing with Linux Virtual Server”.

8. Commit your change:

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

9. Check the result:

```bash
root #crm status
Clone Set: cl-storage [dlm]
    Started: [ factory-1 ]
    Stopped: [ factory-0 ]
Clone Set: cl-clusterfs [clusterfs]
    Started: [ factory-1 ]
    Stopped: [ factory-0 ]
Clone Set: cl-ctdb [g-ctdb]
    Started: [ factory-1 ]
Clone Set: cl-ip [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
```

### 24.3 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 24.1, “Setting Up a Basic Clustered Samba Server”.

2. Install the `samba-winbind` package.

3. Disable the `winbind` service:

   ```
   root # systemctl disable winbind
   ```

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 `g-ctdb` group and insert `winbind` between the `nmb` and `smb` resources:

   ```
   crm(live)configure# edit g-ctdb
   ```

   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:

<table>
<thead>
<tr>
<th>AD and DNS server</th>
<th>win2k3.2k3test.example.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD domain</td>
<td>2k3test.example.com</td>
</tr>
</tbody>
</table>
Cluster AD member NetBIOS name

CTDB-SERVER


Finally, join your cluster to the Active Directory server:

PROCEDURE 24.2: 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 4.5, “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.

24.4 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
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](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.


To test certain aspects of your cluster file system proceed as follows:

**PROCEDURE 24.3: 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):

   ```bash
   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 19, OCFS2*.

### 24.5 For More Information

- [http://ctdb.samba.org](http://ctdb.samba.org)
25 Disaster Recovery with Rear (Relax-and-Recover)

Relax-and-Recover (“Rear”, in this chapter abbreviated as Rear) is a disaster recovery framework for use by system administrators. It is a collection of Bash scripts that need to be adjusted to the specific production environment that is to be protected in case of disaster.

No disaster recovery solution will work out-of-the-box. Therefore it is essential to take preparations before any disaster happens.

25.1 Conceptual Overview

The following sections describe the general disaster recovery concept and the basic steps you need to execute for successful recovery with Rear. They also provide some guidance on Rear requirements, some limitations to be aware of, and scenarios and backup tools.

25.1.1 Creating a Disaster Recovery Plan

Before the worst scenario happens, take action: analyze your IT infrastructure for any substantial risks, evaluate your budget, and create a disaster recovery plan. If you do not already have a disaster recovery plan at hand, find some information on each step below:

- **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 Planning.** The outcome of the analysis is an overview, which risks can be tolerated and which are critical for your business. Ask yourself how you can 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.
25.1.2 What Does Disaster Recovery Mean?

If a system in a production environment has been destroyed (for whatever reasons—be it broken hardware, a misconfiguration or software problems), you need to re-create the system. The recreation can be done either on the same hardware or on compatible replacement hardware. Re-creating a system means more than restoring files from a backup. It also includes preparing the system's storage (with regard to partitioning, file systems, and mount points), and reinstalling the boot loader.

25.1.3 How Does Disaster Recovery With Rear Work?

While the system is up and running, create a backup of the files and create a recovery system on a recovery medium. The recovery system contains a recovery installer.

In case the system has been destroyed, replace broken hardware (if needed), boot the recovery system from the recovery medium and launch the recovery installer. The recovery installer re-creates the system: First, it prepares the storage (partitioning, file systems, mount points), then it restores the files from the backup. Finally, it reinstall the boot loader.

25.1.4 Rear Requirements

To use Rear you need at least two identical systems: the machine that runs your production environment and an identical test machine. “Identical” in this context means that you can, for example, replace a network card with another one using the same Kernel driver.

⚠️ Warning: Identical Drivers Required

If a hardware component does not use the same driver as the one in your production environment, it is not considered identical by Rear.

25.1.5 Rear Version Updates

SUSE Linux Enterprise High Availability Extension 15 SP1 ships with Rear version 2.3, provided by the package `rear23a`.
Note: Find Important Information in Changelogs

Any information about bugfixes, incompatibilities, and other issues can be found in the changelogs of the packages. It is recommended to review also later package versions of Rear in case you need to re-validate your disaster recovery procedure.

Be aware of the following issues with Rear:

- To allow disaster recovery on UEFI systems, you need at least Rear version 1.18.a and the package ebiso. Only this version supports the new helper tool /usr/bin/ebiso. This helper tool is used to create a UEFI-bootable Rear system ISO image.

- If you have a tested and fully functional disaster recovery procedure with one Rear version, do not update Rear. Keep the Rear package and do not change your disaster recovery method!

- Version updates for Rear are provided as separate packages that intentionally conflict with each other to prevent your installed version getting accidentally replaced with another version.

In the following cases you need to completely re-validate your existing disaster recovery procedure:

- For each Rear version update.
- When you update Rear manually.
- For each software that is used by Rear.
- If you update low-level system components such as parted, btrfs and similar.

25.1.6 Limitations with Btrfs

The following limitations apply if you use Btrfs.

Your System Includes Subvolumes, but No Snapshots Subvolumes

At least Rear version 1.17.2.a is required. This version supports re-creating “normal” Btrfs subvolume structure (no snapshot subvolumes).

Your System Includes Snapshot Subvolumes
Warning

Btrfs snapshot subvolumes cannot be backed up and restored as usual with file-based backup software.

While recent snapshot subvolumes on Btrfs file systems need almost no disk space (because of Btrfs's copy-on-write functionality), those files would be backed up as complete files when using file-based backup software. They would end up twice in the backup with their original file size. Therefore, it is impossible to restore the snapshots as they have been before on the original system.

Your SLE System Needs Matching Rear Configuration

For example, the setup in SLE12 GA, SLE12 SP1, and SLE12 SP2 have several different incompatible Btrfs default structures. As such, it is crucial to use a matching Rear configuration file. See the example files /usr/share/rear/conf/examples/SLE12*-btrfs-example.conf.

25.1.7 Scenarios and Backup Tools

Rear can create a disaster recovery system (including a system-specific recovery installer) that can be booted from a local medium (like a hard disk, a flash disk, a DVD/CD-R) or via PXE. The backup data can be stored on a network file system, for example NFS, as described in Example 25.1.

Rear does not replace a file backup, but complements it. By default, Rear supports the generic tar command, and several third-party backup tools (such as Tivoli Storage Manager, QNetix Galaxy, Symantec NetBackup, EMC NetWorker, or HP DataProtector). Refer to Example 25.2 for an example configuration of using Rear with EMC NetWorker as backup tool.

25.1.8 Basic Steps

For a successful recovery with Rear in case of disaster, you need to execute the following basic steps:

Setting Up Rear and Your Backup Solution

This includes tasks like editing the Rear configuration file, adjusting the Bash scripts, and configuring the backup solution that you want to use.
Creating the Recovery Installation System

While the system to be protected is up and running use the `rear mkbackup` command to create a file backup and to generate a recovery system that contains a system-specific Rear recovery installer.

Testing the Recovery Process

Whenever you have created a disaster recovery medium with Rear, test the disaster recovery process thoroughly. It is essential to use a test machine that has identical hardware like the one that is part of your production environment. For details, refer to Section 25.1.4, “Rear Requirements”.

Recovering from Disaster

After a disaster has occurred, replace any broken hardware (if necessary). Then boot the Rear recovery system and start the recovery installer with the `rear recover` command.

25.2 Setting Up Rear and Your Backup Solution

To set up Rear, you need to edit at least the Rear configuration file `/etc/rear/local.conf` and, if needed, the Bash scripts that are part of the Rear framework.

In particular, you need to define the following tasks that Rear should do:

- **When your system is booted with UEFI.** If your system boots with a UEFI boot loader, install the package `ebiso` and add the following line into `/etc/rear/local.conf`:

  ```
  ISO_MKISOFS_BIN=/usr/bin/ebiso
  ```

- **How to back up files and how to create and store the disaster recovery system.** This needs to be configured in `/etc/rear/local.conf`.

- **What to re-create exactly (partitioning, file systems, mount points, etc.).** This can be defined in `/etc/rear/local.conf` (for example, what to exclude). To re-create non-standard systems, you may need to enhance the Bash scripts.

- **How the recovery process works.** To change how Rear generates the recovery installer, or to adapt what the Rear recovery installer does, you need to edit the Bash scripts.

To configure Rear, add your options to the `/etc/rear/local.conf` configuration file. (The former configuration file `/etc/rear/sites.conf` has been removed from the package. However, if you have such a file from your last setup, Rear will still use it.)
All Rear configuration variables and their default values are set in `/usr/share/rear/conf/default.conf`. Some example files (`*example.conf`) for user configurations (for example, what is set in `/etc/rear/local.conf`) are available in the `examples` subdirectory. Find more information in the Rear man page.

You should start with a matching example configuration file as template and adapt it as needed to create your particular configuration file. Copy various options from several example configuration files and paste them into your specific `/etc/rear/local.conf` file that matches your particular system. Do not use original example configuration files, because they provide an overview of variables that can be used for specific setups.

**EXAMPLE 25.1: USING AN NFS SERVER TO STORE THE FILE BACKUP**

Rear can be used in different scenarios. The following example uses an NFS server as storage for the file backup.


2. Define the configuration for your NFS server in the `/etc/exports` file. Make sure the directory on the NFS server (where you want the backup data to be available), has the right mount options. For example:

   ```
   /srv/nfs *(...),rw,no_root_squash,[...])
   ```

   Replace `/srv/nfs` with the path to your backup data on the NFS server and adjust the mount options. You might need `no_root_squash` as the `rear mkbackup` command runs as `root`.

3. Adjust the various `BACKUP` parameters in the configuration file `/etc/rear/local.conf` to make Rear store the file backup on the respective NFS server. Find examples in your installed system under `/usr/share/rear/conf/examples/SLE*-example.conf`.

**EXAMPLE 25.2: USING THIRD-PARTY BACKUP TOOLS LIKE EMC NETWORKER**

Using third-party backup tools instead of `tar` requires appropriate settings in the Rear configuration file.

The following is an example configuration for EMC NetWorker. Add this configuration snippet to `/etc/rear/local.conf` and adjust it according to your setup:

```
BACKUP=NSR
```
25.3 Creating the Recovery Installation System

After you have configured Rear as described in Section 25.2, create the recovery installation system (including the Rear recovery installer) plus the file backup with the following command:

```
rear -d -D mkbackup
```

It executes the following steps:

1. Analyzing the target system and gathering information, in particular about the disk layout (partitioning, file systems, mount points) and about the boot loader.
2. Creating a bootable recovery system with the information gathered in the first step. The resulting Rear recovery installer is specific to the system that you want to protect from disaster. It can only be used to re-create this specific system.
3. Calling the configured backup tool to back up system and user files.

25.4 Testing the Recovery Process

After having created the recovery system, test the recovery process on a test machine with identical hardware. See also Section 25.1.4, “Rear Requirements”. Make sure the test machine is correctly set up and can serve as a replacement for your main machine.

**Warning: Extensive Testing on Identical Hardware**

Thorough testing of the disaster recovery process on machines is required. Test the recovery procedure on a regular basis to ensure everything works as expected.

**PROCEDURE 25.1: PERFORMING A DISASTER RECOVERY ON A TEST MACHINE**

1. Create a recovery medium by burning the recovery system that you have created in Section 25.3 to a DVD or CD. Alternatively, you can use a network boot via PXE.
2. Boot the test machine from the recovery medium.

3. From the menu, select Recover.

4. Log in as root (no password needed).

5. Enter the following command to start the recovery installer:

   ```bash
   rear -d -D recover
   ```

   For details about the steps that Rear takes during the process, see Recovery Process.

6. After the recovery process has finished, check whether the system has been successfully re-created and can serve as a replacement for your original system in the production environment.

25.5 Recovering from Disaster

In case a disaster has occurred, replace any broken hardware if necessary. Then proceed as described in Procedure 25.1, using either the repaired machine (or a tested, identical machine that can serve as a replacement for your original system).

The `rear recover` command will execute the following steps:

**RECOVERY PROCESS**

1. Restoring the disk layout (partitions, file systems, and mount points).

2. Restoring the system and user files from the backup.

3. Restoring the boot loader.

25.6 For More Information


- `rear` man page

- `/usr/share/doc/packages/rear/`
IV Appendix

A Troubleshooting 339
B Naming Conventions 349
C Cluster Management Tools (Command Line) 350
D Running Cluster Reports Without root Access 352
A Troubleshooting

Strange problems may occur that are not easy to understand, especially when start-
ing 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 extension to SUSE Linux Enterprise
Server 15 SP1 on each of the cluster nodes and if the High Availability pattern is installed
on each of the machines as described in the Installation and Setup Quick Start.

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 Chapter 4, Using the YaST
Cluster Module.
Check if the communication channels and options configured in /etc/corosync/coro-
sync.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 except for 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 fire-
wall, the nodes cannot see each other. When doing the initial setup with YaST or the boot-
strap scripts (as described in Chapter 4, Using the YaST Cluster Module or the Article “Installation
and Setup Quick Start”, respectively), the firewall settings are usually automatically adjusted.
To make sure the mcastport is not blocked by the firewall, check the firewall settings on
each node.
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 # crm cluster status
```

In case they are not running, start them by executing the following command:

```
root # crm cluster start
```

### A.2 Logging

**Where to find the log files?**

Pacemaker writes its log files into the `/var/log/pacemaker` directory. The main Pacemaker log file is `/var/log/pacemaker/pacemaker.log`. In case you cannot find the log files, check the logging settings in `/etc/sysconfig/pacemaker`, Pacemaker's own configuration file. If `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 log files?**

The `pacemaker-execd` 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 \( \text{Ctrl} - \text{C} \)). 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 bob
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:
```


**How to view logs?**

For a more detailed view of what is happening in your cluster, use the following command:

```
root #crm history log [NODE]
```

Replace *NODE* with the node you want to examine, or leave it empty. See *Section A.5, “History”* for further information.

### 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 8.4.4, “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 \n/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?**

The terminated node might be 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 because of a failure which happened in the past and which was not “cleaned”. Or it may be because of an earlier administrative action, that is a location constraint with a negative score. Such a location constraint is inserted by the `crm resource move` command, for example.

**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 from outlet names, for example. That depends on the nature of the STONITH plugin. `pacemaker-fenced` 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 `pacemaker-fenced` does not find the node in any of the host lists provided by running STONITH resources, it will ask `pacemaker-fenced` instances on other nodes. If the target node does not show up in the host lists of other `pacemaker-fenced` 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  History

How to retrieve status information or a log from a failed resource?
Use the `history` command and its subcommand `resource`:

```
root # crm history resource NAME1
```

This gives you a full transition log for the given resource only. However, it is possible to investigate more than one resource. Append the resource names after the first.

If you followed some naming conventions (see section), the `resource` command makes it easier to investigate a group of resources. For example, this command investigates all primitives starting with `db`:

```
root # crm history resource db*
```

View the log file in `/var/cache/crm/history/live/alice/ha-log.txt`.

**How can I reduce the history output?**

There are two options for the `history` command:

- Use `exclude`

- Use `timeframe`

The `exclude` command let you set an additive regular expression that excludes certain patterns from the log. For example, the following command excludes all SSH, systemd, and kernel messages:

```
root # crm history exclude ssh|systemd|kernel.
```

With the `timeframe` command you limit the output to a certain range. For example, the following command shows all the events on August 23rd from 12:00 to 12:30:

```
root # crm history timeframe "Aug 23 12:00" "Aug 23 12:30"
```

**How can I store a “session” for later inspection?**

When you encounter a bug or an event that needs further examination, it is useful to store all the current settings. This file can be sent to support or viewed with `bzless`. For example:

```
crm(live)history# timeframe "Oct 13 15:00" "Oct 13 16:00"
crm(live)history# session save tux-test
crm(live)history# session pack
Report saved in '/root/tux-test.tar.bz2'
```
A.6 Hawk2

Replacing the Self-Signed Certificate

To avoid the warning about the self-signed certificate on first Hawk2 start-up, replace the automatically created certificate with your own certificate (or a certificate that was signed by an official Certificate Authority, CA):

1. Replace `/etc/hawk/hawk.key` with the private key.
2. Replace `/etc/hawk/hawk.pem` with the certificate that Hawk2 should present.
3. Restart the Hawk2 services to reload the new certificate:

   ```bash
   root # systemctl restart hawk-backend hawk
   ```

Change ownership of the files to `root:haclient` and make the files accessible to the group:

```bash
chown root:haclient /etc/hawk/hawk.key /etc/hawk/hawk.pem
chmod 640 /etc/hawk/hawk.key /etc/hawk/hawk.pem
```

A.7 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 host names 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:

```bash
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
-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 and 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`. 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 pacemaker-execd: [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 `crm report` to create a report. This tool compiles:

- 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 alice -n bob
```
The command extracts all information since 0am on the hosts alice and bob 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 \texttt{crm report} tool tries to remove any sensitive information from the CIB and the PE input files, however, it cannot know everything. If you have more sensitive information, supply additional patterns with the \-p option (see man page). The log files and the \texttt{crm_mon}, \texttt{ccm_tool}, and \texttt{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 \texttt{crm report}.

After \texttt{crm report} has analyzed all the relevant log files and created the directory (or archive), check the log files for an uppercase \texttt{ERROR} string. The most important files in the top level directory of the report are:

- \texttt{analysis.txt}
  
  Compares files that should be identical on all nodes.

- \texttt{corosync.txt}
  
  Contains a copy of the Corosync configuration file.

- \texttt{crm_mon.txt}
  
  Contains the output of the \texttt{crm_mon} command.

- \texttt{description.txt}
  
  Contains all cluster package versions on your nodes. There is also the \texttt{sysinfo.txt} file which is node specific. It is linked to the top directory.

  This file can be used as a template to describe the issue you encountered and post it to https://github.com/ClusterLabs/crmsh/issues.
members.txt
A list of all nodes

sysinfo.txt
Contains a list of all relevant package names and their versions. Additionally, there
is also a list of configuration files which are different from the original RPM package.
Node-specific files are stored in a subdirectory named by the node’s name. It contains a
copy of the directory */etc* of the respective node.
In case you need to simplify the arguments, set your default values in the configuration
file */etc/crm/crm.conf*, section *report*. Further information is written in the man page
*man 8 crmsh_hb_report*.

A.8 For More Information

For additional information about high availability on Linux, including configuring cluster re-
sources and managing and customizing a High Availability cluster, see [http://clusterlabs.org/wi-
ki/Documentation](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, charlie, doro, and eris

Cluster Site Names
Cluster sites are named after cities:
amsterdam, berlin, canberra, dublin, fukuoka, gizeh, hanoi, and istanbul

Cluster Resources

<table>
<thead>
<tr>
<th>Primitives</th>
<th>No prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Groups</td>
<td>Prefix g-</td>
</tr>
<tr>
<td>Clones</td>
<td>Prefix cl-</td>
</tr>
<tr>
<td>Promotable Clones</td>
<td>(formerly known as multi-state resources) Prefix ms-</td>
</tr>
</tbody>
</table>

Constraints

<table>
<thead>
<tr>
<th>Ordering constraints</th>
<th>Prefix o-</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location constraints</td>
<td>Prefix loc-</td>
</tr>
<tr>
<td>Colocation constraints</td>
<td>Prefix col-</td>
</tr>
</tbody>
</table>
C  Cluster Management Tools (Command Line)

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. Usually 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 Running Cluster Reports Without root Access

All cluster nodes must be able to access each other via SSH. Tools like crm report (for troubleshooting) and Hawk2's History Explorer require passwordless SSH access between the nodes, otherwise they can only collect data from the current node.

If passwordless SSH root access does not comply with regulatory requirements, you can use a work-around for running cluster reports. It consists of the following basic steps:

1. Creating a dedicated local user account (for running crm report).
2. Configuring passwordless SSH access for that user account, ideally by using a non-standard SSH port.
3. Configuring sudo for that user.
4. Running crm report as that user.

By default when crm report is run, it attempts to log in to remote nodes first as root, then as user hacluster. However, if your local security policy prevents root login using SSH, the script execution will fail on all remote nodes. Even attempting to run the script as user hacluster will fail because this is a service account, and its shell is set to /bin/false, which prevents login. Creating a dedicated local user is the only option to successfully run the crm report script on all nodes in the High Availability cluster.

D.1 Creating a Local User Account

In the following example, we will create a local user named hareport from command line. The password can be anything that meets your security requirements. Alternatively, you can create the user account and set the password with YaST.

PROCEDURE D.1: CREATING A DEDICATED USER ACCOUNT FOR RUNNING CLUSTER REPORTS

1. Start a shell and create a user hareport with a home directory /home/hareport:

   root # useradd -m -d /home/hareport -c "HA Report" hareport

2. Set a password for the user:

   root # passwd hareport

3. When prompted, enter and re-enter a password for the user.
D.2 Configuring a Passwordless SSH Account

PROCEDURE D.2: CONFIGURING THE SSH DAEMON FOR A NON-STANDARD PORT

By default, the SSH daemon and the SSH client talk and listen on port 22. If your network security guidelines require the default SSH port to be changed to an alternate high numbered port, you need to modify the daemon's configuration file /etc/ssh/sshd_config.

1. To modify the default port, search the file for the Port line, uncomment it and edit it according to your wishes. For example, set it to:

   Port 5022

2. If your organization does not permit the root user to access other servers, search the file for the PermitRootLogin entry, uncomment it and set it to no:

   PermitRootLogin no

3. Alternatively, add the respective lines to the end of the file by executing the following commands:

   root # echo "PermitRootLogin no" >> /etc/ssh/sshd_config
   root # echo "Port 5022" >> /etc/ssh/sshd_config

4. After modifying /etc/ssh/sshd_config, restart the SSH daemon to make the new settings take effect:

   root # systemctl restart sshd

Important: Same Settings Are Required On Each Cluster Node

Repeat the SSH daemon configuration above on each cluster node.

PROCEDURE D.3: CONFIGURING THE SSH CLIENT FOR A NON-STANDARD PORT

If the SSH port change is going to be made on all nodes in the cluster, it is useful to modify the SSH configuration file, /etc/ssh/sshd_config.
1. To modify the default port, search the file for the Port line, uncomment it and edit it according to your wishes. For example, set it to:

   Port 5022

2. Alternatively, add the respective line to the end of the file by executing the following commands:

   root # echo “Port 5022” >> /etc/ssh/ssh_config

**Note: Settings Only Required on One Node**

The SSH client configuration above is only needed on the node on which you want to run the cluster report.

Alternatively, you can use the -X option to run the crm report with a custom SSH port or even make crm report use your custom SSH port by default. For details, see Procedure D.5, “Generating a Cluster Report Using a Custom SSH Port”.

**PROCEDURE D.4: CONFIGURING SHARED SSH KEYS**

You can access other servers using SSH and not be asked for a password. While this may appear insecure at first sight, it is actually a very secure access method since the users can only access servers that their public key has been shared with. The shared key must be created as the user that will use the key.

1. Log in to one of the nodes with the user account that you have created for running cluster reports (in our example above, the user account was hareport).

2. Generate a new key:

   hareport > ssh-keygen -t rsa

   This command will generate a 2048 bit key by default. The default location for the key is ~/.ssh/. You are asked to set a passphrase on the key. However, do not enter a passphrase because for passwordless login there must not be a passphrase on the key.

3. After the keys have been generated, copy the public key to each of the other nodes (including the node where you created the key):

   hareport > ssh-copy-id -i ~/.ssh/id_rsa.pub HOSTNAME_OR_IP
In the command, you can either use the DNS name for each server, an alias, or the IP address. During the copy process you will be asked to accept the host key for each node, and you will need to provide the password for the `hareport` user account (this will be the only time you need to enter it).

4. After the key is shared to all cluster nodes, test if you can log in as user `hareport` to the other nodes by using passwordless SSH:

```
    hareport > ssh HOSTNAME_OR_IP
```

You should be automatically connected to the remote server without being asked to accept a certificate or enter a password.

**Note: Settings Only Required on One Node**

If you intend to run the cluster report from the same node each time, it is sufficient to execute the procedure above on this node only. Otherwise repeat the procedure on each node.

### D.3 Configuring `sudo`

The `sudo` command allows a regular user to quickly become `root` and issue a command, with or without providing a password. Sudo access can be given to all root-level commands or to specific commands only. Sudo typically uses aliases to define the entire command string.

To configure `sudo` either use `visudo` *(not vi)* or YaST.

**Warning: Do Not Use vi**

For `sudo` configuration from command line, you must edit the `sudoers` file as `root` with `visudo`. Using any other editor may result in syntax or file permission errors that prevent `sudo` from running.

1. Log in as `root`.
2. To open the `/etc/sudoers` file, enter `visudo`.
3. Look for the following categories: `Host alias specification`, `User alias specification`, `Cmd alias specification`, and `Runas alias specification`. 
4. Add the following entries to the respective categories in `/etc/sudoers`:

```plaintext
Host_Alias CLUSTER = alice,bob,charlie
User_Alias HA = hareport
Cmd_Alias HA_ALLOWED = /bin/su, /usr/sbin/crm report *
Runas_Alias R = root
```

1. The host alias defines on which server (or range of servers) the sudo user has rights to issue commands. In the host alias you can use DNS names, or IP addresses, or specify an entire network range (for example, `172.17.12.0/24`). To limit the scope of access you should specify the host names for the cluster nodes only.

2. The user alias allows you to add multiple local user accounts to a single alias. However, in this case you could avoid creating an alias since only one account is being used. In the example above, we added the `hareport` user which we have created for running cluster reports.

3. The command alias defines which commands can be executed by the user. This is useful if you want to limit what the non-root user can access when using `sudo`. In this case the `hareport` user account will need access to the commands `crm report` and `su`.

4. The runas alias specifies the account that the command will be run as. In this case `root`.

5. Search for the following two lines:

```plaintext
Defaults targetpw
ALL ALL=(ALL) ALL
```

As they would conflict with the setup we want to create, disable them:

```plaintext
#Defaults targetpw
#ALL ALL=(ALL) ALL
```


7. After having defined the aliases above, you can now add the following rule there:

```
HA CLUSTER = (R) NOPASSWD:HA_ALLOWED
```

The `NOPASSWD` option ensures that the user `hareport` can execute the cluster report without providing a password.
Important: Same sudo Configuration Is Required on Each Cluster Node

This sudo configuration must be made on all nodes in the cluster. No other changes are needed for sudo and no services need to be restarted.

D.4 Generating a Cluster Report

To run cluster reports with the settings you have configured above, you need to be logged in to one of the nodes as user `hareport`. To start a cluster report, use the `crm report` command. For example:

```bash
root # crm report -f 0:00 -n "alice bob charlie"
```

This command will extract all information since 0 am on the named nodes and create a `*.tar.bz2` archive named `pcmk-DATE.tar.bz2` in the current directory.

PROCEDURE D.5: GENERATING A CLUSTER REPORT USING A CUSTOM SSH PORT

1. When using a custom SSH port, use the `-X` with `crm report` to modify the client’s SSH port. For example, if your custom SSH port is 5022, use the following command:

```bash
root # crm report -X "-p 5022" [...]
```

2. To set your custom SSH port permanently for `crm report`, start the interactive crm shell:

```bash
crm options
```

3. Enter the following:

```bash
crm(live)options# set core.report_tool_options "-X -oPort=5022"
```
**Glossary**

**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 take over other services from the other node. Compare with 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, re-
sources, constraints and the relationship to each other). It is written in XML and resides
in memory. A primary 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 primary CIB.

cluster
A high-performance cluster is a group of computers (real or virtual) sharing the application
load 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).

cluster stack
The ensemble of software technologies and components which compose a cluster.

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 management entity responsible for coordinating all non-local interactions in a High
Availability cluster. The High Availability Extension uses Pacemaker as CRM. The CRM is
implemented as pacemaker-controld. It interacts with several components: local resource
managers, both on its own node and on the other nodes, non-local CRMs, administrative
commands, the fencing functionality, and the membership layer.

crmsh
The command line utility crmsh manages your cluster, nodes, and resources.
See *Chapter 8, 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)**
The DC is elected from all nodes in the cluster. This happens if there is no DC yet or if the current DC leaves the cluster for any reason. 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. All other nodes get their configuration and resource allocation information from the current DC.

**Disaster**
Unexpected interruption of critical infrastructure induced by nature, humans, hardware failure, or software bugs.

**Disaster Recover Plan**
A strategy to recover from a disaster with minimum impact on IT infrastructure.

**Disaster Recovery**
Disaster recovery is the process by which a business function is restored to the normal, steady state after a disaster.

**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 if a node fails.

fencing
Describes the concept of preventing access to a shared resource by isolated or failing cluster members. There are two classes of fencing: resource level fencing and node level fencing. Resource level fencing ensures exclusive access to a given resource. Node level fencing prevents a failed node from accessing shared resources entirely and prevents resources from running on a node whose status is uncertain. This is usually done in a simple and abrupt way: reset or power off the node.

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 need to cope with limited network bandwidth and high latency. Storage is replicated asynchronously.

Geo cluster (geographically dispersed cluster)
See Geo cluster.

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)
The local resource manager is located between the Pacemaker layer and the resources layer on each node. It is implemented as pacemaker-execd daemon. Through this daemon, Pacemaker can start, stop, and monitor resources.

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.

node
Any computer (real or virtual) that is a member of a cluster and invisible to the user.

pacemaker-controld (cluster controller daemon)
The CRM is implemented as daemon, pacemaker-controld. It has an instance on each cluster node. All cluster decision-making is centralized by electing one of the pacemaker-controld instances to act as a primary. If the elected pacemaker-controld process fails (or the node it ran on), a new one is established.

PE (policy engine)
The policy engine is implemented as pacemaker-schedulerd daemon. When a cluster transition is needed, based on the current state and configuration, pacemaker-schedulerd calculates the expected next state of the cluster. It determines what actions need to be scheduled to achieve the next state.

quorum
In a cluster, a cluster partition is defined to have quorum (be “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 different kinds of resource agents: For details, see Section 6.3.2, “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 Block Device)
Provides a node fencing mechanism through the exchange of messages via shared block storage (SAN, iSCSI, FCoE, etc.). Can also be used in diskless mode. Needs a hardware or software 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. In a Pacemaker cluster, the implementation of node level fencing is STONITH. For this, Pacemaker comes with a fencing subsystem, pacemaker-fenced.

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).
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. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for free software, because free software needs free documentation: a free program needs a manual that documents how to use that program. Nothing in this License restricts the use of software in a free program.

The parties to a "瑪j=500x5000
documentation License" are the copyright holder and a licensee(s), who replaces with "you" and "your".

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publisher 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 or a mathematics paper, a "Secondary Section" may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, TeX info 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-generat- ed 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 XX 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.

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 enrol 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 the 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.

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.

GFDL 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.

E GNU Licenses

This appendix contains the GNU Free Documentation License version 1.2.
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 in the Document 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.

L. 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 Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not relabel 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 include 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 these 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 as 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 Docu-
mentation 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.