Pacemaker 2.0
Configuration Explained

An A-Z guide to Pacemaker's Configuration Options

Andrew Beekhof
Pacemaker 2.0 Configuration Explained
An A-Z guide to Pacemaker's Configuration Options
Edition 12

Copyright © 2009-2019 Andrew Beekhof.

The text of and illustrations in this document are licensed under version 4.0 or later of the Creative Commons Attribution-ShareAlike International Public License ("CC-BY-SA")¹.

In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

In addition to the requirements of this license, the following activities are looked upon favorably:
1. If you are distributing Open Publication works on hardcopy or CD-ROM, you provide email notification to the authors of your intent to redistribute at least thirty days before your manuscript or media freeze, to give the authors time to provide updated documents. This notification should describe modifications, if any, made to the document.

2. All substantive modifications (including deletions) be either clearly marked up in the document or else described in an attachment to the document.

3. Finally, while it is not mandatory under this license, it is considered good form to offer a free copy of any hardcopy or CD-ROM expression of the author(s) work.

The purpose of this document is to definitively explain the concepts used to configure Pacemaker. To achieve this, it will focus exclusively on the XML syntax used to configure Pacemaker's Cluster Information Base (CIB).

¹ An explanation of CC-BY-SA is available at https://creativecommons.org/licenses/by-sa/4.0/
Table of Contents

1. Read-Me-First ................................................................. 1
   1.1. The Scope of this Document ..................................................... 1
   1.2. What Is Pacemaker? ............................................................... 1
   1.3. Cluster Architecture ............................................................ 2
   1.4. Pacemaker Architecture ....................................................... 3
   1.5. Node Redundancy Designs .................................................... 4

2. Cluster-Wide Configuration .............................................. 7
   2.1. Configuration Layout .......................................................... 7
   2.2. CIB Properties ................................................................. 8
   2.3. Cluster Options .................................................................. 8

3. Cluster Nodes ..................................................................... 13
   3.1. Defining a Cluster Node ......................................................... 13
   3.2. Where Pacemaker Gets the Node Name ................................... 13
   3.3. Node Attributes ................................................................. 13
       3.3.1. Setting and querying node attributes .............................. 14
       3.3.2. Special node attributes ............................................. 14

4. Cluster Resources ........................................................... 17
   4.1. What is a Cluster Resource? ................................................. 17
   4.2. Resource Classes ............................................................. 17
       4.2.1. Open Cluster Framework ............................................ 18
       4.2.2. Linux Standard Base .................................................. 18
       4.2.3. Systemd ................................................................. 19
       4.2.4. Upstart ................................................................. 19
       4.2.5. System Services ...................................................... 20
       4.2.6. STONITH ............................................................ 20
       4.2.7. Nagios Plugins ....................................................... 20
   4.3. Resource Properties ......................................................... 21
   4.4. Resource Options ............................................................ 21
       4.4.1. Resource Meta-Attributes .......................................... 21
       4.4.2. Setting Global Defaults for Resource Meta-Attributes .... 24
       4.4.3. Resource Instance Attributes .................................... 24
   4.5. Resource Operations ....................................................... 26
       4.5.1. Monitoring Resources for Failure ................................. 27
       4.5.2. Monitoring Resources When Administration is Disabled .... 28
       4.5.3. Setting Global Defaults for Operations ......................... 29
       4.5.4. When Implicit Operations Take a Long Time ................. 29
       4.5.5. Multiple Monitor Operations ..................................... 29
       4.5.6. Disabling a Monitor Operation .................................... 30

5. Resource Constraints ...................................................... 31
   5.1. Scores ........................................................................... 31
       5.1.1. Infinity Math ........................................................ 31
   5.2. Deciding Which Nodes a Resource Can Run On ................. 32
       5.2.1. Location Properties ................................................. 32
List of Figures

1.1. Example Cluster Stack .......................................................................................................... 3
1.2. Internal Components ........................................................................................................... 3
1.3. Active/Passive Redundancy .............................................................................................. 5
1.4. Shared Failover .................................................................................................................. 5
1.5. N to N Redundancy ............................................................................................................ 6
5.1. Visual representation of the four resources’ start order for the above constraints .......... 39
5.2. Visual representation of the start order for two ordered sets of unordered resources ....... 40
5.3. Visual representation of the start order for the three sets defined above ......................... 41
5.4. Visual representation of the above example (resources are placed from left to right) ........ 45
List of Tables

2.1. CIB Properties ...................................................................................................................... 8
2.2. Cluster Options .................................................................................................................... 9
3.1. Node attributes with special significance .............................................................................. 14
4.1. Properties of a Primitive Resource ....................................................................................... 21
4.2. Meta-attributes of a Primitive Resource ................................................................................ 22
4.3. Properties of an Operation .................................................................................................. 26
5.1. Properties of a rsc_location Constraint ................................................................................. 32
5.2. Properties of a rsc_order Constraint ..................................................................................... 35
5.3. Properties of a rsc_colocation Constraint ............................................................................. 37
5.4. Properties of a resource_set ................................................................................................ 38
6.1. Meta-Attributes of an Alert ................................................................................................... 48
6.2. Environment variables passed to alert agents ....................................................................... 50
7.1. Properties of a Rule ............................................................................................................ 53
7.2. Properties of an Expression ................................................................................................ 54
7.3. Built-in node attributes ......................................................................................................... 54
7.4. Properties of a Date Expression .......................................................................................... 55
7.5. Properties of a Date Specification ........................................................................................ 56
8.1. Common Options for a ping Resource ................................................................................... 68
8.2. Allowed Values for Node Health Attributes ........................................................................... 71
8.3. Node Health Strategies ....................................................................................................... 71
9.1. Properties of a Group Resource .......................................................................................... 76
9.2. Properties of a Clone Resource ........................................................................................... 78
9.3. Clone-specific configuration options ..................................................................................... 78
9.4. Additional colocation constraint options for promotable clone resources ....................... 80
9.5. Additional colocation set options relevant to promotable clone resources ......................... 81
9.6. Additional ordered set options relevant to promotable clone resources ............................... 81
9.7. Role implications of OCF return codes ................................................................................. 83
9.8. Environment variables supplied with Clone notify actions ................................................... 83
9.9. Extra environment variables supplied for promotable clones ............................................. 85
9.10. Properties of a Bundle ....................................................................................................... 90
9.11. Properties of a Bundle's Docker Element ........................................................................... 90
9.12. Properties of a Bundle's rkt Element .................................................................................. 91
9.13. Properties of a Bundle's Network Element .......................................................................... 92
9.15. Properties of a Bundle's Storage-Mapping Element .......................................................... 93
12.1. Additional Properties of Fencing Resources ...................................................................... 108
12.2. Properties of Fencing Levels ............................................................................................ 116
13.1. Authoritative Sources for State Information ....................................................................... 123
13.2. Node Status Fields .......................................................................................................... 123
13.3. Contents of an lrm_rsc_op job .......................................................................................... 125
List of Examples

2.1. An empty configuration ................................................................. 7
3.1. Example Corosync cluster node entry ........................................ 13
3.2. Result of using crm_attribute to specify which kernel pcmk-1 is running ........................................ 14
4.1. A system resource definition .................................................... 21
4.2. An OCF resource definition ..................................................... 21
4.3. An LSB resource with cluster options ....................................... 24
4.4. An example OCF resource with instance attributes ..................... 24
4.5. Displaying the metadata for the Dummy resource agent template  ..................................................... 25
4.6. An OCF resource with a non-default start timeout ................. 26
4.7. An OCF resource with a recurring health check ...................... 28
4.8. An OCF resource with custom timeouts for its implicit actions .... 29
4.9. An OCF resource with two recurring health checks, performing different levels of checks specified via OCF_CHECK_LEVEL ................................................................. 29
4.10. Example of an OCF resource with a disabled health check .... 30
5.1. Opt-in location constraints for two resources ......................... 34
5.2. Opt-out location constraints for two resources ....................... 34
5.3. Constraints where a resource prefers two nodes equally .......... 34
5.4. Optional and mandatory ordering constraints ......................... 36
5.5. Mandatory colocation constraint for two resources .................. 37
5.6. Mandatory anti-colocation constraint for two resources ........... 37
5.7. Advisory colocation constraint for two resources ................... 38
5.8. A set of 3 resources .............................................................. 38
5.9. A chain of ordered resources .................................................. 39
5.10. A chain of ordered resources expressed as a set ..................... 39
5.11. Ordered sets of unordered resources ...................................... 40
5.12. Advanced use of set ordering - Three ordered sets, two of which are internally unordered ...... 41
5.13. Resource Set "OR" logic: Three ordered sets, where the first set is internally unordered with "OR" logic ........................................................................................................ 42
5.14. Colocation chain as individual constraints, where A is placed first, then B, then C, then D ...... 42
5.15. Equivalent colocation chain expressed using resource_set ........ 43
5.16. Using colocated sets to specify a shared dependency ............ 43
5.17. Colocation in which the members of the middle set have no interdependencies, and the last set listed applies only to instances in the master role ........................................... 44
6.1. Simple alert configuration ....................................................... 47
6.2. Alert configuration with recipient .......................................... 47
6.3. Alert configuration with meta-attributes ................................ 48
6.4. Alert configuration with instance attributes .......................... 49
6.5. Alert configuration to receive only node events and fencing events ............................................... 49
6.6. Alert configuration to be called when certain node attributes change ........................................... 49
6.7. Sending cluster events as SNMP traps .................................... 50
6.8. Sending cluster events as e-mails .......................................... 50
7.1. True if now is any time in the year 2005 ................................. 56
7.2. Equivalent expression .......................................................... 57
7.3. 9am-5pm Monday-Friday ..................................................... 57
7.4. 9am-6pm Monday through Friday or anytime Saturday .......... 57
7.5. 9am-5pm or 9pm-12am Monday through Friday .................... 57
7.6. Mondays in March 2005 ....................................................... 57
7.7. A full moon on Friday the 13th .............................................. 58
7.8. Prevent myApacheRsc from running on c001n03 .................... 58
7.9. Prevent myApacheRsc from running on c001n03 - expanded version .............................................. 58
7.10. A sample nodes section for use with score-attribute ............. 59
Preface

Table of Contents

1. Document Conventions ................................................................. xiii
   1.1. Typographic Conventions ....................................................... xiii
   1.2. Pull-quote Conventions ......................................................... xiv
   1.3. Notes and Warnings ............................................................. xv
2. We Need Feedback! ....................................................................... xv

1. Document Conventions

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

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

1.1. Typographic Conventions

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

Mono-spaced Bold

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

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

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

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

Press Enter to execute the command.

Press Ctrl+Alt+F2 to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

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

\(^1\) https://fedorahosted.org/liberation-fonts/
Preface

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

**Proportional Bold**

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

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

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

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

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

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

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

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

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

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

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

Publican is a DocBook publishing system.

1.2. Pull-quote Conventions

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

Output sent to a terminal is set in **mono-spaced roman** and presented thus:
1.3. Notes and Warnings
Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

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

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

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

2. We Need Feedback!
Preface

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla against the product Pacemaker.

When submitting a bug report, be sure to mention the manual's identifier: Pacemaker_Explained

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

2 http://bugs.clusterlabs.org
Read-Me-First

Table of Contents

1.1. The Scope of this Document ........................................................................................................ 1
1.2. What Is Pacemaker? ................................................................................................................ 1
1.3. Cluster Architecture .................................................................................................................. 2
1.4. Pacemaker Architecture ............................................................................................................ 3
1.5. Node Redundancy Designs ......................................................................................................... 4

1.1. The Scope of this Document

This document is intended to be an exhaustive reference for configuring Pacemaker.

To achieve this, it focuses on the XML syntax used to configure the CIB. For those that are allergic to XML, multiple higher-level front-ends (both command-line and GUI) are available. These tools will not be covered at all in this document.

Users may be interested in other parts of the Pacemaker documentation set, such as Clusters from Scratch, a step-by-step guide to setting up an example cluster, and Pacemaker Administration, a guide to maintaining a cluster.

1.2. What Is Pacemaker?

Pacemaker is a high-availability cluster resource manager — software that runs on a set of hosts (a cluster of nodes) in order to preserve integrity and minimize downtime of desired services (resources).

It is maintained by the ClusterLabs community.

Pacemaker’s key features include:

• Detection of and recovery from node- and service-level failures

• Ability to ensure data integrity by fencing faulty nodes

• Support for one or more nodes per cluster

• Support for multiple resource interface standards (anything that can be scripted can be clustered)

• Support (but no requirement) for shared storage

• Support for practically any redundancy configuration (active/passive, N+1, etc.)

• Automatically replicated configuration that can be updated from any node

• Ability to specify cluster-wide relationships between services, such as ordering, colocation and anti-colocation

---

1 I hope, however, that the concepts explained here make the functionality of these tools more easily understood.

2 https://www.clusterlabs.org/pacemaker/doc/

3 Cluster is sometimes used in other contexts to refer to hosts grouped together for other purposes, such as high-performance computing (HPC), but Pacemaker is not intended for those purposes.

4 https://www.ClusterLabs.org/
Chapter 1. Read-Me-First

- Support for advanced service types, such as clones (services that need to be active on multiple nodes), stateful resources (clones that can run in one of two modes), and containerized services

- Unified, scriptable cluster management tools

**Fencing**

Fencing, also known as STONITH (an acronym for Shoot The Other Node In The Head), is the ability to ensure that it is not possible for a node to be running a service. This is accomplished via fence devices such as intelligent power switches that cut power to the target, or intelligent network switches that cut the target’s access to the local network.

Pacemaker represents fence devices as a special class of resource.

A cluster cannot safely recover from certain failure conditions, such as an unresponsive node, without fencing.

1.3. Cluster Architecture

At a high level, a cluster can viewed as having these parts (which together are often referred to as the cluster stack):

- **Resources**: These are the reason for the cluster’s being — the services that need to be kept highly available.

- **Resource agents**: These are scripts or operating system components that start, stop, and monitor resources, given a set of resource parameters. These provide a uniform interface between Pacemaker and the managed services.

- **Fence agents**: These are scripts that execute node fencing actions, given a target and fence device parameters.

- **Cluster membership layer**: This component provides reliable messaging, membership, and quorum information about the cluster. Currently, Pacemaker supports Corosync\(^5\) as this layer.

- **Cluster resource manager**: Pacemaker provides the brain that processes and reacts to events that occur in the cluster. These events may include nodes joining or leaving the cluster; resource events caused by failures, maintenance, or scheduled activities; and other administrative actions. To achieve the desired availability, Pacemaker may start and stop resources and fence nodes.

- **Cluster tools**: These provide an interface for users to interact with the cluster. Various command-line and graphical (GUI) interfaces are available.

Most managed services are not, themselves, cluster-aware. However, many popular open-source cluster filesystems make use of a common Distributed Lock Manager (DLM), which makes direct use of Corosync for its messaging and membership capabilities and Pacemaker for the ability to fence nodes.

---

\(^5\) [http://www.corosync.org/](http://www.corosync.org/)
1.4. Pacemaker Architecture
Pacemaker itself is composed of multiple daemons that work together:

- pacemakerd
- pacemaker-atrtd
- pacemaker-based
- pacemaker-controld
- pacemaker-execd
- pacemaker-fenced
- pacemaker-schedulerd

---

Figure 1.1. Example Cluster Stack

Figure 1.2. Internal Components
The Pacemaker master process (pacemakerd) spawns all the other daemons, and respawns them if they unexpectedly exit.

The *Cluster Information Base* (CIB) is an **XML**\(^6\) representation of the cluster’s configuration and the state of all nodes and resources. The *CIB manager* (pacemaker-based) keeps the CIB synchronized across the cluster, and handles requests to modify it.

The attribute manager (pacemaker-atrtd) maintains a database of attributes for all nodes, keeps it synchronized across the cluster, and handles requests to modify them. These attributes are usually recorded in the CIB.

Given a snapshot of the CIB as input, the *scheduler* (pacemaker-schedulerd) determines what actions are necessary to achieve the desired state of the cluster.

The *local executor* (pacemaker-execld) handles requests to execute resource agents on the local cluster node, and returns the result.

The *fencer* (pacemaker-fenced) handles requests to fence nodes. Given a target node, the fencer decides which cluster node(s) should execute which fencing device(s), and calls the necessary fencing agents (either directly, or via requests to the fencer peers on other nodes), and returns the result.

The *controller* (pacemaker-controld) is Pacemaker’s coordinator, maintaining a consistent view of the cluster membership and orchestrating all the other components.

Pacemaker centralizes cluster decision-making by electing one of the controller instances as the *Designated Controller (DC)*. Should the elected DC process (or the node it is on) fail, a new one is quickly established. The DC responds to cluster events by taking a current snapshot of the CIB, feeding it to the scheduler, then asking the executors (either directly on the local node, or via requests to controller peers on other nodes) and the fencer to execute any necessary actions.

### Old daemon names

<table>
<thead>
<tr>
<th>Old name</th>
<th>New name</th>
</tr>
</thead>
<tbody>
<tr>
<td>attrd</td>
<td>pacemaker-atrtd</td>
</tr>
<tr>
<td>cib</td>
<td>pacemaker-based</td>
</tr>
<tr>
<td>crmd</td>
<td>pacemaker-controld</td>
</tr>
<tr>
<td>lrmd</td>
<td>pacemaker-execld</td>
</tr>
<tr>
<td>stonithd</td>
<td>pacemaker-fenced</td>
</tr>
<tr>
<td>pacemaker_remoted</td>
<td>pacemaker-remoted</td>
</tr>
</tbody>
</table>

1.5. Node Redundancy Designs

Pacemaker supports practically any *node redundancy configuration*\(^7\) including *Active/Active, Active/Passive, N+1, N+M, N-to-1 and N-to-N*.

---


\(^7\) [https://en.wikipedia.org/wiki/High-availability_cluster#Node_configurations](https://en.wikipedia.org/wiki/High-availability_cluster#Node_configurations)
Active/passive clusters with two (or more) nodes using Pacemaker and DRBD[^1] are a cost-effective high-availability solution for many situations. One of the nodes provides the desired services, and if it fails, the other node takes over.

![Active/Passive Redundancy](image1)

**Figure 1.3. Active/Passive Redundancy**

Pacemaker also supports multiple nodes in a shared-failover design, reducing hardware costs by allowing several active/passive clusters to be combined and share a common backup node.

![Shared Failover](image2)

**Figure 1.4. Shared Failover**

When shared storage is available, every node can potentially be used for failover. Pacemaker can even run multiple copies of services to spread out the workload.

[^1]: https://en.wikipedia.org/wiki/Distributed_Replicated_Block_Device:
Figure 1.5. N to N Redundancy
Chapter 2.

Cluster-Wide Configuration

Table of Contents

2.1. Configuration Layout ................................................................. 7
2.2. CIB Properties ................................................................. 8
2.3. Cluster Options ................................................................. 8

2.1. Configuration Layout

The cluster is defined by the Cluster Information Base (CIB), which uses XML notation. The simplest CIB, an empty one, looks like this:

Example 2.1. An empty configuration

```xml
<cib crm_feature_set="3.0.7" validate-with="pacemaker-1.2" admin_epoch="1" epoch="0"
    num_updates="0">
  <configuration>
    <crm_config/>
    <nodes/>
    <resources/>
    <constraints/>
  </configuration>
  <status/>
</cib>
```

The empty configuration above contains the major sections that make up a CIB:

- **cib**: The entire CIB is enclosed with a `cib` tag. Certain fundamental settings are defined as attributes of this tag.

- **configuration**: This section — the primary focus of this document — contains traditional configuration information such as what resources the cluster serves and the relationships among them.

  - **crm_config**: cluster-wide configuration options
  
  - **nodes**: the machines that host the cluster
  
  - **resources**: the services run by the cluster

  - **constraints**: indications of how resources should be placed

- **status**: This section contains the history of each resource on each node. Based on this data, the cluster can construct the complete current state of the cluster. The authoritative source for this section is the local executor (pacemaker-execd process) on each cluster node, and the cluster will occasionally repopulate the entire section. For this reason, it is never written to disk, and administrators are advised against modifying it in any way.

In this document, configuration settings will be described as *properties* or *options* based on how they are defined in the CIB:

- Properties are XML attributes of an XML element.

- Options are name-value pairs expressed as `nvpair` child elements of an XML element.
Normally, you will use command-line tools that abstract the XML, so the distinction will be unimportant; both properties and options are cluster settings you can tweak.

### 2.2. CIB Properties

Certain settings are defined by CIB properties (that is, attributes of the `cib` tag) rather than with the rest of the cluster configuration in the `configuration` section.

The reason is simply a matter of parsing. These options are used by the configuration database which is, by design, mostly ignorant of the content it holds. So the decision was made to place them in an easy-to-find location.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>admin_epoch</strong></td>
<td>When a node joins the cluster, the cluster performs a check to see which node has the best configuration. It asks the node with the highest (<em>admin_epoch, epoch, num_updates</em>) tuple to replace the configuration on all the nodes — which makes setting them, and setting them correctly, very important. <em>admin_epoch</em> is never modified by the cluster; you can use this to make the configurations on any inactive nodes obsolete. <em>Never set this value to zero</em>. In such cases, the cluster cannot tell the difference between your configuration and the “empty” one used when nothing is found on disk.</td>
</tr>
<tr>
<td><strong>epoch</strong></td>
<td>The cluster increments this every time the configuration is updated (usually by the administrator).</td>
</tr>
<tr>
<td><strong>num_updates</strong></td>
<td>The cluster increments this every time the configuration or status is updated (usually by the cluster) and resets it to 0 when epoch changes.</td>
</tr>
<tr>
<td><strong>validate-with</strong></td>
<td>Determines the type of XML validation that will be done on the configuration. If set to <em>none</em>, the cluster will not verify that updates conform to the DTD (nor reject ones that don’t). This option can be useful when operating a mixed-version cluster during an upgrade.</td>
</tr>
<tr>
<td><strong>cib-last-written</strong></td>
<td>Indicates when the configuration was last written to disk. Maintained by the cluster; for informational purposes only.</td>
</tr>
<tr>
<td><strong>have-quorum</strong></td>
<td>Indicates if the cluster has quorum. If false, this may mean that the cluster cannot start resources or fence other nodes (see <em>no-quorum-policy</em> below). Maintained by the cluster.</td>
</tr>
<tr>
<td><strong>dc-uuid</strong></td>
<td>Indicates which cluster node is the current leader. Used by the cluster when placing resources and determining the order of some events. Maintained by the cluster.</td>
</tr>
</tbody>
</table>

### 2.3. Cluster Options

Cluster options, as you might expect, control how the cluster behaves when confronted with certain situations.

They are grouped into sets within the `crm_config` section, and, in advanced configurations, there may be more than one set. (This will be described later in the section on *Chapter 8, Rules* where we will show how to have the cluster use different sets of options during working hours than during weekends.) For now, we will describe the simple case where each option is present at most once.
You can obtain an up-to-date list of cluster options, including their default values, by running the `man pacemaker-schedulerd` and `man pacemaker-controld` commands.

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dc-version</td>
<td></td>
<td>Version of Pacemaker on the cluster's DC. Determined automatically by the cluster. Often includes the hash which identifies the exact Git changeset it was built from. Used for diagnostic purposes.</td>
</tr>
<tr>
<td>cluster-infrastructure</td>
<td></td>
<td>The messaging stack on which Pacemaker is currently running. Determined automatically by the cluster. Used for informational and diagnostic purposes.</td>
</tr>
<tr>
<td>no-quorum-policy</td>
<td>stop</td>
<td>What to do when the cluster does not have quorum. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- ignore: continue all resource management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- freeze: continue resource management, but don't recover resources from nodes not in the affected partition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- stop: stop all resources in the affected cluster partition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- suicide: fence all nodes in the affected cluster partition</td>
</tr>
<tr>
<td>batch-limit</td>
<td>0</td>
<td>The maximum number of actions that the cluster may execute in parallel across all nodes. The &quot;correct&quot; value will depend on the speed and load of your network and cluster nodes. If zero, the cluster will impose a dynamically calculated limit only when any node has high load.</td>
</tr>
<tr>
<td>migration-limit</td>
<td>-1</td>
<td>The number of migration jobs that the TE is allowed to execute in parallel on a node. A value of -1 means unlimited.</td>
</tr>
<tr>
<td>symmetric-cluster</td>
<td>TRUE</td>
<td>Can all resources run on any node by default?</td>
</tr>
<tr>
<td>stop-all-resources</td>
<td>FALSE</td>
<td>Should the cluster stop all resources?</td>
</tr>
<tr>
<td>stop-orphan-resources</td>
<td>TRUE</td>
<td>Should deleted resources be stopped? This value takes precedence over <code>is-managed</code> (i.e. even unmanaged resources will be stopped if deleted from the configuration when this value is TRUE).</td>
</tr>
<tr>
<td>stop-orphan-actions</td>
<td>TRUE</td>
<td>Should deleted actions be cancelled?</td>
</tr>
<tr>
<td>start-failure-is-fatal</td>
<td>TRUE</td>
<td>Should a failure to start a resource on a particular node prevent further start attempts on that node? If FALSE, the cluster will decide whether the same node is still eligible based on the resource's current failure count and <code>migration-threshold</code> (see Section 8.2, &quot;Handling Resource Failure&quot;).</td>
</tr>
<tr>
<td>enable-startup-probes</td>
<td>TRUE</td>
<td>Should the cluster check for active resources during startup?</td>
</tr>
</tbody>
</table>
### Chapter 2. Cluster-Wide Configuration

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maintenance-mode</td>
<td>FALSE</td>
<td>Should the cluster refrain from monitoring, starting and stopping resources?</td>
</tr>
<tr>
<td>stonith-enabled</td>
<td>TRUE</td>
<td>Should failed nodes and nodes with resources that can’t be stopped be shot?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If true, or unset, the cluster will refuse to start resources unless one or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>more STONITH resources have been configured. If false, unresponsive nodes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>are immediately assumed to be running no resources, and resource takeover to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>online nodes starts without any further protection (which means data loss if</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the unresponsive node still accesses shared storage, for example). See also</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the requires meta-attribute in Section 4.4, “Resource Options”.</td>
</tr>
<tr>
<td>stonith-action</td>
<td>reboot</td>
<td>Action to send to STONITH device. Allowed values are reboot and off. The</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value poweroff is also allowed, but is only used for legacy devices.</td>
</tr>
<tr>
<td>stonith-timeout</td>
<td>60s</td>
<td>How long to wait for STONITH actions (reboot, on, off) to complete</td>
</tr>
<tr>
<td>stonith-max-attempts</td>
<td>10</td>
<td>How many times fencing can fail for a target before the cluster will no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>longer immediately re-attempt it.</td>
</tr>
<tr>
<td>stonith-watchdog-timeout</td>
<td>0</td>
<td>If nonzero, rely on hardware watchdog self-fencing. If positive, assume</td>
</tr>
<tr>
<td></td>
<td></td>
<td>unseen nodes self-fence within this much time. If negative, and the SBD_</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WATCHDOG_TIMEOUT environment variable is set, use twice that value.</td>
</tr>
<tr>
<td>concurrent-fencing</td>
<td>FALSE</td>
<td>Is the cluster allowed to initiate multiple fence actions concurrently?</td>
</tr>
<tr>
<td>cluster-delay</td>
<td>60s</td>
<td>Estimated maximum round-trip delay over the network (excluding action</td>
</tr>
<tr>
<td></td>
<td></td>
<td>execution). If the TE requires an action to be executed on another node, it</td>
</tr>
<tr>
<td></td>
<td></td>
<td>will consider the action failed if it does not get a response from the other</td>
</tr>
<tr>
<td></td>
<td></td>
<td>node in this time (after considering the action’s own timeout). The “correct”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value will depend on the speed and load of your network and cluster nodes.</td>
</tr>
<tr>
<td>dc-deadtime</td>
<td>20s</td>
<td>How long to wait for a response from other nodes during startup. The</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“correct” value will depend on the speed/load of your network and the type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of switches used.</td>
</tr>
<tr>
<td>cluster-recheck-interval</td>
<td>15min</td>
<td>Polling interval for time-based changes to options, resource parameters</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and constraints.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Cluster is primarily event-driven, but your configuration can have</td>
</tr>
<tr>
<td></td>
<td></td>
<td>elements that take effect based on the time of day. To ensure these changes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>take effect, we can optionally poll the cluster’s status for changes. A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value of 0 disables polling. Positive values are an</td>
</tr>
<tr>
<td>Option</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>interval (in seconds unless other SI units are specified, e.g. 5min).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cluster-ipc-limit</td>
<td>500</td>
<td>The maximum IPC message backlog before one cluster daemon will disconnect another. This is of use in large clusters, for which a good value is the number of resources in the cluster multiplied by the number of nodes. The default of 500 is also the minimum. Raise this if you see &quot;Evicting client&quot; messages for cluster daemon PIDs in the logs.</td>
</tr>
<tr>
<td>pe-error-series-max</td>
<td>-1</td>
<td>The number of PE inputs resulting in ERRORs to save. Used when reporting problems. A value of -1 means unlimited (report all).</td>
</tr>
<tr>
<td>pe-warn-series-max</td>
<td>-1</td>
<td>The number of PE inputs resulting in WARNINGs to save. Used when reporting problems. A value of -1 means unlimited (report all).</td>
</tr>
<tr>
<td>pe-input-series-max</td>
<td>-1</td>
<td>The number of &quot;normal&quot; PE inputs to save. Used when reporting problems. A value of -1 means unlimited (report all).</td>
</tr>
<tr>
<td>placement-strategy</td>
<td>default</td>
<td>How the cluster should allocate resources to nodes (see Chapter 11, Utilization and Placement Strategy [103]). Allowed values are default, utilization, balanced, and minimal.</td>
</tr>
<tr>
<td>node-health-strategy</td>
<td>none</td>
<td>How the cluster should react to node health attributes (see Section 8.4, “Tracking Node Health”). Allowed values are none, migrate-on-red, only-green, progressive, and custom.</td>
</tr>
<tr>
<td>node-health-base</td>
<td>0</td>
<td>The base health score assigned to a node. Only used when node-health-strategy is progressive.</td>
</tr>
<tr>
<td>node-health-green</td>
<td>0</td>
<td>The score to use for a node health attribute whose value is green. Only used when node-health-strategy is progressive or custom.</td>
</tr>
<tr>
<td>node-health-yellow</td>
<td>0</td>
<td>The score to use for a node health attribute whose value is yellow. Only used when node-health-strategy is progressive or custom.</td>
</tr>
<tr>
<td>node-health-red</td>
<td>0</td>
<td>The score to use for a node health attribute whose value is red. Only used when node-health-strategy is progressive or custom.</td>
</tr>
<tr>
<td>remove-after-stop</td>
<td>FALSE</td>
<td><strong>Advanced Use Only:</strong> Should the cluster remove resources from the LRM after they are stopped? Values other than the default are, at best, poorly tested and potentially dangerous.</td>
</tr>
<tr>
<td>startup-fencing</td>
<td>TRUE</td>
<td><strong>Advanced Use Only:</strong> Should the cluster shoot unseen nodes? Not using the default is very unsafe!</td>
</tr>
<tr>
<td>election-timeout</td>
<td>2min</td>
<td><strong>Advanced Use Only:</strong> If you need to adjust this value, it probably indicates the presence of a bug.</td>
</tr>
<tr>
<td>shutdown-escalation</td>
<td>20min</td>
<td><strong>Advanced Use Only:</strong> If you need to adjust this value, it probably indicates the presence of a bug.</td>
</tr>
</tbody>
</table>
Chapter 2. Cluster-Wide Configuration

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>join-integration-timeout</code></td>
<td>3min</td>
<td>Advanced Use Only: If you need to adjust this value, it probably indicates the presence of a bug.</td>
</tr>
<tr>
<td><code>join-finalization-timeout</code></td>
<td>30min</td>
<td>Advanced Use Only: If you need to adjust this value, it probably indicates the presence of a bug.</td>
</tr>
<tr>
<td><code>transition-delay</code></td>
<td>0s</td>
<td>Advanced Use Only: Delay cluster recovery for the configured interval to allow for additional/related events to occur. Useful if your configuration is sensitive to the order in which ping updates arrive. Enabling this option will slow down cluster recovery under all conditions.</td>
</tr>
</tbody>
</table>
Cluster Nodes

Table of Contents

3.1. Defining a Cluster Node ...................................................................................................... 13
3.2. Where Pacemaker Gets the Node Name ............................................................................. 13
3.3. Node Attributes ................................................................................................................... 13
    3.3.1. Setting and querying node attributes ......................................................................... 14
    3.3.2. Special node attributes ............................................................................................. 14

3.1. Defining a Cluster Node

Each node in the cluster will have an entry in the nodes section containing its UUID, uname, and type.

Example 3.1. Example Corosync cluster node entry

```
<node id="101" uname="pcmk-1"/>
```

In normal circumstances, the admin should let the cluster populate this information automatically from the communications and membership data.

3.2. Where Pacemaker Gets the Node Name

Traditionally, Pacemaker required nodes to be referred to by the value returned by `uname -n`. This can be problematic for services that require the `uname -n` to be a specific value (e.g. for a licence file).

This requirement has been relaxed for clusters using Corosync 2.0 or later. The name Pacemaker uses is:

1. The value stored in `corosync.conf` under `ring0_addr` in the `nodelist`, if it does not contain an IP address; otherwise
2. The value stored in `corosync.conf` under `name` in the `nodelist`; otherwise
3. The value of `uname -n`

Pacemaker provides the `crm_node -n` command which displays the name used by a running cluster.

If a Corosync `nodelist` is used, `crm_node --name-for-id number` is also available to display the name used by the node with the corosync `nodeid` of `number`, for example: `crm_node --name-for-id 2`.

3.3. Node Attributes

Pacemaker allows node-specific values to be specified using `node attributes`. A node attribute has a name, and may have a distinct value for each node.

While certain node attributes have specific meanings to the cluster, they are mainly intended to allow administrators and resource agents to track any information desired.
Chapter 3. Cluster Nodes

For example, an administrator might choose to define node attributes for how much RAM and disk space each node has, which OS each uses, or which server room rack each node is in.

Users can configure rules that use node attributes to affect where resources are placed.

3.3.1. Setting and querying node attributes
Node attributes can be set and queried using the crm_attribute and attrd_updater commands, so that the user does not have to deal with XML configuration directly.

Here is an example of what XML configuration would be generated if an administrator ran this command:

Example 3.2. Result of using crm_attribute to specify which kernel pcmk-1 is running

```
# crm_attribute --type nodes --node pcmk-1 --name kernel --update $(uname -r)

<node id="1" uname="pcmk-1">
    <instance_attributes id="nodes-1-attributes">
        <nvpair id="nodes-1-kernel" name="kernel" value="3.10.0-862.14.4.el7.x86_64"/>
    </instance_attributes>
</node>
```

To read back the value that was just set:

```
# crm_attribute --type nodes --node pcmk-1 --name kernel --query
scope=nodes  name=kernel value=3.10.0-862.14.4.el7.x86_64
```

By specifying --type nodes the admin tells the cluster that this attribute is persistent across reboots. There are also transient attributes which are kept in the status section and are "forgotten" whenever the node leaves the cluster. Administrators can use this section by specifying --type status.

3.3.2. Special node attributes
Certain node attributes have special meaning to the cluster.

Node attribute names beginning with # are considered reserved for these special attributes. Some special attributes do not start with #, for historical reasons.

Certain special attributes are set automatically by the cluster, should never be modified directly, and can be used only within rules; these are listed under Section 7.2, “Node Attribute Expressions” [54].

For true/false values, the cluster considers a value of "1", "y", "yes", "on", or "true" (case-insensitively) to be true, "0", "n", "no", "off", "false", or unset to be false, and anything else to be an error.

Table 3.1. Node attributes with special significance

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fail-count-*</td>
<td>Attributes whose names start with fail-count- are managed by the cluster to track how many times particular resource operations have failed on this node. These should be queried and cleared via the crm_failcount or crm_resource --cleanup commands rather than directly.</td>
</tr>
<tr>
<td>last-failure-*</td>
<td>Attributes whose names start with last-failure- are managed by the cluster to track when particular resource operations have</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>most recently failed on this node. These should be cleared via the <code>crm_failcount</code> or <code>crm_resource --cleanup</code> commands rather than directly.</td>
</tr>
<tr>
<td>maintenance</td>
<td>Similar to the <code>maintenance-mode</code> cluster option, but for a single node. If true, resources will not be started or stopped on the node, resources and individual clone instances running on the node will become unmanaged, and any recurring operations for those will be cancelled.</td>
</tr>
<tr>
<td>probe_complete</td>
<td>This is managed by the cluster to detect when nodes need to be reprobed, and should never be used directly.</td>
</tr>
<tr>
<td>resource-discovery-enabled</td>
<td>If the node is a remote node, fencing is enabled, and this attribute is explicitly set to false (unset means true in this case), resource discovery (probes) will not be done on this node. This is highly discouraged; the <code>resource-discovery</code> location constraint property is preferred for this purpose.</td>
</tr>
<tr>
<td>shutdown</td>
<td>This is managed by the cluster to orchestrate the shutdown of a node, and should never be used directly.</td>
</tr>
<tr>
<td>site-name</td>
<td>If set, this will be used as the value of the <code>#site-name</code> node attribute used in rules. (If not set, the value of the <code>cluster-name</code> cluster option will be used as <code>#site-name</code> instead.)</td>
</tr>
<tr>
<td>standby</td>
<td>If true, the node is in standby mode. This is typically set and queried via the <code>crm_standby</code> command rather than directly.</td>
</tr>
<tr>
<td>terminate</td>
<td>If the value is true or begins with any nonzero number, the node will be fenced. This is typically set by tools rather than directly.</td>
</tr>
<tr>
<td>#digests-*</td>
<td>Attributes whose names start with <code>#digests-</code> are managed by the cluster to detect when unfencing needs to be redone, and should never be used directly.</td>
</tr>
<tr>
<td>#node-unfenced</td>
<td>When the node was last unfenced (as seconds since the epoch). This is managed by the cluster and should never be used directly.</td>
</tr>
</tbody>
</table>

**Warning**

Restarting pacemaker on a node that is in single-node maintenance mode will likely lead to undesirable effects. If `maintenance` is set as a transient attribute, it will be erased when pacemaker is stopped, which will immediately take the node out of maintenance mode and likely get it fenced. Even if permanent, if pacemaker is restarted, any resources active on the node will have their local history erased when the node rejoins, so the cluster will no longer consider them running on the node and thus will consider them managed again, leading them to be started elsewhere. This behavior might be improved in a future release.
Chapter 4.

Cluster Resources

Table of Contents

4.1. What is a Cluster Resource? ................................................................. 17
4.2. Resource Classes .............................................................................. 17
  4.2.1. Open Cluster Framework ............................................................. 18
  4.2.2. Linux Standard Base ................................................................. 18
  4.2.3. Systemd .................................................................................. 19
  4.2.4. Upstart .......................................................... .................................. 19
  4.2.5. System Services ........................................................................ 20
  4.2.6. STONITH ................................................................................ 20
  4.2.7. Nagios Plugins ......................................................................... 20
4.3. Resource Properties ........................................................................... 21
4.4. Resource Options ............................................................................... 21
  4.4.1. Resource Meta-Attributes ......................................................... 21
  4.4.2. Setting Global Defaults for Resource Meta-Attributes .............. 24
  4.4.3. Resource Instance Attributes .................................................. 24
4.5. Resource Operations .......................................................................... 26
  4.5.1. Monitoring Resources for Failure ............................................. 27
  4.5.2. Monitoring Resources When Administration is Disabled ............ 28
  4.5.3. Setting Global Defaults for Operations ...................................... 29
  4.5.4. When Implicit Operations Take a Long Time ............................. 29
  4.5.5. Multiple Monitor Operations ................................................... 29
  4.5.6. Disabling a Monitor Operation .................................................. 30

4.1. What is a Cluster Resource?

A resource is a service made highly available by a cluster. The simplest type of resource, a primitive resource, is described in this section. More complex forms, such as groups and clones, are described in later sections.

Every primitive resource has a resource agent. A resource agent is an external program that abstracts the service it provides and present a consistent view to the cluster.

This allows the cluster to be agnostic about the resources it manages. The cluster doesn’t need to understand how the resource works because it relies on the resource agent to do the right thing when given a start, stop or monitor command. For this reason, it is crucial that resource agents are well-tested.

Typically, resource agents come in the form of shell scripts. However, they can be written using any technology (such as C, Python or Perl) that the author is comfortable with.

4.2. Resource Classes

Pacemaker supports several classes of agents:

- OCF
- LSB
Chapter 4. Cluster Resources

- Upstart
- Systemd
- Service
- Fencing
- Nagios Plugins

4.2.1. Open Cluster Framework

The OCF standard is basically an extension of the Linux Standard Base conventions for init scripts to:

- support parameters,
- make them self-describing, and
- make them extensible

OCF specs have strict definitions of the exit codes that actions must return.

The cluster follows these specifications exactly, and giving the wrong exit code will cause the cluster to behave in ways you will likely find puzzling and annoying. In particular, the cluster needs to distinguish a completely stopped resource from one which is in some erroneous and indeterminate state.

Parameters are passed to the resource agent as environment variables, with the special prefix `OCF_RESKEY_`. So, a parameter which the user thinks of as `ip` will be passed to the resource agent as `OCF_RESKEY_ip`. The number and purpose of the parameters is left to the resource agent; however, the resource agent should use the `meta-data` command to advertise any that it supports.

The OCF class is the most preferred as it is an industry standard, highly flexible (allowing parameters to be passed to agents in a non-positional manner) and self-describing.

For more information, see the reference and the Resource Agents section of Pacemaker Administration.

4.2.2. Linux Standard Base

LSB resource agents are more commonly known as init scripts. If a full path is not given, they are assumed to be located in `/etc/init.d`.

Commonly, they are provided by the OS distribution. In order to be used with a Pacemaker cluster, they must conform to the LSB specification.

---

1. See [http://www.opencf.org/cgi-bin/viewcvs.cgi/specs/ra/resource-agent-api.txt?rev=HEAD](http://www.opencf.org/cgi-bin/viewcvs.cgi/specs/ra/resource-agent-api.txt?rev=HEAD) — at least as it relates to resource agents. The Pacemaker implementation has been somewhat extended from the OCF specs, but none of those changes are incompatible with the original OCF specification.

2. The resource-agents source code includes the `ocf-tester` script, which can be useful in this regard.


4. See [http://refspecs.linux-foundation.org/LSB_3.0.0/LSB-Core-generic/LSB-Core-generic/iniscrptact.html](http://refspecs.linux-foundation.org/LSB_3.0.0/LSB-Core-generic/LSB-Core-generic/iniscrptact.html) for the LSB Spec as it relates to init scripts.
Warning

Many distributions or particular software packages claim LSB compliance but ship with broken init scripts. For details on how to check whether your init script is LSB-compatible, see the Resource Agents section of Pacemaker Administration. Common problematic violations of the LSB standard include:

- Not implementing the status operation at all
- Not observing the correct exit status codes for start/stop/status actions
- Starting a started resource returns an error
- Stopping a stopped resource returns an error

Important

Remember to make sure the computer is not configured to start any services at boot time — that should be controlled by the cluster.

4.2.3. Systemd

Some newer distributions have replaced the old “SysV”® style of initialization daemons and scripts with an alternative called Systemd.

Pacemaker is able to manage these services if they are present.

Instead of init scripts, systemd has unit files. Generally, the services (unit files) are provided by the OS distribution, but there are online guides for converting from init scripts. 7

Important

Remember to make sure the computer is not configured to start any services at boot time — that should be controlled by the cluster.

4.2.4. Upstart

5 http://en.wikipedia.org/wiki/Init#SysV-style
6 http://www.freedesktop.org/wiki/Software/systemd
7 For example, http://0pointer.de/blog/projects/systemd-for-admins-3.html
Some newer distributions have replaced the old "SysV" style of initialization daemons (and scripts) with an alternative called Upstart. Pacemaker is able to manage these services if they are present. Instead of init scripts, upstart has jobs. Generally, the services (jobs) are provided by the OS distribution.

**Important**

Remember to make sure the computer is not configured to start any services at boot time — that should be controlled by the cluster.

**4.2.5. System Services**

Since there are various types of system services (systemd, upstart, and lsb), Pacemaker supports a special service 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.

In order, Pacemaker will try to find the named service as:
1. an LSB init script
2. a Systemd unit file
3. an Upstart job

**4.2.6. STONITH**

The STONITH class is used exclusively for fencing-related resources. This is discussed later in Chapter 13, STONITH.

**4.2.7. Nagios Plugins**

Nagios Plugins allow us to monitor services on remote hosts. Pacemaker is able to do remote monitoring with the plugins if they are present.

A common use case is to configure them as resources belonging to a resource container (usually a virtual machine), and the container will be restarted if any of them has failed. Another use is to configure them as ordinary resources to be used for monitoring hosts or services via the network.

---

10 The project has two independent forks, hosted at [https://www.nagios-plugins.org/](https://www.nagios-plugins.org/) and [https://www.monitoring-plugins.org/](https://www.monitoring-plugins.org/). Output from both projects’ plugins is similar, so plugins from either project can be used with pacemaker.
The supported parameters are same as the long options of the plugin.

### 4.3. Resource Properties

These values tell the cluster which resource agent to use for the resource, where to find that resource agent and what standards it conforms to.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Your name for the resource</td>
</tr>
<tr>
<td>class</td>
<td>The standard the resource agent conforms to. Allowed values: lsb, nagios, ocf, service, stonith, systemd, upstart</td>
</tr>
<tr>
<td>type</td>
<td>The name of the Resource Agent you wish to use. E.g. IPaddr or Filesystem</td>
</tr>
<tr>
<td>provider</td>
<td>The OCF spec allows multiple vendors to supply the same resource agent. To use the OCF resource agents supplied by the Heartbeat project, you would specify heartbeat here.</td>
</tr>
</tbody>
</table>

The XML definition of a resource can be queried with the `crm_resource` tool. For example:

```
# crm_resource --resource Email --query-xml
```

might produce:

```
Example 4.1. A system resource definition

<primitive id="Email" class="service" type="exim"/>
```

**Note**

One of the main drawbacks to system services (LSB, systemd or Upstart) resources is that they do not allow any parameters!

```
Example 4.2. An OCF resource definition

<primitive id="Public-IP" class="ocf" type="IPaddr" provider="heartbeat">  
    <instance_attributes id="Public-IP-params">  
        <nvpair id="Public-IP-ip" name="ip" value="192.0.2.2"/>
    </instance_attributes>
</primitive>
```

### 4.4. Resource Options

Resources have two types of options: meta-attributes and instance attributes. Meta-attributes apply to any type of resource, while instance attributes are specific to each resource agent.

#### 4.4.1. Resource Meta-Attributes

Meta-attributes are used by the cluster to decide how a resource should behave and can be easily set using the `--meta` option of the `crm_resource` command.
### Table 4.2. Meta-attributes of a Primitive Resource

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>priority</strong></td>
<td>0</td>
<td>If not all resources can be active, the cluster will stop lower priority resources in order to keep higher priority ones active.</td>
</tr>
<tr>
<td><strong>target-role</strong></td>
<td>Started</td>
<td>What state should the cluster attempt to keep this resource in? Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>Stopped</strong>: Force the resource to be stopped</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>Started</strong>: Allow the resource to be started (and in the case of <em>promotable clone resources</em>, promoted to master if appropriate)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>Slave</strong>: Allow the resource to be started, but only in Slave mode if the resource is <em>promotable</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>Master</strong>: Equivalent to <strong>Started</strong></td>
</tr>
<tr>
<td><strong>is-managed</strong></td>
<td>TRUE</td>
<td>Is the cluster allowed to start and stop the resource? Allowed values: <em>true, false</em></td>
</tr>
<tr>
<td><strong>resource-stickiness</strong></td>
<td>1 for individual clone instances, 0 for all other resources</td>
<td>A score that will be added to the current node when a resource is already active. This allows running resources to stay where they are, even if they would be placed elsewhere if they were being started from a stopped state.</td>
</tr>
<tr>
<td><strong>requires</strong></td>
<td>quorum for resources with a class of <em>stonith</em>, otherwise unfencing if unfencing is active in the cluster, otherwise fencing if <em>stonith-enabled</em> is true, otherwise quorum</td>
<td>Conditions under which the resource can be started Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>nothing</strong>: can always be started</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>quorum</strong>: The cluster can only start this resource if a majority of the configured nodes are active</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>fencing</strong>: The cluster can only start this resource if a majority of the configured nodes are active and any failed or unknown nodes have been fenced</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>unfencing</strong>: The cluster can only start this resource if a majority of the configured nodes are active and any failed or unknown nodes have been fenced and only on nodes that have been unfenced</td>
</tr>
<tr>
<td><strong>migration-threshold</strong></td>
<td>INFINITY</td>
<td>How many failures may occur for this resource on a node, before this node is marked ineligible to host this resource. A value of 0 indicates that this feature is disabled (the node will never be marked ineligible); by constrast, the cluster treats INFINITY (the default) as a very large but finite number. This option has an effect only if the failed operation specifies <strong>on-fail</strong> as <strong>restart</strong> (the default),</td>
</tr>
<tr>
<td>Field</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>failure-timeout</td>
<td>0</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. A value of 0 indicates that this feature is disabled. As with any time-based actions, this is not guaranteed to be checked more frequently than the value of <code>cluster-recheck-interval</code> (see Section 2.3, “Cluster Options”).</td>
</tr>
<tr>
<td>multiple-active</td>
<td>stop_start</td>
<td>What should the cluster do if it ever finds the resource active on more than one node? Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• block: mark the resource as unmanaged</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• stop_only: stop all active instances and leave them that way</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• stop_start: stop all active instances and start the resource in one location only</td>
</tr>
<tr>
<td>allow-migrate</td>
<td>TRUE for ocf:pacemaker:remote resources, FALSE otherwise</td>
<td>Whether the cluster should try to &quot;live migrate&quot; this resource when it needs to be moved (see Section 8.3.3, “Migrating Resources”)</td>
</tr>
<tr>
<td>container-attribute-target</td>
<td></td>
<td>Specific to bundle resources; see Section 9.3.7, “Bundle Node Attributes”</td>
</tr>
<tr>
<td>remote-node</td>
<td></td>
<td>The name of the Pacemaker Remote guest node this resource is associated with, if any. If specified, this both enables the resource as a guest node and defines the unique name used to identify the guest node. The guest must be configured to run the Pacemaker Remote daemon when it is started. WARNING: This value cannot overlap with any resource or node IDs.</td>
</tr>
<tr>
<td>remote-port</td>
<td>3121</td>
<td>If <code>remote-node</code> is specified, the port on the guest used for its Pacemaker Remote connection. The Pacemaker Remote daemon on the guest must be configured to listen on this port.</td>
</tr>
<tr>
<td>remote-addr</td>
<td>value of <code>remote-node</code></td>
<td>If <code>remote-node</code> is specified, the IP address or hostname used to connect to the guest via Pacemaker Remote. The Pacemaker Remote daemon on the guest must be configured to accept connections on this address.</td>
</tr>
</tbody>
</table>
Field | Default | Description
--- | --- | ---
remote-connect-timeout | 60s | If remote-node is specified, how long before a pending guest connection will time out.

As an example of setting resource options, if you performed the following commands on an LSB Email resource:

```
# crm_resource --meta --resource Email --set-parameter priority --parameter-value 100
# crm_resource -m -r Email -p multiple-active -v block
```

the resulting resource definition might be:

```
Example 4.3. An LSB resource with cluster options

<primitive id="Email" class="lsb" type="exim">
  <meta_attributes id="Email-meta_attributes">
    <nvpair id="Email-meta_attributes-priority" name="priority" value="100"/>
    <nvpair id="Email-meta_attributes-multiple-active" name="multiple-active" value="block"/>
  </meta_attributes>
</primitive>
```

In addition to the cluster-defined meta-attributes described above, you may also configure arbitrary meta-attributess of your own choosing. Most commonly, this would be done for use in rules. For example, an IT department might define a custom meta-attribute to indicate which company department each resource is intended for. To reduce the chance of name collisions with cluster-defined meta-attributes added in the future, it is recommended to use a unique, organization-specific prefix for such attributes.

### 4.4.2. Setting Global Defaults for Resource Meta-Attributes

To set a default value for a resource option, add it to the rsc_defaults section with `crm_attribute`. For example,

```
# crm_attribute --type rsc_defaults --name is-managed --update false
```

would prevent the cluster from starting or stopping any of the resources in the configuration (unless of course the individual resources were specifically enabled by having their is-managed set to true).

### 4.4.3. Resource Instance Attributes

The resource agents of some resource classes (lsb, systemd and upstart not among them) 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. For example,

```
# crm_resource --resource Public-IP --set-parameter ip --parameter-value 192.0.2.2
```

would create an entry in the resource like this:

```
Example 4.4. An example OCF resource with instance attributes

<primitive id="Public-IP" class="ocf" type="IPaddr" provider="heartbeat">
```

24
For an OCF resource, the result would be an environment variable called **OCF_RESKEY_ip** with a value of **192.0.2.2**.

The list of instance attributes supported by an OCF resource agent can be found by calling the resource agent with the **meta-data** command. The output contains an XML description of all the supported attributes, their purpose and default values.

Example 4.5. Displaying the metadata for the Dummy resource agent template

```bash
# export OCF_ROOT=/usr/lib/ocf
# $OCF_ROOT/resource.d/pacemaker/Dummy meta-data

<?xml version="1.0"?>
<!DOCTYPE resource-agent SYSTEM "ra-api-1.dtd">
<resource-agent name="Dummy" version="1.0">
  <version>1.0</version>

  <longdesc>
  This is a Dummy Resource Agent. It does absolutely nothing except keep track of whether its running or not.
  Its purpose in life is for testing and to serve as a template for RA writers.
  
  NB: Please pay attention to the timeouts specified in the actions section below. They should be meaningful for the kind of resource the agent manages. They should be the minimum advised timeouts, but they shouldn't/cannot cover _all_ possible resource instances. So, try to be neither overly generous nor too stingy, but moderate. The minimum timeouts should never be below 10 seconds.
  </longdesc>
  <shortdesc>Example stateless resource agent</shortdesc>

  <parameters>
  <parameter name="state" unique="1">
    <longdesc>
    Location to store the resource state in.
    </longdesc>
    <shortdesc>State file</shortdesc>
    <content type="string" default="/var/run/Dummy-default.state" />
  </parameter>

  <parameter name="fake" unique="0">
    <longdesc>
    Fake attribute that can be changed to cause a reload
    </longdesc>
    <shortdesc>Fake attribute that can be changed to cause a reload</shortdesc>
    <content type="string" default="dummy" />
  </parameter>

  <parameter name="op_sleep" unique="1">
    <longdesc>
    Number of seconds to sleep during operations. This can be used to test how the cluster reacts to operation timeouts.
    </longdesc>
    <shortdesc>Operation sleep duration in seconds.</shortdesc>
    <content type="string" default="0" />
  </parameter>
  
  </parameters>

</resource-agent>
Chapter 4. Cluster Resources

4.5. Resource Operations

Operations are actions the cluster can perform on a resource by calling the resource agent. Resource agents must support certain common operations such as start, stop, and monitor, and may implement any others.

Operations may be explicitly configured for two purposes: to override defaults for options (such as timeout) that the cluster will use whenever it initiates the operation, and to run an operation on a recurring basis (for example, to monitor the resource for failure).

Example 4.6. An OCF resource with a non-default start timeout

Pacemaker identifies operations by a combination of name and interval, so this combination must be unique for each resource. That is, you should not configure two operations for the same resource with the same name and interval.

Table 4.3. Properties of an Operation

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>A unique name for the operation.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>The action to perform. This can be any action supported by the agent; common values include monitor, start, and stop.</td>
<td></td>
</tr>
<tr>
<td>interval</td>
<td>0</td>
<td>How frequently (in seconds) to perform the operation. A value of 0 means &quot;when needed&quot;. A positive value defines a recurring action, which is typically used with monitor.</td>
</tr>
<tr>
<td>timeout</td>
<td>How long to wait before declaring the action has failed</td>
<td></td>
</tr>
<tr>
<td>on-fail</td>
<td>restart (except for stop operations, which default to fence when)</td>
<td>The action to take if this action ever fails. Allowed values:</td>
</tr>
<tr>
<td>Field</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Field</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>enabled</td>
<td>TRUE</td>
<td>If false, ignore this operation definition. This is typically used to pause a particular recurring monitor operation; for instance, it can complement the respective resource being unmanaged (is-managed=false), as this alone will not block any configured monitoring. Disabling the operation does not suppress all actions of the given type. Allowed values: true, false.</td>
</tr>
<tr>
<td>record-pending</td>
<td>TRUE</td>
<td>If true, the intention to perform the operation is recorded so that GUIs and CLI tools can indicate that an operation is in progress. This is best set as an operation default (see Section 4.5.3, “Setting Global Defaults for Operations”). Allowed values: true, false.</td>
</tr>
<tr>
<td>role</td>
<td></td>
<td>Run the operation only on node(s) that the cluster thinks should be in the specified role. This only makes sense for recurring monitor operations. Allowed (case-sensitive) values: Stopped, Started, and in the case of promotable clone resources, Slave and Master.</td>
</tr>
</tbody>
</table>

4.5.1. Monitoring Resources for Failure

When Pacemaker first starts a resource, it runs one-time monitor operations (referred to as probes) to ensure the resource is running where it’s supposed to be, and not running where it’s not supposed to be. (This behavior can be affected by the resource-discovery location constraint property.)

Other than those initial probes, Pacemaker will not (by default) check that the resource continues to stay healthy. You must configure monitor operations explicitly to perform these checks.

Currently, anyway. Automatic monitoring operations may be added in a future version of Pacemaker.
Example 4.7. An OCF resource with a recurring health check

```xml
<primitive id="Public-IP" class="ocf" type="IPaddr" provider="heartbeat">
  <operations>
    <op id="Public-IP-start" name="start" timeout="60s"/>
    <op id="Public-IP-monitor" name="monitor" interval="60s"/>
  </operations>
  <instance_attributes id="params-public-ip">
    <nvpair id="public-ip-addr" name="ip" value="192.0.2.2"/>
  </instance_attributes>
</primitive>
```

By default, a monitor operation will ensure that the resource is running where it is supposed to. The target-role property can be used for further checking.

For example, if a resource has one monitor operation with `interval=10 role=Started` and a second monitor operation with `interval=11 role=Stopped`, the cluster will run the first monitor on any nodes it thinks should be running the resource, and the second monitor on any nodes that it thinks should not be running the resource (for the truly paranoid, who want to know when an administrator manually starts a service by mistake).

**Note**

Currently, monitors with `role=Stopped` are not implemented for clone resources.

### 4.5.2. Monitoring Resources When Administration is Disabled

Recurring monitor operations behave differently under various administrative settings:

- **When a resource is unmanaged (by setting `is-managed=false`):** No monitors will be stopped.

  If the unmanaged resource is stopped on a node where the cluster thinks it should be running, the cluster will detect and report that it is not, but it will not consider the monitor failed, and will not try to start the resource until it is managed again.

  Starting the unmanaged resource on a different node is strongly discouraged and will at least cause the cluster to consider the resource failed, and may require the resource's target-role to be set to Stopped then Started to be recovered.

- **When a node is put into standby:** All resources will be moved away from the node, and all monitor operations will be stopped on the node, except those specifying role as Stopped (which will be newly initiated if appropriate).

- **When the cluster is put into maintenance mode:** All resources will be marked as unmanaged. All monitor operations will be stopped, except those specifying role as Stopped (which will be newly initiated if appropriate). As with single unmanaged resources, starting a resource on a node other than where the cluster expects it to be will cause problems.
4.5.3. Setting Global Defaults for Operations

You can change the global default values for operation properties in a given cluster. These are defined in an op_defaults section of the CIB’s configuration section, and can be set with crm_attribute. For example,

```
# crm_attribute --type op_defaults --name timeout --update 20s
```

would default each operation’s timeout to 20 seconds. If an operation’s definition also includes a value for timeout, then that value would be used for that operation instead.

4.5.4. When Implicit Operations Take a Long Time

The cluster will always perform a number of implicit operations: start, stop and a non-recurring monitor operation used at startup to check whether the resource is already active. If one of these is taking too long, then you can create an entry for them and specify a longer timeout.

Example 4.8. An OCF resource with custom timeouts for its implicit actions

```
<primitive id="Public-IP" class="ocf" type="IPaddr" provider="heartbeat">
    <operations>
        <op id="public-ip-startup" name="monitor" interval="0" timeout="90s"/>
        <op id="public-ip-start" name="start" interval="0" timeout="180s"/>
        <op id="public-ip-stop" name="stop" interval="0" timeout="15min"/>
    </operations>
    <instance_attributes id="params-public-ip">
        <nvpair id="public-ip-addr" name="ip" value="192.0.2.2"/>
    </instance_attributes>
</primitive>
```

4.5.5. Multiple Monitor Operations

Provided no two operations (for a single resource) have the same name and interval, you can have as many monitor operations as you like. In this way, you can do a superficial health check every minute and progressively more intense ones at higher intervals.

To tell the resource agent what kind of check to perform, you need to provide each monitor with a different value for a common parameter. The OCF standard creates a special parameter called OCF_CHECK_LEVEL for this purpose and dictates that it is "made available to the resource agent without the normal OCF_RESKEY prefix".

Whatever name you choose, you can specify it by adding an instance_attributes block to the op tag. It is up to each resource agent to look for the parameter and decide how to use it.

Example 4.9. An OCF resource with two recurring health checks, performing different levels of checks specified via OCF_CHECK_LEVEL.

```
<primitive id="Public-IP" class="ocf" type="IPaddr" provider="heartbeat">
    <operations>
        <op id="public-ip-health-60" name="monitor" interval="60">
            <instance_attributes id="params-public-ip-depth-60">
                <nvpair id="public-ip-depth-60" name="OCF_CHECK_LEVEL" value="10"/>
            </instance_attributes>
        </op>
        <op id="public-ip-health-300" name="monitor" interval="300">
            <instance_attributes id="params-public-ip-depth-300">
                <nvpair id="public-ip-depth-300" name="OCF_CHECK_LEVEL" value="20"/>
            </instance_attributes>
        </op>
    </operations>
</primitive>
```
4.5.6. Disabling a Monitor Operation

The easiest way to stop a recurring monitor is to just delete it. However, there can be times when you only want to disable it temporarily. In such cases, simply add **enabled=false** to the operation's definition.

**Example 4.10. Example of an OCF resource with a disabled health check**

```xml
<primitive id="Public-IP" class="ocf" type="IPaddr" provider="heartbeat">
    <operations>
        <op id="public-ip-check" name="monitor" interval="60s" enabled="false"/>
    </operations>
    <instance_attributes id="params-public-ip">
        <nvpair id="public-ip-addr" name="ip" value="192.0.2.2"/>
    </instance_attributes>
</primitive>
```

This can be achieved from the command line by executing:

```
# cibadmin --modify --xml-text '<op id="public-ip-check" enabled="false"/>'
```

Once you've done whatever you needed to do, you can then re-enable it with

```
# cibadmin --modify --xml-text '<op id="public-ip-check" enabled="true"/>'
```
# Resource Constraints

## Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1. Scores</td>
<td>31</td>
</tr>
<tr>
<td>5.1.1. Infinity Math</td>
<td>31</td>
</tr>
<tr>
<td>5.2. Deciding Which Nodes a Resource Can Run On</td>
<td>32</td>
</tr>
<tr>
<td>5.2.1. Location Properties</td>
<td>32</td>
</tr>
<tr>
<td>5.2.2. Asymmetrical “Opt-In” Clusters</td>
<td>33</td>
</tr>
<tr>
<td>5.2.3. Symmetrical “Opt-Out” Clusters</td>
<td>34</td>
</tr>
<tr>
<td>5.2.4. What if Two Nodes Have the Same Score</td>
<td>34</td>
</tr>
<tr>
<td>5.3. Specifying the Order in which Resources Should Start/Stop</td>
<td>34</td>
</tr>
<tr>
<td>5.3.1. Ordering Properties</td>
<td>35</td>
</tr>
<tr>
<td>5.3.2. Optional and mandatory ordering</td>
<td>36</td>
</tr>
<tr>
<td>5.4. Placing Resources Relative to other Resources</td>
<td>36</td>
</tr>
<tr>
<td>5.4.1. Colocation Properties</td>
<td>37</td>
</tr>
<tr>
<td>5.4.2. Mandatory Placement</td>
<td>37</td>
</tr>
<tr>
<td>5.4.3. Advisory Placement</td>
<td>38</td>
</tr>
<tr>
<td>5.4.4. Colocation by Node Attribute</td>
<td>38</td>
</tr>
<tr>
<td>5.5. Resource Sets</td>
<td>38</td>
</tr>
<tr>
<td>5.6. Ordering Sets of Resources</td>
<td>39</td>
</tr>
<tr>
<td>5.6.1. Ordered Set</td>
<td>39</td>
</tr>
<tr>
<td>5.6.2. Ordering Multiple Sets</td>
<td>40</td>
</tr>
<tr>
<td>5.6.3. Resource Set OR Logic</td>
<td>41</td>
</tr>
<tr>
<td>5.7. Colocating Sets of Resources</td>
<td>42</td>
</tr>
</tbody>
</table>

## 5.1. Scores

Scores of all kinds are integral to how the cluster works. Practically everything from moving a resource to deciding which resource to stop in a degraded cluster is achieved by manipulating scores in some way.

Scores are calculated per resource and node. Any node with a negative score for a resource can’t run that resource. The cluster places a resource on the node with the highest score for it.

### 5.1.1. Infinity Math

Pacemaker implements INFINITY (or equivalently, +INFINITY) internally as a score of 1,000,000. Addition and subtraction with it follow these three basic rules:

- Any value + INFINITY = INFINITY
- Any value - INFINITY = -INFINITY
- INFINITY - INFINITY = -INFINITY
Chapter 5. Resource Constraints

Note

What if you want to use a score higher than 1,000,000? Typically this possibility arises when someone wants to base the score on some external metric that might go above 1,000,000.

The short answer is you can't.

The long answer is it is sometimes possible work around this limitation creatively. You may be able to set the score to some computed value based on the external metric rather than use the metric directly. For nodes, you can store the metric as a node attribute, and query the attribute when computing the score (possibly as part of a custom resource agent).

5.2. Deciding Which Nodes a Resource Can Run On

Location constraints tell the cluster which nodes a resource can run on.

There are two alternative strategies. One way is to say that, by default, resources can run anywhere, and then the location constraints specify nodes that are not allowed (an opt-out cluster). The other way is to start with nothing able to run anywhere, and use location constraints to selectively enable allowed nodes (an opt-in cluster).

Whether you should choose opt-in or opt-out depends on your personal preference and the make-up of your cluster. If most of your resources can run on most of the nodes, then an opt-out arrangement is likely to result in a simpler configuration. On the other-hand, if most resources can only run on a small subset of nodes, an opt-in configuration might be simpler.

5.2.1. Location Properties

Table 5.1. Properties of a rsc_location Constraint

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td></td>
<td>A unique name for the constraint</td>
</tr>
<tr>
<td>rsc</td>
<td></td>
<td>The name of the resource to which this constraint applies</td>
</tr>
<tr>
<td>rsc-pattern</td>
<td></td>
<td>An extended regular expression (as defined in POSIX(^1)) matching the names of resources to which this constraint applies, if rsc is not specified; if the regular expression contains submatches and the constraint is governed by a rule (see Chapter 8, Rules), the submatches can be referenced as %0 through %9 in the rule's score-attribute or a rule expression's attribute</td>
</tr>
<tr>
<td>node</td>
<td></td>
<td>A node's name</td>
</tr>
<tr>
<td>score</td>
<td></td>
<td>Positive values indicate a preference for running the affected resource(s) on this node — the higher the value, the stronger the preference. Negative values indicate the resource(s) should avoid this node (a value of -INFINITY changes &quot;should&quot; to &quot;must&quot;).</td>
</tr>
</tbody>
</table>

\(^1\) http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap09.html#tag_09_04
<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resource-discovery</td>
<td>always</td>
<td>Whether Pacemaker should perform resource discovery (that is, check whether the resource is already running) for this resource on this node. This should normally be left as the default, so that rogue instances of a service can be stopped when they are running where they are not supposed to be. However, there are two situations where disabling resource discovery is a good idea: when a service is not installed on a node, discovery might return an error (properly written OCF agents will not, so this is usually only seen with other agent types); and when Pacemaker Remote is used to scale a cluster to hundreds of nodes, limiting resource discovery to allowed nodes can significantly boost performance.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>always</strong>: Always perform resource discovery for the specified resource on this node.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>never</strong>: Never perform resource discovery for the specified resource on this node. This option should generally be used with a -INFINITY score, although that is not strictly required.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>exclusive</strong>: Perform resource discovery for the specified resource only on this node (and other nodes similarly marked as exclusive). Multiple location constraints using exclusive discovery for the same resource across different nodes creates a subset of nodes resource-discovery is exclusive to. If a resource is marked for exclusive discovery on one or more nodes, that resource is only allowed to be placed within that subset of nodes.</td>
</tr>
</tbody>
</table>

**Warning**

Setting resource-discovery to **never** or **exclusive** removes Pacemaker's ability to detect and stop unwanted instances of a service running where it's not supposed to be. It is up to the system administrator (you!) to make sure that the service can never be active on nodes without resource-discovery (such as by leaving the relevant software uninstalled).

**5.2.2. Asymmetrical "Opt-In" Clusters**

To create an opt-in cluster, start by preventing resources from running anywhere by default:

```
# crm_attribute --name symmetric-cluster --update false
```

Then start enabling nodes. The following fragment says that the web server prefers **sles-1**, the database prefers **sles-2** and both can fail over to **sles-3** if their most preferred node fails.
Chapter 5. Resource Constraints

Example 5.1. Opt-in location constraints for two resources

```xml
<constraints>
  <rsc_location id="loc-1" rsc="Webserver" node="sles-1" score="200"/>
  <rsc_location id="loc-2" rsc="Webserver" node="sles-3" score="0"/>
  <rsc_location id="loc-3" rsc="Database" node="sles-2" score="200"/>
  <rsc_location id="loc-4" rsc="Database" node="sles-3" score="0"/>
</constraints>
```

5.2.3. Symmetrical "Opt-Out" Clusters

To create an opt-out cluster, start by allowing resources to run anywhere by default:

```
# crm_attribute --name symmetric-cluster --update true
```

Then start disabling nodes. The following fragment is the equivalent of the above opt-in configuration.

Example 5.2. Opt-out location constraints for two resources

```xml
<constraints>
  <rsc_location id="loc-1" rsc="Webserver" node="sles-1" score="200"/>
  <rsc_location id="loc-2-dont-run" rsc="Webserver" node="sles-2" score="-INFINITY"/>
  <rsc_location id="loc-3-dont-run" rsc="Database" node="sles-1" score="-INFINITY"/>
  <rsc_location id="loc-4" rsc="Database" node="sles-2" score="200"/>
</constraints>
```

5.2.4. What if Two Nodes Have the Same Score

If two nodes have the same score, then the cluster will choose one. This choice may seem random and may not be what was intended, however the cluster was not given enough information to know any better.

Example 5.3. Constraints where a resource prefers two nodes equally

```xml
<constraints>
  <rsc_location id="loc-1" rsc="Webserver" node="sles-1" score="INFINITY"/>
  <rsc_location id="loc-2-dont-run" rsc="Webserver" node="sles-2" score="INFINITY"/>
  <rsc_location id="loc-3-dont-run" rsc="Database" node="sles-1" score="500"/>
  <rsc_location id="loc-4-dont-run" rsc="Database" node="sles-2" score="300"/>
  <rsc_location id="loc-5-dont-run" rsc="Database" node="sles-2" score="200"/>
</constraints>
```

In the example above, assuming no other constraints and an inactive cluster, Webserver would probably be placed on sles-1 and Database on sles-2. It would likely have placed Webserver based on the node's uname and Database based on the desire to spread the resource load evenly across the cluster. However other factors can also be involved in more complex configurations.

5.3. Specifying the Order in which Resources Should Start/Stop
Ordering constraints tell the cluster the order in which resources should start.

**Important**

Ordering constraints affect only the ordering of resources; they do not require that the resources be placed on the same node. If you want resources to be started on the same node and in a specific order, you need both an ordering constraint and a colocation constraint (see Section 5.4, “Placing Resources Relative to other Resources”), or alternatively, a group (see Section 9.1, “Groups - A Syntactic Shortcut”).

### 5.3.1. Ordering Properties

Table 5.2. Properties of a rsc_order Constraint

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td></td>
<td>A unique name for the constraint</td>
</tr>
<tr>
<td>first</td>
<td></td>
<td>Name of the resource that the then resource depends on</td>
</tr>
<tr>
<td>then</td>
<td></td>
<td>Name of the dependent resource</td>
</tr>
<tr>
<td>first-action</td>
<td>start</td>
<td>The action that the first resource must complete before then-action can be initiated for the then resource. Allowed values: start, stop, promote, demote.</td>
</tr>
<tr>
<td>then-action</td>
<td>value of first-action</td>
<td>The action that the then resource can execute only after the first-action on the first resource has completed. Allowed values: start, stop, promote, demote.</td>
</tr>
<tr>
<td>kind</td>
<td>Mandatory</td>
<td>How to enforce the constraint. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Optional: Just a suggestion. Only applies if both resources are executing the specified actions. Any change in state by the first resource will have no effect on the then resource.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mandatory: Always. If first does not perform first-action, then will not be allowed to performed then-action. If first is restarted, then (if running) will be stopped beforehand and started afterward.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Serializel: Ensure that no two stop/start actions occur concurrently for the resources. First and then can start in either order, but one must complete starting before the other can be started. A typical use case is when resource start-up puts a high load on the host.</td>
</tr>
<tr>
<td>symmetrical</td>
<td>TRUE for Mandatory and Optional kinds. FALSE for Serialize kind.</td>
<td>If true, the reverse of the constraint applies for the opposite action (for example, if B starts after A starts, then B stops before A stops). Serialize orders cannot be symmetrical.</td>
</tr>
</tbody>
</table>
**Chapter 5. Resource Constraints**

*Promote* and *demote* apply to the master role of *promotable* resources.

### 5.3.2. Optional and mandatory ordering

Here is an example of ordering constraints where *Database* must start before *Webserver*, and *IP* should start before *Webserver* if they both need to be started:

Example 5.4. Optional and mandatory ordering constraints

```xml
<constraints>
<rsc_order id="order-1" first="IP" then="Webserver" kind="Optional"/>
<rsc_order id="order-2" first="Database" then="Webserver" kind="Mandatory"/>
</constraints>
```

Because the above example lets *symmetrical* default to TRUE, *Webserver* must be stopped before *Database* can be stopped, and *Webserver* should be stopped before *IP* if they both need to be stopped.

### 5.4. Placing Resources Relative to other Resources

*Colocation constraints* tell the cluster that the location of one resource depends on the location of another one.

Colocation has an important side-effect: it affects the order in which resources are assigned to a node. Think about it: You can't place A relative to B unless you know where B is. ²

So when you are creating colocation constraints, it is important to consider whether you should colocate A with B, or B with A.

Another thing to keep in mind is that, assuming A is colocated with B, the cluster will take into account A's preferences when deciding which node to choose for B.

For a detailed look at exactly how this occurs, see *Colocation Explained³*.

---

² While the human brain is sophisticated enough to read the constraint in any order and choose the correct one depending on the situation, the cluster is not quite so smart. Yet.

5.4.1. Colocation Properties

Table 5.3. Properties of a rsc_colocation Constraint

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>A unique name for the constraint (required).</td>
<td></td>
</tr>
<tr>
<td>rsc</td>
<td>The name of a resource that should be located relative to with-rsc (required).</td>
<td></td>
</tr>
<tr>
<td>with-rsc</td>
<td>The name of the resource used as the colocation target. The cluster will decide where to put this resource first and then decide where to put rsc (required).</td>
<td></td>
</tr>
<tr>
<td>node-attribute</td>
<td>#uname</td>
<td>The node attribute that must be the same on the node running rsc and the node running with-rsc for the constraint to be satisfied. (For details, see Section 5.4.4, “Colocation by Node Attribute”.)</td>
</tr>
<tr>
<td>score</td>
<td>Positive values indicate the resources should run on the same node. Negative values indicate the resources should run on different nodes. Values of +/- INFINITY change &quot;should&quot; to &quot;must&quot;.</td>
<td></td>
</tr>
</tbody>
</table>

5.4.2. Mandatory Placement

Mandatory placement occurs when the constraint’s score is +INFINITY or -INFINITY. In such cases, if the constraint can’t be satisfied, then the rsc resource is not permitted to run. For score=INFINITY, this includes cases where the with-rsc resource is not active.

If you need resource A to always run on the same machine as resource B, you would add the following constraint:

Example 5.5. Mandatory colocation constraint for two resources

```
<rsc_colocation id="colocate" rsc="A" with-rsc="B" score="INFINITY"/>
```

Remember, because INFINITY was used, if B can’t run on any of the cluster nodes (for whatever reason) then A will not be allowed to run. Whether A is running or not has no effect on B.

Alternatively, you may want the opposite — that A cannot run on the same machine as B. In this case, use score="-INFINITY".

Example 5.6. Mandatory anti-colocation constraint for two resources

```
<rsc_colocation id="anti-colocate" rsc="A" with-rsc="B" score="-INFINITY"/>
```

Again, by specifying -INFINITY, the constraint is binding. So if the only place left to run is where B already is, then A may not run anywhere.

As with INFINITY, B can run even if A is stopped. However, in this case A also can run if B is stopped, because it still meets the constraint of A and B not running on the same node.
5.4.3. Advisory Placement

If mandatory placement is about "must" and "must not", then advisory placement is the "I'd prefer if" alternative. For constraints with scores greater than -\texttt{INFINITY} and less than \texttt{INFINITY}, the cluster will try to accommodate your wishes but may ignore them if the alternative is to stop some of the cluster resources.

As in life, where if enough people prefer something it effectively becomes mandatory, advisory colocation constraints can combine with other elements of the configuration to behave as if they were mandatory.

Example 5.7. Advisory colocation constraint for two resources

\begin{verbatim}
<rsc_colocation id="colocate-maybe" rsc="A" with-rsc="B" score="500"/>
\end{verbatim}

5.4.4. Colocation by Node Attribute

The \texttt{node-attribute} property of a colocation constraints allows you to express the requirement, "these resources must be on similar nodes".

As an example, imagine that you have two Storage Area Networks (SANs) that are not controlled by the cluster, and each node is connected to one or the other. You may have two resources \texttt{r1} and \texttt{r2} such that \texttt{r2} needs to use the same SAN as \texttt{r1}, but doesn't necessarily have to be on the same exact node. In such a case, you could define a \texttt{node attribute} named \texttt{san}, with the value \texttt{san1} or \texttt{san2} on each node as appropriate. Then, you could colocate \texttt{r2} with \texttt{r1} using \texttt{node-attribute} set to \texttt{san}.

5.5. Resource Sets

Resource sets allow multiple resources to be affected by a single constraint.

Example 5.8. A set of 3 resources

\begin{verbatim}
<resource_set id="resource-set-example">
  <resource_ref id="A"/>
  <resource_ref id="B"/>
  <resource_ref id="C"/>
</resource_set>
\end{verbatim}

Resource sets are valid inside \texttt{rsc_location}, \texttt{rsc_order} (see Section 5.6, "Ordering Sets of Resources"), \texttt{rsc_colocation} (see Section 5.7, "Colocating Sets of Resources"), and \texttt{rsc_ticket} (see Section 14.3, "Configuring Ticket Dependencies") constraints.

A resource set has a number of properties that can be set, though not all have an effect in all contexts.

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td></td>
<td>A unique name for the set</td>
</tr>
<tr>
<td>sequential</td>
<td>true</td>
<td>Whether the members of the set must be acted on in order. Meaningful within rsc_order and rsc_colocation.</td>
</tr>
<tr>
<td>require-all</td>
<td>true</td>
<td>Whether all members of the set must be active before continuing. With the current implementation, the cluster may continue even if only one member of the set is started,</td>
</tr>
</tbody>
</table>
### 5.6. Ordering Sets of Resources

A common situation is for an administrator to create a chain of ordered resources, such as:

**Example 5.9. 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>
```

![Figure 5.1. Visual representation of the four resources' start order for the above constraints](image)

**5.6.1. Ordered Set**

To simplify this situation, resource sets (see Section 5.5, “Resource Sets”) can be used within ordering constraints:

**Example 5.10. A chain of ordered resources expressed as a 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>
```

While the set-based format is not less verbose, it is significantly easier to get right and maintain.
5.6.2. Ordering Multiple Sets

The syntax can be expanded to allow sets of resources to be ordered relative to each other, where the members of each individual set may be ordered or unordered (controlled by the `sequential` property). In the example below, A and B can both start in parallel, as can C and D, however C and D can only start once both A and B are active.

Example 5.11. Ordered sets of unordered resources

```xml
<constraints>
    <rsc_order id="order-1">
        <resource_set id="ordered-set-1" sequential="false">
            <resource_ref id="A"/>
            <resource_ref id="B"/>
        </resource_set>
        <resource_set id="ordered-set-2" sequential="false">
            <resource_ref id="C"/>
            <resource_ref id="D"/>
        </resource_set>
    </rsc_order>
</constraints>
```

![Diagram showing the start order for two ordered sets of unordered resources](image)

Of course either set—or both sets—of resources can also be internally ordered (by setting `sequential="true"`) and there is no limit to the number of sets that can be specified.
Example 5.12. Advanced use of set ordering - Three ordered sets, two of which are internally unordered

```xml
<constraints>
  <rsc_order id="order-1">
    <resource_set id="ordered-set-1" sequential="false">
      <resource_ref id="A"/>
      <resource_ref id="B"/>
    </resource_set>
    <resource_set id="ordered-set-2" sequential="true">
      <resource_ref id="C"/>
      <resource_ref id="D"/>
    </resource_set>
    <resource_set id="ordered-set-3" sequential="false">
      <resource_ref id="E"/>
      <resource_ref id="F"/>
    </resource_set>
  </rsc_order>
</constraints>
```

Figure 5.3. Visual representation of the start order for the three sets defined above

**Important**

An ordered set with `sequential=false` makes sense only if there is another set in the constraint. Otherwise, the constraint has no effect.

5.6.3. Resource Set OR Logic

The unordered set logic discussed so far has all been "AND" logic. To illustrate this take the 3 resource set figure in the previous section. Those sets can be expressed, \((A \text{ and } B) \text{ then } (C) \text{ then } (D) \text{ then } (E \text{ and } F)\).

Say for example we want to change the first set, \((A \text{ and } B)\), to use “OR” logic so the sets look like this: \((A \text{ or } B) \text{ then } (C) \text{ then } (D) \text{ then } (E \text{ and } F)\). This functionality can be achieved through the use of the `require-all` option. This option defaults to TRUE which is why the “AND” logic is used by default. Setting `require-all=false` means only one resource in the set needs to be started before continuing on to the next set.
Example 5.13. Resource Set “OR” logic: Three ordered sets, where the first set is internally unordered with “OR” logic

```xml
<constraints>
    <rsc_order id="order-1">
        <resource_set id="ordered-set-1" sequential="false" require-all="false">
            <resource_ref id="A"/>
            <resource_ref id="B"/>
        </resource_set>
        <resource_set id="ordered-set-2" sequential="true">
            <resource_ref id="C"/>
            <resource_ref id="D"/>
        </resource_set>
        <resource_set id="ordered-set-3" sequential="false">
            <resource_ref id="E"/>
            <resource_ref id="F"/>
        </resource_set>
    </rsc_order>
</constraints>
```

Important

An ordered set with require-all=false makes sense only in conjunction with sequential=false. Think of it like this: sequential=false modifies the set to be an unordered set using "AND" logic by default, and adding require-all=false flips the unordered set's "AND" logic to "OR" logic.

5.7. Colocating Sets of Resources

Another common situation is for an administrator to create a set of colocated resources.

The simplest way to do this is to define a resource group (see Section 9.1, “Groups - A Syntactic Shortcut”), but that cannot always accurately express the desired relationships. For example, maybe the resources do not need to be ordered.

Another way would be to define each relationship as an individual constraint, but that causes a difficult-to-follow constraint explosion as the number of resources and combinations grow.

Example 5.14. Colocation chain as individual constraints, where A is placed first, then B, then C, then D

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

To express complicated relationships with a simplified syntax, resource sets can be used within colocation constraints.

---

4 which is not the same as saying easy to follow
Example 5.15. Equivalent colocation chain expressed using 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>
```

Note

Within a resource_set, the resources are listed in the order they are placed, which is the reverse of the order in which they are colocated. In the above example, resource A is placed before resource B, which is the same as saying resource B is colocated with resource A.

As with individual constraints, a resource that can’t be active prevents any resource that must be colocated with it from being active. In both of the two previous examples, if B is unable to run, then both C and by inference D must remain stopped.

Important

If you use a higher-level tool, pay attention to how it exposes this functionality. Depending on the tool, creating a set A B may be equivalent to A with B, or B with A.

Resource sets can also be used to tell the cluster that entire sets of resources must be colocated relative to each other, while the individual members within any one set may or may not be colocated relative to each other (determined by the set’s sequential property).

In the following example, resources B, C, and D will each be colocated with A (which will be placed first). A must be able to run in order for any of the resources to run, but any of B, C, or D may be stopped without affecting any of the others.

Example 5.16. Using colocated sets to specify a shared dependency

```xml
<constraints>
  <rsc_colocation id="coloc-1" score="INFINITY">
    <resource_set id="colocated-set-2" sequential="false">
      <resource_ref id="B"/>
      <resource_ref id="C"/>
      <resource_ref id="D"/>
    </resource_set>
    <resource_set id="colocated-set-1" sequential="true">
      <resource_ref id="A"/>
    </resource_set>
  </rsc_colocation>
</constraints>
```
Chapter 5. Resource Constraints

Note

Pay close attention to the order in which resources and sets are listed. While the members of any one sequential set are placed first to last (i.e., the colocation dependency is last with first), multiple sets are placed last to first (i.e. the colocation dependency is first with last).

Important

A colocated set with \texttt{sequential="false"} makes sense only if there is another set in the constraint. Otherwise, the constraint has no effect.

There is no inherent limit to the number and size of the sets used. The only thing that matters is that in order for any member of one set in the constraint to be active, all members of sets listed after it must also be active (and naturally on the same node); and if a set has \texttt{sequential="true"}, then in order for one member of that set to be active, all members listed before it must also be active.

If desired, you can restrict the dependency to instances of promotable clone resources that are in a specific role, using the set's \texttt{role} property.

Example 5.17. Colocation in which the members of the middle set have no interdependencies, and the last set listed applies only to instances in the master role

```xml
<constraints>
  <rsc_colocation id="coloc-1" score="INFINITY">
    <resource_set id="colocated-set-1" sequential="true">
      <resource_ref id="F"/>
      <resource_ref id="G"/>
    </resource_set>
    <resource_set id="colocated-set-2" sequential="false">
      <resource_ref id="C"/>
      <resource_ref id="D"/>
      <resource_ref id="E"/>
    </resource_set>
    <resource_set id="colocated-set-3" sequential="true" role="Master">
      <resource_ref id="A"/>
      <resource_ref id="B"/>
    </resource_set>
  </rsc_colocation>
</constraints>
```
Unlike ordered sets, colocated sets do not use the \texttt{require-all} option.
Alerts

Table of Contents

6.1. Alert Agents ................................................................. 47
6.2. Alert Recipients ............................................................ 47
6.3. Alert Meta-Attributes ..................................................... 48
6.4. Alert Instance Attributes ............................................... 48
6.5. Alert Filters ................................................................. 49
6.6. Using the Sample Alert Agents ....................................... 50
6.7. Writing an Alert Agent .................................................. 50

Alerts may be configured to take some external action when a cluster event occurs (node failure, resource starting or stopping, etc.).

6.1. Alert Agents

As with resource agents, the cluster calls an external program (an alert agent) to handle alerts. The cluster passes information about the event to the agent via environment variables. Agents can do anything desired with this information (send an e-mail, log to a file, update a monitoring system, etc.).

Example 6.1. Simple alert configuration

```xml
<configuration>
  <alerts>
    <alert id="my-alert" path="/path/to/my-script.sh" />
  </alerts>
</configuration>
```

In the example above, the cluster will call my-script.sh for each event.

Multiple alert agents may be configured; the cluster will call all of them for each event.

Alert agents will be called only on cluster nodes. They will be called for events involving Pacemaker Remote nodes, but they will never be called on those nodes.

6.2. Alert Recipients

Usually alerts are directed towards a recipient. Thus each alert may be additionally configured with one or more recipients. The cluster will call the agent separately for each recipient.

Example 6.2. Alert configuration with recipient

```xml
<configuration>
  <alerts>
    <alert id="my-alert" path="/path/to/my-script.sh">
      <recipient id="my-alert-recipient" value="some-address"/>
    </alert>
  </alerts>
</configuration>
```
In the above example, the cluster will call **my-script.sh** for each event, passing the recipient **some-address** as an environment variable.

The recipient may be anything the alert agent can recognize — an IP address, an e-mail address, a file name, whatever the particular agent supports.

### 6.3. Alert Meta-Attributes

As with resource agents, meta-attributes can be configured for alert agents to affect how Pacemaker calls them.

<table>
<thead>
<tr>
<th>Meta-Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timestamp-format</td>
<td>%H:%M:%S.%06N</td>
<td>Format the cluster will use when sending the event's timestamp to the agent. This is a string as used with the date(1) command.</td>
</tr>
<tr>
<td>timeout</td>
<td>30s</td>
<td>If the alert agent does not complete within this amount of time, it will be terminated.</td>
</tr>
</tbody>
</table>

Meta-attributes can be configured per alert agent and/or per recipient.

#### Example 6.3. Alert configuration with meta-attributes

```xml
<configuration>
  <alerts>
    <alert id="my-alert" path="/path/to/my-script.sh">
      <meta_attributes id="my-alert-attributes">
        <nvpair id="my-alert-attributes-timeout" name="timeout" value="15s"/>
      </meta_attributes>
      <recipient id="my-alert-recipient1" value="someuser@example.com">
        <meta_attributes id="my-alert-recipient1-attributes">
          <nvpair id="my-alert-recipient1-timestamp-format" name="timestamp-format" value="%D %H:%M"/>
        </meta_attributes>
      </recipient>
      <recipient id="my-alert-recipient2" value="otheruser@example.com">
        <meta_attributes id="my-alert-recipient2-attributes">
          <nvpair id="my-alert-recipient2-timestamp-format" name="timestamp-format" value="%c"/>
        </meta_attributes>
      </recipient>
    </alert>
  </alerts>
</configuration>
```

In the above example, the **my-script.sh** will get called twice for each event, with each call using a 15-second timeout. One call will be passed the recipient **someuser@example.com** and a timestamp in the format **%D %H:%M**, while the other call will be passed the recipient **otheruser@example.com** and a timestamp in the format **%c**.

### 6.4. Alert Instance Attributes

As with resource agents, agent-specific configuration values may be configured as instance attributes. These will be passed to the agent as additional environment variables. The number, names and allowed values of these instance attributes are completely up to the particular agent.
Example 6.4. Alert configuration with instance attributes

```xml
<configuration>
  <alerts>
    <alert id="my-alert" path="/path/to/my-script.sh">
      <meta_attributes id="my-alert-attributes">
        <nvpair id="my-alert-attributes-timeout" name="timeout" value="15s"/>
      </meta_attributes>
      <instance_attributes id="my-alert-options">
        <nvpair id="my-alert-options-debug" name="debug" value="false"/>
      </instance_attributes>
      <recipient id="my-alert-recipient1" value="someuser@example.com"/>
    </alert>
  </alerts>
</configuration>
```

6.5. Alert Filters

By default, an alert agent will be called for node events, fencing events, and resource events. An agent may choose to ignore certain types of events, but there is still the overhead of calling it for those events. To eliminate that overhead, you may select which types of events the agent should receive.

Example 6.5. Alert configuration to receive only node events and fencing events

```xml
<configuration>
  <alerts>
    <alert id="my-alert" path="/path/to/my-script.sh">
      <select>
        <select_nodes />
        <select_fencing />
      </select>
      <recipient id="my-alert-recipient1" value="someuser@example.com"/>
    </alert>
  </alerts>
</configuration>
```

The possible options within `<select>` are `<select_nodes>`, `<select_fencing>`, `<select_resources>`, and `<select_attributes>`.

With `<select_attributes>` (the only event type not enabled by default), the agent will receive alerts when a node attribute changes. If you wish the agent to be called only when certain attributes change, you can configure that as well.

Example 6.6. Alert configuration to be called when certain node attributes change

```xml
<configuration>
  <alerts>
    <alert id="my-alert" path="/path/to/my-script.sh">
      <select>
        <select_attributes>
          <attribute id="alert-standby" name="standby" />
          <attribute id="alert-shutdown" name="shutdown" />
        </select_attributes>
      </select>
      <recipient id="my-alert-recipient1" value="someuser@example.com"/>
    </alert>
  </alerts>
</configuration>
```
Chapter 6. Alerts

Node attribute alerts are currently considered experimental. Alerts may be limited to attributes set via `attrd_updater`, and agents may be called multiple times with the same attribute value.

6.6. Using the Sample Alert Agents

Pacemaker provides several sample alert agents, installed in `/usr/share/pacemaker/alerts` by default.

While these sample scripts may be copied and used as-is, they are provided mainly as templates to be edited to suit your purposes. See their source code for the full set of instance attributes they support.

Example 6.7. Sending cluster events as SNMP traps

```xml
<configuration>
  <alerts>
    <alert id="snmp_alert" path="/path/to/alert_snmp.sh">
      <instance_attributes id="config_for_alert_snmp">
        <nvpair id="trap_node_states" name="trap_node_states" value="all"/>
      </instance_attributes>
      <meta_attributes id="config_for_timestamp">
        <nvpair id="ts_fmt" name="timestamp-format" value="%Y-%m-%d,%H:%M:%S.%01N"/>
      </meta_attributes>
      <recipient id="snmp_destination" value="192.168.1.2"/>
    </alert>
  </alerts>
</configuration>
```

Example 6.8. Sending cluster events as e-mails

```xml
<configuration>
  <alerts>
    <alert id="smtp_alert" path="/path/to/alert_smtp.sh">
      <instance_attributes id="config_for_alert_smtp">
        <nvpair id="email_sender" name="email_sender" value="donotreply@example.com"/>
      </instance_attributes>
      <recipient id="smtp_destination" value="admin@example.com"/>
    </alert>
  </alerts>
</configuration>
```

6.7. Writing an Alert Agent

Table 6.2. Environment variables passed to alert agents

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRM_alert_kind</td>
<td>The type of alert (node, fencing, resource, or attribute)</td>
</tr>
<tr>
<td>CRM_alert_version</td>
<td>The version of Pacemaker sending the alert</td>
</tr>
<tr>
<td>CRM_alert_recipient</td>
<td>The configured recipient</td>
</tr>
<tr>
<td>CRM_alert_node_sequence</td>
<td>A sequence number increased whenever an alert is being issued on the local node, which can be used to reference the</td>
</tr>
<tr>
<td>Environment Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>order in which alerts have been issued by Pacemaker. An alert for an event that happened later in time reliably has a higher sequence number than alerts for earlier events. Be aware that this number has no cluster-wide meaning.</td>
</tr>
<tr>
<td>CRM_alert_timestamp</td>
<td>A timestamp created prior to executing the agent, in the format specified by the <code>timestamp-format</code> meta-attribute. This allows the agent to have a reliable, high-precision time of when the event occurred, regardless of when the agent itself was invoked (which could potentially be delayed due to system load, etc.).</td>
</tr>
<tr>
<td>CRM_alert_timestamp_epoch</td>
<td>The same time as CRM_alert_timestamp, expressed as the integer number of seconds since January 1, 1970. This (along with CRM_alert_timestamp_usec) can be useful for alert agents that need to format time in a specific way rather than let the user configure it.</td>
</tr>
<tr>
<td>CRM_alert_timestamp_usec</td>
<td>The same time as CRM_alert_timestamp, expressed as the integer number of microseconds since CRM_alert_timestamp_epoch.</td>
</tr>
<tr>
<td>CRM_alert_node</td>
<td>Name of affected node</td>
</tr>
<tr>
<td>CRM_alert_desc</td>
<td>Detail about event. For node alerts, this is the node’s current state (member or lost). For fencing alerts, this is a summary of the requested fencing operation, including origin, target, and fencing operation error code, if any. For resource alerts, this is a readable string equivalent of CRM_alert_status.</td>
</tr>
<tr>
<td>CRM_alert_nodeid</td>
<td>ID of node whose status changed (provided with node alerts only)</td>
</tr>
<tr>
<td>CRM_alert_task</td>
<td>The requested fencing or resource operation (provided with fencing and resource alerts only)</td>
</tr>
<tr>
<td>CRM_alert_rc</td>
<td>The numerical return code of the fencing or resource operation (provided with fencing and resource alerts only)</td>
</tr>
<tr>
<td>CRM_alert_rsc</td>
<td>The name of the affected resource (resource alerts only)</td>
</tr>
<tr>
<td>CRM_alert_interval</td>
<td>The interval of the resource operation (resource alerts only)</td>
</tr>
<tr>
<td>CRM_alert_target_rc</td>
<td>The expected numerical return code of the operation (resource alerts only)</td>
</tr>
<tr>
<td>CRM_alert_status</td>
<td>A numerical code used by Pacemaker to represent the operation result (resource alerts only)</td>
</tr>
<tr>
<td>CRM_alert_exec_time</td>
<td>The (wall-clock) time, in milliseconds, that it took to execute the action. If the action timed out, CRM_alert_status will be 2, CRM_alert_desc will be &quot;Timed Out&quot;, and this value will be the action timeout. May not be supported on all platforms. (resource alerts only)</td>
</tr>
<tr>
<td>CRM_alert_attribute_name</td>
<td>The name of the node attribute that changed (attribute alerts only)</td>
</tr>
<tr>
<td>CRM_alert_attribute_value</td>
<td>The new value of the node attribute that changed (attribute alerts only)</td>
</tr>
</tbody>
</table>
Chapter 6. Alerts

Special concerns when writing alert agents:

- Alert agents may be called with no recipient (if none is configured), so the agent must be able to handle this situation, even if it only exits in that case. (Users may modify the configuration in stages, and add a recipient later.)

- If more than one recipient is configured for an alert, the alert agent will be called once per recipient. If an agent is not able to run concurrently, it should be configured with only a single recipient. The agent is free, however, to interpret the recipient as a list.

- When a cluster event occurs, all alerts are fired off at the same time as separate processes. Depending on how many alerts and recipients are configured, and on what is done within the alert agents, a significant load burst may occur. The agent could be written to take this into consideration, for example by queueing resource-intensive actions into some other instance, instead of directly executing them.

- Alert agents are run as the hacluster user, which has a minimal set of permissions. If an agent requires additional privileges, it is recommended to configure sudo to allow the agent to run the necessary commands as another user with the appropriate privileges.

- As always, take care to validate and sanitize user-configured parameters, such as CRM_alert_timestamp (whose content is specified by the user-configured timestamp-format), CRM_alert_recipient, and all instance attributes. Mostly this is needed simply to protect against configuration errors, but if some user can modify the CIB without having hacluster-level access to the cluster nodes, it is a potential security concern as well, to avoid the possibility of code injection.

**Note**

The alerts interface is designed to be backward compatible with the external scripts interface used by the ocf:pacemaker:ClusterMon resource, which is now deprecated. To preserve this compatibility, the environment variables passed to alert agents are available prepended with CRM_notify_ as well as CRM_alert_. One break in compatibility is that ClusterMon ran external scripts as the root user, while alert agents are run as the hacluster user.
Rules

Table of Contents

7.1. Rule Properties ................................................................................................................... 53
7.2. Node Attribute Expressions ................................................................................................. 54
7.3. Time- and Date-Based Expressions ..................................................................................... 55
  7.3.1. Date Specifications ................................................................................................... 56
  7.3.2. Durations ................................................................................................................. 56
  7.3.3. Sample Time-Based Expressions .............................................................................. 56
7.4. Using Rules to Determine Resource Location ....................................................................... 58
  7.4.1. Location Rules Based on Other Node Properties ....................................................... 58
  7.4.2. Using score-attribute Instead of score ...................................................................... 59
7.5. Using Rules to Control Resource Options ............................................................................ 59
7.6. Using Rules to Control Cluster Options ................................................................................ 60
7.7. Ensuring Time-Based Rules Take Effect .............................................................................. 61

Rules can be used to make your configuration more dynamic. One common example is to set one value for resource-stickiness during working hours, to prevent resources from being moved back to their most preferred location, and another on weekends when no-one is around to notice an outage.

Another use of rules might be to assign machines to different processing groups (using a node attribute) based on time and to then use that attribute when creating location constraints.

Each rule can contain a number of expressions, date-expressions and even other rules. The results of the expressions are combined based on the rule’s boolean-op field to determine if the rule ultimately evaluates to true or false. What happens next depends on the context in which the rule is being used.

7.1. Rule Properties

Table 7.1. Properties of a Rule

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td></td>
<td>A unique name for the rule (required)</td>
</tr>
<tr>
<td>role</td>
<td>Started</td>
<td>Limits the rule to apply only when the resource is in the specified role. Allowed values are Started, Slave, and Master. A rule with role=&quot;Master&quot; cannot determine the initial location of a clone instance and will only affect which of the active instances will be promoted.</td>
</tr>
<tr>
<td>score</td>
<td></td>
<td>The score to apply if the rule evaluates to true. Limited to use in rules that are part of location constraints.</td>
</tr>
<tr>
<td>score-attribute</td>
<td></td>
<td>The node attribute to look up and use as a score if the rule evaluates to true. Limited to use in rules that are part of location constraints.</td>
</tr>
<tr>
<td>boolean-op</td>
<td>and</td>
<td>How to combine the result of multiple expression objects. Allowed values are and and or.</td>
</tr>
</tbody>
</table>
7.2. Node Attribute Expressions

Expression objects are used to control a resource based on the attributes defined by a node or nodes.

Table 7.2. Properties of an Expression

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td></td>
<td>A unique name for the expression (required)</td>
</tr>
<tr>
<td>attribute</td>
<td></td>
<td>The node attribute to test (required)</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>Determines how the value(s) should be tested. Allowed values are string, integer, and version.</td>
</tr>
<tr>
<td>operation</td>
<td></td>
<td>The comparison to perform (required). Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>lt</strong>: True if the value of the node's attribute is less than <strong>value</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>gt</strong>: True if the value of the node's attribute is greater than <strong>value</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>lte</strong>: True if the value of the node's attribute is less than or equal to <strong>value</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>gte</strong>: True if the value of the node's attribute is greater than or equal to <strong>value</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>eq</strong>: True if the value of the node's attribute is equal to <strong>value</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>ne</strong>: True if the value of the node's attribute is not equal to <strong>value</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>defined</strong>: True if the node has the named attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>not_defined</strong>: True if the node does not have the named attribute</td>
</tr>
<tr>
<td>value</td>
<td></td>
<td>User-supplied value for comparison (required)</td>
</tr>
<tr>
<td>value-source</td>
<td>literal</td>
<td>How the <strong>value</strong> is derived. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>literal</strong>: <strong>value</strong> is a literal string to compare against</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>param</strong>: <strong>value</strong> is the name of a resource parameter to compare against (only valid in location constraints)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>meta</strong>: <strong>value</strong> is the name of a resource meta-attribute to compare against (only valid in location constraints)</td>
</tr>
</tbody>
</table>

In addition to any attributes added by the administrator, the cluster defines special, built-in node attributes for each node that can also be used.

Table 7.3. Built-in node attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#uname</td>
<td>Node name</td>
</tr>
<tr>
<td>#id</td>
<td>Node ID</td>
</tr>
</tbody>
</table>
### 7.3. Time- and Date-Based Expressions

As the name suggests, *date_expressions* are used to control a resource or cluster option based on the current date/time. They may contain an optional *date_spec* and/or *duration* object depending on the context.

**Table 7.4. Properties of a Date Expression**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>start</strong></td>
<td>A date/time conforming to the <em>ISO8601</em> specification.</td>
</tr>
<tr>
<td><strong>end</strong></td>
<td>A date/time conforming to the <em>ISO8601</em> specification. Can be inferred by supplying a value for <strong>start</strong> and a <strong>duration</strong>.</td>
</tr>
<tr>
<td><strong>operation</strong></td>
<td>Compares the current date/time with the start and/or end date, depending on the context. Allowed values:</td>
</tr>
<tr>
<td></td>
<td>• <em>gt</em>: True if the current date/time is after <strong>start</strong></td>
</tr>
<tr>
<td></td>
<td>• <em>lt</em>: True if the current date/time is before <strong>end</strong></td>
</tr>
<tr>
<td></td>
<td>• <em>in_range</em>: True if the current date/time is after <strong>start</strong> and before <strong>end</strong></td>
</tr>
<tr>
<td></td>
<td>• <em>date_spec</em>: True if the current date/time matches a <em>date_spec</em> object (described below)</td>
</tr>
</tbody>
</table>

**Note**

As these comparisons (except for *date_spec*) include the time, the *eq*, *neq*, *gte* and *lte* operators have not been implemented since they would only be valid for a single second.

---

7.3.1. Date Specifications

date_spec objects are used to create cron-like expressions relating to time. Each field can contain a single number or a single range. Instead of defaulting to zero, any field not supplied is ignored.

For example, monthdays="1" matches the first day of every month and hours="09-17" matches the hours between 9am and 5pm (inclusive). At this time, multiple ranges (e.g. weekdays="1,2" or weekdays="1-2,5-6") are not supported; depending on demand, this might be implemented in a future release.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>A unique name for the object</td>
</tr>
<tr>
<td>hours</td>
<td>Allowed values: 0-23</td>
</tr>
<tr>
<td>monthdays</td>
<td>Allowed values: 1-31 (depending on month and year)</td>
</tr>
<tr>
<td>weekdays</td>
<td>Allowed values: 1-7 (1=Monday, 7=Sunday)</td>
</tr>
<tr>
<td>yeardays</td>
<td>Allowed values: 1-366 (depending on the year)</td>
</tr>
<tr>
<td>months</td>
<td>Allowed values: 1-12</td>
</tr>
<tr>
<td>weeks</td>
<td>Allowed values: 1-53 (depending on weekyear)</td>
</tr>
<tr>
<td>years</td>
<td>Year according to the Gregorian calendar</td>
</tr>
<tr>
<td>weekyears</td>
<td>Year in which the week started; e.g. 1 January 2005 can be specified as 2005-001 Ordinal, 2005-01-01 Gregorian or 2004-W53-6 Weekly and thus would match years=&quot;2005&quot; or weekyears=&quot;2004&quot;</td>
</tr>
<tr>
<td>moon</td>
<td>Allowed values are 0-7 (0 is new, 4 is full moon). Seriously, you can use this. This was implemented to demonstrate the ease with which new comparisons could be added.</td>
</tr>
</tbody>
</table>

7.3.2. Durations

Durations are used to calculate a value for end when one is not supplied to in_range operations. They contain the same fields as date_spec objects but without the limitations (e.g. you can have a duration of 19 months). As with date_specs, any field not supplied is ignored.

7.3.3. Sample Time-Based Expressions

A small sample of how time-based expressions can be used:

Example 7.1. True if now is any time in the year 2005

```xml
<rule id="rule1">
  <date_expression id="date_expr1" start="2005-001" operation="in_range">
    <duration years="1"/>
  </date_expression>
</rule>
```
Sample Time-Based Expressions

Example 7.2. Equivalent expression

```xml
<rule id="rule2">
  <date_expression id="date_expr2" operation="date_spec">
    <date_spec years="2005"/>
  </date_expression>
</rule>
```

Example 7.3. 9am-5pm Monday-Friday

```xml
<rule id="rule3">
  <date_expression id="date_expr3" operation="date_spec">
    <date_spec hours="9-16" days="1-5"/>
  </date_expression>
</rule>
```

Please note that the `16` matches up to `16:59:59`, as the numeric value (hour) still matches!

Example 7.4. 9am-6pm Monday through Friday or anytime Saturday

```xml
<rule id="rule4" boolean-op="or">
  <date_expression id="date_expr4-1" operation="date_spec">
    <date_spec hours="9-16" days="1-5"/>
  </date_expression>
  <date_expression id="date_expr4-2" operation="date_spec">
    <date_spec days="6"/>
  </date_expression>
</rule>
```

Example 7.5. 9am-5pm or 9pm-12am Monday through Friday

```xml
<rule id="rule5" boolean-op="and">
  <rule id="rule5-nested1" boolean-op="or">
    <date_expression id="date_expr5-1" operation="date_spec">
      <date_spec hours="9-16"/>
    </date_expression>
    <date_expression id="date_expr5-2" operation="date_spec">
      <date_spec hours="21-23"/>
    </date_expression>
  </rule>
  <date_expression id="date_expr5-3" operation="date_spec">
    <date_spec days="1-5"/>
  </date_expression>
</rule>
```

Example 7.6. Mondays in March 2005

```xml
<rule id="rule6" boolean-op="and">
  <date_expression id="date_expr6-1" operation="date_spec">
    <date_spec weekdays="1"/>
  </date_expression>
  <date_expression id="date_expr6-2" operation="in_range" start="2005-03-01" end="2005-04-01"/>
</rule>
```
Because no time is specified with the above dates, 00:00:00 is implied. This means that the range includes all of 2005-03-01 but none of 2005-04-01. You may wish to write end="2005-03-31T23:59:59" to avoid confusion.

Example 7.7. A full moon on Friday the 13th

```xml
<rule id="rule7" boolean-op="and">
  <date_expression id="date_expr7" operation="date_spec">
    <date_spec weekdays="5" monthdays="13" moon="4"/>
  </date_expression>
</rule>
```

7.4. Using Rules to Determine Resource Location

A location constraint may contain rules. When the constraint's outermost rule evaluates to **false**, the cluster treats the constraint as if it were not there. When the rule evaluates to **true**, the node's preference for running the resource is updated with the score associated with the rule.

If this sounds familiar, it is because you have been using a simplified syntax for location constraint rules already. Consider the following location constraint:

Example 7.8. Prevent myApacheRsc from running on c001n03

```xml
<rsc_location id="dont-run-apache-on-c001n03" rsc="myApacheRsc" score="-INFINITY" node="c001n03"/>
```

This constraint can be more verbosely written as:

Example 7.9. Prevent myApacheRsc from running on c001n03 - expanded version

```xml
<rsc_location id="dont-run-apache-on-c001n03" rsc="myApacheRsc">
  <rule id="dont-run-apache-rule" score="-INFINITY">
    <expression id="dont-run-apache-expr" attribute="#uname" operation="eq" value="c001n03"/>
  </rule>
</rsc_location>
```

The advantage of using the expanded form is that one can then add extra clauses to the rule, such as limiting the rule such that it only applies during certain times of the day or days of the week.

7.4.1. Location Rules Based on Other Node Properties

The expanded form allows us to match on node properties other than its name. If we rated each machine's CPU power such that the cluster had the following nodes section:
Using `score-attribute` Instead of `score`

Example 7.10. A sample nodes section for use with `score-attribute`

```xml
<nodes>
  <node id="uuid1" uname="c001n01" type="normal">
    <instance_attributes id="uuid1-custom_attrs">
      <nvpair id="uuid1-cpu_mips" name="cpu_mips" value="1234"/>
    </instance_attributes>
  </node>
  <node id="uuid2" uname="c001n02" type="normal">
    <instance_attributes id="uuid2-custom_attrs">
      <nvpair id="uuid2-cpu_mips" name="cpu_mips" value="5678"/>
    </instance_attributes>
  </node>
</nodes>
```

then we could prevent resources from running on underpowered machines with this rule:

```xml
<rule id="need-more-power-rule" score="-INFINITY">
  <expression id="need-more-power-expr" attribute="cpu_mips" operation="lt" value="3000"/>
</rule>
```

7.4.2. Using `score-attribute` Instead of `score`

When using `score-attribute` instead of `score`, each node matched by the rule has its score adjusted differently, according to its value for the named node attribute. Thus, in the previous example, if a rule used `score-attribute="cpu_mips"`, `c001n01` would have its preference to run the resource increased by `1234` whereas `c001n02` would have its preference increased by `5678`.

7.5. Using Rules to Control Resource Options

Often some cluster nodes will be different from their peers. Sometimes, these differences — e.g. the location of a binary or the names of network interfaces — require resources to be configured differently depending on the machine they’re hosted on.

By defining multiple `instance_attributes` objects for the resource and adding a rule to each, we can easily handle these special cases.

In the example below, `mySpecialRsc` will use eth1 and port 9999 when run on `node1`, eth2 and port 8888 on `node2` and default to eth0 and port 9999 for all other nodes.

Example 7.11. Defining different resource options based on the node name

```xml
<primitive id="mySpecialRsc" class="ocf" type="Special" provider="me">
  <instance_attributes id="special-node1" score="3">
    <rule id="node1-special-case" score="INFINITY">
      <expression id="node1-special-case-expr" attribute="#uname" operation="eq" value="node1"/>
    </rule>
    <nvpair id="node1-interface" name="interface" value="eth1"/>
  </instance_attributes>
  <instance_attributes id="special-node2" score="2">
    <rule id="node2-special-case" score="INFINITY">
      <expression id="node2-special-case-expr" attribute="#uname" operation="eq" value="node2"/>
    </rule>
    <nvpair id="node2-interface" name="interface" value="eth2"/>
  </instance_attributes>
</primitive>
```
The order in which instance_attributes objects are evaluated is determined by their score (highest to lowest). If not supplied, score defaults to zero, and objects with an equal score are processed in listed order. If the instance_attributes object has no rule or a rule that evaluates to true, then for any parameter the resource does not yet have a value for, the resource will use the parameter values defined by the instance_attributes.

For example, given the configuration above, if the resource is placed on node1:

1. special-node1 has the highest score (3) and so is evaluated first; its rule evaluates to true, so interface is set to eth1.
2. special-node2 is evaluated next with score 2, but its rule evaluates to false, so it is ignored.
3. defaults is evaluated last with score 1, and has no rule, so its values are examined; interface is already defined, so the value here is not used, but port is not yet defined, so port is set to 9999.

7.6. Using Rules to Control Cluster Options

Controlling cluster options is achieved in much the same manner as specifying different resource options on different nodes.

The difference is that because they are cluster options, one cannot (or should not, because they won't work) use attribute-based expressions. The following example illustrates how to set a different resource-stickiness value during and outside work hours. This allows resources to automatically move back to their most preferred hosts, but at a time that (in theory) does not interfere with business activities.

Example 7.12. Change resource-stickiness during working hours

```
<rsc_defaults>
  <meta_attributes id="core-hours" score="2">
    <rule id="core-hour-rule" score="0">
      <date_expression id="nine-to-five-Mon-to-Fri" operation="date_spec">
        <date_spec id="nine-to-five-Mon-to-Fri-spec" hours="9-16" weekdays="1-5"/>
      </date_expression>
      <nvpair id="core-stickiness" name="resource-stickiness" value="INFINITY"/>
    </rule>
  </meta_attributes>
  <meta_attributes id="after-hours" score="1">
    <nvpair id="after-stickiness" name="resource-stickiness" value="0"/>
  </meta_attributes>
</rsc_defaults>
```
7.7. Ensuring Time-Based Rules Take Effect

A Pacemaker cluster is an event-driven system. As such, it won't recalculate the best place for resources to run unless something (like a resource failure or configuration change) happens. This can mean that a location constraint that only allows resource X to run between 9am and 5pm is not enforced.

If you rely on time-based rules, the `cluster-recheck-interval` cluster option (which defaults to 15 minutes) is essential. This tells the cluster to periodically recalculate the ideal state of the cluster.

For example, if you set `cluster-recheck-interval="5m"`, then sometime between 09:00 and 09:05 the cluster would notice that it needs to start resource X, and between 17:00 and 17:05 it would realize that X needed to be stopped. The timing of the actual start and stop actions depends on what other actions the cluster may need to perform first.
Advanced Configuration

8.1. Specifying When Recurring Actions are Performed

By default, recurring actions are scheduled relative to when the resource started. So if your resource was last started at 14:32 and you have a backup set to be performed every 24 hours, then the backup will always run in the middle of the business day — hardly desirable.

To specify a date and time that the operation should be relative to, set the operation's `interval-origin`. The cluster uses this point to calculate the correct `start-delay` such that the operation will occur at `origin + (interval * N)`.

So, if the operation's interval is 24h, its interval-origin is set to 02:00 and it is currently 14:32, then the cluster would initiate the operation with a start delay of 11 hours and 28 minutes. If the resource is moved to another node before 2am, then the operation is cancelled.

The value specified for `interval` and `interval-origin` can be any date/time conforming to the ISO8601 standard[^1]. By way of example, to specify an operation that would run on the first Monday of 2009 and every Monday after that, you would add:

```
<op id="my-weekly-action" name="custom-action" interval="P7D" interval-origin="2009-W01-1"/>
```

8.2. Handling Resource Failure

By default, Pacemaker will attempt to recover failed resources by restarting them. However, failure recovery is highly configurable.

---

Chapter 8. Advanced Configuration

8.2.1. Failure Counts

Pacemaker tracks resource failures for each combination of node, resource, and operation (start, stop, monitor, etc.).

You can query the fail count for a particular node, resource, and/or operation using the `crm_failcount` command. For example, to see how many times the 10-second monitor for `myrsc` has failed on `node1`, run:

```
# crm_failcount --query -r myrsc -N node1 -n monitor -I 10s
```

If you omit the node, `crm_failcount` will use the local node. If you omit the operation and interval, `crm_failcount` will display the sum of the fail counts for all operations on the resource.

You can use `crm_resource --cleanup` or `crm_failcount --delete` to clear fail counts. For example, to clear the above monitor failures, run:

```
# crm_resource --cleanup -r myrsc -N node1 -n monitor -I 10s
```

If you omit the resource, `crm_resource --cleanup` will clear failures for all resources. If you omit the node, it will clear failures on all nodes. If you omit the operation and interval, it will clear the failures for all operations on the resource.

---

**Note**

Even when cleaning up only a single operation, all failed operations will disappear from the status display. This allows us to trigger a re-check of the resource’s current status.

---

Higher-level tools may provide other commands for querying and clearing fail counts.

The `crm_mon` tool shows the current cluster status, including any failed operations. To see the current fail counts for any failed resources, call `crm_mon` with the `--failcounts` option. This shows the fail counts per resource (that is, the sum of any operation fail counts for the resource).

8.2.2. Failure Response

Normally, if a running resource fails, pacemaker will try to stop it and start it again. Pacemaker will choose the best location to start it each time, which may be the same node that it failed on.

However, if a resource fails repeatedly, it is possible that there is an underlying problem on that node, and you might desire trying a different node in such a case. Pacemaker allows you to set your preference via the `migration-threshold` resource meta-attribute. 2

If you define `migration-threshold=N` for a resource, it will be banned from the original node after `N` failures.

---

2 The naming of this option was perhaps unfortunate as it is easily confused with live migration, the process of moving a resource from one node to another without stopping it. Xen virtual guests are the most common example of resources that can be migrated in this manner.
Note

The **migration-threshold** is per resource, even though fail counts are tracked per operation. The operation fail counts are added together to compare against the **migration-threshold**.

By default, fail counts remain until manually cleared by an administrator using `crm_resource --cleanup` or `crm_failcount --delete` (hopefully after first fixing the failure's cause). It is possible to have fail counts expire automatically by setting the **failure-timeout** resource meta-attribute.

**Important**

A successful operation does not clear past failures. If a recurring monitor operation fails once, succeeds many times, then fails again days later, its fail count is 2. Fail counts are cleared only by manual intervention or failure timeout.

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

Note

**failure-timeout** is measured since the most recent failure. That is, older failures do not individually time out and lower the fail count. Instead, all failures are timed out simultaneously (and the fail count is reset to 0) if there is no new failure for the timeout period.

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

If the cluster property **start-failure-is-fatal** is set to **true** (which is the default), start failures cause the fail count to be set to **INFINITY** and thus always cause the resource to move immediately.

Stop failures are slightly different and crucial. If a resource fails to stop and STONITH is enabled, then the cluster will fence the node in order to be able to start the resource elsewhere. If STONITH is not enabled, then the cluster has no way to continue and will not try to start the resource elsewhere, but will try to stop it again after the failure timeout.

**Important**

Please read **Section 7.7, “Ensuring Time-Based Rules Take Effect”** to understand how timeouts work before configuring a **failure-timeout**.
8.3. Moving Resources

8.3.1. Moving Resources Manually
There are primarily two occasions when you would want to move a resource from its current location: when the whole node is under maintenance, and when a single resource needs to be moved.

8.3.1.1. Standby Mode
Since everything eventually comes down to a score, you could create constraints for every resource to prevent them from running on one node. While pacemaker configuration can seem convoluted at times, not even we would require this of administrators.

Instead, one can set a special node attribute which tells the cluster "don't let anything run here". There is even a helpful tool to help query and set it, called `crm_standby`. To check the standby status of the current machine, run:

```
# crm_standby -G
```

A value of `on` indicates that the node is *not* able to host any resources, while a value of `off` says that it *can*.

You can also check the status of other nodes in the cluster by specifying the `--node` option:

```
# crm_standby -G --node sles-2
```

To change the current node's standby status, use `-v` instead of `-G`:

```
# crm_standby -v on
```

Again, you can change another host's value by supplying a hostname with `--node`.

A cluster node in standby mode will not run resources, but still contributes to quorum, and may fence or be fenced by nodes.

8.3.1.2. Moving One Resource
When only one resource is required to move, we could do this by creating location constraints. However, once again we provide a user-friendly shortcut as part of the `crm_resource` command, which creates and modifies the extra constraints for you. If `Email` were running on `sles-1` and you wanted it moved to a specific location, the command would look something like:

```
# crm_resource -M -r Email -H sles-2
```

Behind the scenes, the tool will create the following location constraint:

```
<rsc_location rsc="Email" node="sles-2" score="INFINITY"/>
```

It is important to note that subsequent invocations of `crm_resource -M` are not cumulative. So, if you ran these commands
# crm_resource -M -r Email -H sles-2
# crm_resource -M -r Email -H sles-3

then it is as if you had never performed the first command.

To allow the resource to move back again, use:

# crm_resource -U -r Email

Note the use of the word allow. The resource can move back to its original location but, depending on resource-stickiness, it might stay where it is. To be absolutely certain that it moves back to sles-1, move it there before issuing the call to crm_resource -U:

# crm_resource -M -r Email -H sles-1
# crm_resource -U -r Email

Alternatively, if you only care that the resource should be moved from its current location, try:

# crm_resource -B -r Email

Which will instead create a negative constraint, like

<&rsc_location rsc="Email" node="sles-1" score="-INFINITY"/>

This will achieve the desired effect, but will also have long-term consequences. As the tool will warn you, the creation of a -INFINITY constraint will prevent the resource from running on that node until crm_resource -U is used. This includes the situation where every other cluster node is no longer available!

In some cases, such as when resource-stickiness is set to INFINITY, it is possible that you will end up with the problem described in Section 5.2.4, "What if Two Nodes Have the Same Score". The tool can detect some of these cases and deals with them by creating both positive and negative constraints. E.g.

Email prefers sles-1 with a score of -INFINITY

Email prefers sles-2 with a score of INFINITY

which has the same long-term consequences as discussed earlier.

8.3.2. Moving Resources Due to Connectivity Changes
You can configure the cluster to move resources when external connectivity is lost in two steps.

8.3.2.1. Tell Pacemaker to Monitor Connectivity
First, add an ocf:pacemaker:ping resource to the cluster. The ping resource uses the system utility of the same name to test whether list of machines (specified by DNS hostname or IPv4/IPv6 address) are reachable and uses the results to maintain a node attribute called pingd by default. ³

³ The attribute name is customizable, in order to allow multiple ping groups to be defined.
Older versions of Pacemaker used a different agent `ocf:pacemaker:pingd` which is now deprecated in favor of `ping`. If your version of Pacemaker does not contain the `ping` resource agent, download the latest version from [https://github.com/ClusterLabs/pacemaker/tree/master/extra/resources/ping](https://github.com/ClusterLabs/pacemaker/tree/master/extra/resources/ping).

Normally, the ping resource should run on all cluster nodes, which means that you'll need to create a clone. A template for this can be found below along with a description of the most interesting parameters.

### Table 8.1. Common Options for a `ping` Resource

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dampen</code></td>
<td>The time to wait (dampening) for further changes to occur. Use this to prevent a resource from bouncing around the cluster when cluster nodes notice the loss of connectivity at slightly different times.</td>
</tr>
<tr>
<td><code>multiplier</code></td>
<td>The number of connected ping nodes gets multiplied by this value to get a score. Useful when there are multiple ping nodes configured.</td>
</tr>
<tr>
<td><code>host_list</code></td>
<td>The machines to contact in order to determine the current connectivity status. Allowed values include resolvable DNS host names, IPv4 and IPv6 addresses.</td>
</tr>
</tbody>
</table>

**Example 8.2. An example ping cluster resource that checks node connectivity once every minute**

```xml
<clone id="Connected">
    <primitive id="ping" provider="pacemaker" class="ocf" type="ping">
        <instance_attributes id="ping-attrs">
            <nvpair id="pingd-dampen" name="dampen" value="5s"/>
            <nvpair id="pingd-multiplier" name="multiplier" value="1000"/>
            <nvpair id="pingd-hosts" name="host_list" value="my.gateway.com www.bigcorp.com"/>
        </instance_attributes>
        <operations>
            <op id="ping-monitor-60s" interval="60s" name="monitor"/>
        </operations>
    </primitive>
</clone>
```

**Important**

You're only half done. The next section deals with telling Pacemaker how to deal with the connectivity status that `ocf:pacemaker:ping` is recording.
8.3.2.2. Tell Pacemaker How to Interpret the Connectivity Data

Important

Before attempting the following, make sure you understand Chapter 8, Rules.

There are a number of ways to use the connectivity data.

The most common setup is for people to have a single ping target (e.g., the service network’s default gateway), to prevent the cluster from running a resource on any unconnected node.

Example 8.3. Don’t run a resource on unconnected nodes

```xml
<rsc_location id="WebServer-no-connectivity" rsc="Webserver">
    <rule id="ping-exclude-rule" score="-INFINITY">
        <expression id="ping-exclude" attribute="pingd" operation="not_defined"/>
    </rule>
</rsc_location>
```

A more complex setup is to have a number of ping targets configured. You can require the cluster to only run resources on nodes that can connect to all (or a minimum subset) of them.

Example 8.4. Run only on nodes connected to three or more ping targets.

```xml
<primitive id="ping" provider="pacemaker" class="ocf" type="ping">
    ... <!-- omitting some configuration to highlight important parts -->
    <nvpair id="pingd-multiplier" name="multiplier" value="1000"/>
    ...
</primitive>
...
<rsc_location id="WebServer-connectivity" rsc="Webserver">
    <rule id="ping-prefer-rule" score-attribute="pingd">
        <expression id="ping-prefer" attribute="pingd" operation="lt" value="3000"/>
    </rule>
</rsc_location>
```

Alternatively, you can tell the cluster only to prefer nodes with the best connectivity. Just be sure to set multiplier to a value higher than that of resource-stickiness (and don’t set either of them to INFINITY).

Example 8.5. Prefer the node with the most connected ping nodes

```xml
<rsc_location id="WebServer-connectivity" rsc="Webserver">
    <rule id="ping-prefer-rule" score-attribute="pingd">
        <expression id="ping-prefer" attribute="pingd" operation="defined"/>
    </rule>
</rsc_location>
```

It is perhaps easier to think of this in terms of the simple constraints that the cluster translates it into. For example, if sles-1 is connected to all five ping nodes but sles-2 is only connected to two, then it would be as if you instead had the following constraints in your configuration:
Example 8.6. How the cluster translates the above location constraint

```xml
<rsc_location id="ping-1" rsc="Webserver" node="sles-1" score="5000"/>
<rsc_location id="ping-2" rsc="Webserver" node="sles-2" score="2000"/>
```

The advantage is that you don't have to manually update any constraints whenever your network connectivity changes.

You can also combine the concepts above into something even more complex. The example below shows how you can prefer the node with the most connected ping nodes provided they have connectivity to at least three (again assuming that \texttt{multiplier} is set to 1000).

Example 8.7. A more complex example of choosing a location based on connectivity

```xml
<rsc_location id="WebServer-connectivity" rsc="Webserver">
  <rule id="ping-exclude-rule" score="-INFINITY">
    <expression id="ping-exclude" attribute="pingd" operation="lt" value="3000"/>
  </rule>
  <rule id="ping-prefer-rule" score-attribute="pingd">
    <expression id="ping-prefer" attribute="pingd" operation="defined"/>
  </rule>
</rsc_location>
```

### 8.3.3. Migrating Resources

Normally, when the cluster needs to move a resource, it fully restarts the resource (i.e. stops the resource on the current node and starts it on the new node).

However, some types of resources, such as Xen virtual guests, are able to move to another location without loss of state (often referred to as live migration or hot migration). In pacemaker, this is called resource migration. Pacemaker can be configured to migrate a resource when moving it, rather than restarting it.

Not all resources are able to migrate; see the Migration Checklist below, and those that can, won't do so in all situations. Conceptually, there are two requirements from which the other prerequisites follow:

- The resource must be active and healthy at the old location; and
- everything required for the resource to run must be available on both the old and new locations.

The cluster is able to accommodate both \textit{push} and \textit{pull} migration models by requiring the resource agent to support two special actions: \texttt{migrate\_to} (performed on the current location) and \texttt{migrate\_from} (performed on the destination).

In push migration, the process on the current location transfers the resource to the new location where it is later activated. In this scenario, most of the work would be done in the \texttt{migrate\_to} action and, if anything, the activation would occur during \texttt{migrate\_from}.

Conversely for pull, the \texttt{migrate\_to} action is practically empty and \texttt{migrate\_from} does most of the work, extracting the relevant resource state from the old location and activating it.

There is no wrong or right way for a resource agent to implement migration, as long as it works.

**Migration Checklist**

- The resource may not be a clone.
• The resource must use an OCF style agent.

• The resource must not be in a failed or degraded state.

• The resource agent must support `migrate_to` and `migrate_from` actions, and advertise them in its metadata.

• The resource must have the `allow-migrate` meta-attribute set to `true` (which is not the default).

If an otherwise migratable resource depends on another resource via an ordering constraint, there are special situations in which it will be restarted rather than migrated.

For example, if the resource depends on a clone, and at the time the resource needs to be moved, the clone has instances that are stopping and instances that are starting, then the resource will be restarted. The scheduler is not yet able to model this situation correctly and so takes the safer (if less optimal) path.

Also, if a migratable resource depends on a non-migratable resource, and both need to be moved, the migratable resource will be restarted.

**8.4. Tracking Node Health**

A node may be functioning adequately as far as cluster membership is concerned, and yet be “unhealthy” in some respect that makes it an undesirable location for resources. For example, a disk drive may be reporting SMART errors, or the CPU may be highly loaded.

Pacemaker offers a way to automatically move resources off unhealthy nodes.

**8.4.1. Node Health Attributes**

Pacemaker will treat any node attribute whose name starts with `#health` as an indicator of node health. Node health attributes may have one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Intended significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>This indicator is unhealthy</td>
</tr>
<tr>
<td>yellow</td>
<td>This indicator is becoming unhealthy</td>
</tr>
<tr>
<td>green</td>
<td>This indicator is healthy</td>
</tr>
<tr>
<td>integer</td>
<td>A numeric score to apply to all resources on this node (0 or positive is healthy, negative is unhealthy)</td>
</tr>
</tbody>
</table>

**8.4.2. Node Health Strategy**

Pacemaker assigns a node health score to each node, as the sum of the values of all its node health attributes. This score will be used as a location constraint applied to this node for all resources.

The `node-health-strategy` cluster option controls how Pacemaker responds to changes in node health attributes, and how it translates `red`, `yellow`, and `green` to scores.

Allowed values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>Do not track node health attributes at all.</td>
</tr>
</tbody>
</table>
### Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>migrate-on-red</td>
<td>Assign the value of <code>-INFINITY</code> to red, and 0 to yellow and green. This will cause all resources to move off the node if any attribute is red.</td>
</tr>
<tr>
<td>only-green</td>
<td>Assign the value of <code>-INFINITY</code> to red and yellow, and 0 to green. This will cause all resources to move off the node if any attribute is red or yellow.</td>
</tr>
<tr>
<td>progressive</td>
<td>Assign the value of the <code>node-health-red</code> cluster option to red, the value of <code>node-health-yellow</code> to yellow, and the value of <code>node-health-green</code> to green. Each node is additionally assigned a score of <code>node-health-base</code> (this allows resources to start even if some attributes are yellow). This strategy gives the administrator finer control over how important each value is.</td>
</tr>
<tr>
<td>custom</td>
<td>Track node health attributes using the same values as <code>progressive</code> for red, yellow, and green, but do not take them into account. The administrator is expected to implement a policy by defining rules (see Chapter 8, Rules) referencing node health attributes.</td>
</tr>
</tbody>
</table>

#### 8.4.3. Measuring Node Health

Since Pacemaker calculates node health based on node attributes, any method that sets node attributes may be used to measure node health. The most common ways are resource agents or separate daemons.

Pacemaker provides examples that can be used directly or as a basis for custom code. The `ocf:pacemaker:HealthCPU` and `ocf:pacemaker:HealthSMART` resource agents set node health attributes based on CPU and disk parameters. The `ipmiservicelogd` daemon sets node health attributes based on IPMI values (the `ocf:pacemaker:SystemHealth` resource agent can be used to manage the daemon as a cluster resource).

#### 8.5. Reloading Services After a Definition Change

The cluster automatically detects changes to the definition of services it manages. The normal response is to stop the service (using the old definition) and start it again (with the new definition). This works well, but some services are smarter and can be told to use a new set of options without restarting.

To take advantage of this capability, the resource agent must:

1. Accept the `reload` operation and perform any required actions. *The actions here depend completely on your application!*

   **Example 8.8. The DRBD agent’s logic for supporting reload**

   ```bash
   case $1 in
     start)
       drbd_start
     ;;
     stop)
       drbd_stop
     ;;
     reload)
       drbd_reload
     ;;
     monitor)
   ```
2. Advertise the **reload** operation in the **actions** section of its metadata

**Example 8.9. The DRBD Agent Advertising Support for the reload Operation**

```xml
<?xml version="1.0"?>
<!DOCTYPE resource-agent SYSTEM "ra-api-1.dtd">
<resource-agent name="drbd">
  <version>1.1</version>
  ...
  <actions>
    <action name="start" timeout="240" />
    <action name="reload" timeout="240" />
    ...
  </actions>
</resource-agent>
```

3. Advertise one or more parameters that can take effect using **reload**.

Any parameter with the **unique** set to 0 is eligible to be used in this way.

**Example 8.10. Parameter that can be changed using reload**

```xml
<parameter name="drbdconf" unique="0">
  <longdesc>Full path to the drbd.conf file.</longdesc>
  <shortdesc>Path to drbd.conf</shortdesc>
  <content type="string" default="${OCF_RESKEY_drbdconf_default}"/>
</parameter>
```

Once these requirements are satisfied, the cluster will automatically know to reload the resource (instead of restarting) when a non-unique field changes.

**Note**

Metadata will not be re-read unless the resource needs to be started. This may mean that the resource will be restarted the first time, even though you changed a parameter with **unique=0**.
If both a unique and non-unique field are changed simultaneously, the resource will still be restarted.
Advanced Resource Types

Table of Contents

9.1. Groups - A Syntactic Shortcut ............................................................................................. 75
  9.1.1. Group Properties ...................................................................................................... 76
  9.1.2. Group Options ......................................................................................................... 76
  9.1.3. Group Instance Attributes ......................................................................................... 76
  9.1.4. Group Contents ........................................................................................................ 76
  9.1.5. Group Constraints .................................................................................................... 77
  9.1.6. Group Stickiness ...................................................................................................... 77

9.2. Clones - Resources That Can Have Multiple Active Instances ............................................... 77
  9.2.1. Anonymous versus Unique Clones ............................................................................ 77
  9.2.2. Promotable clones .................................................................................................... 77
  9.2.3. Clone Properties ...................................................................................................... 78
  9.2.4. Clone Options .......................................................................................................... 78
  9.2.5. Clone Contents ........................................................................................................ 79
  9.2.6. Clone Instance Attributes .......................................................................................... 79
  9.2.7. Clone Constraints ..................................................................................................... 79
  9.2.8. Clone Stickiness ....................................................................................................... 82
  9.2.9. Clone Resource Agent Requirements ........................................................................ 82
  9.2.10. Monitoring Promotable Clone Resources ................................................................. 88
  9.2.11. Determining Which Instance is Promoted ................................................................. 89

9.3. Bundles - Isolated Environments .......................................................................................... 89
  9.3.1. Bundle Properties ..................................................................................................... 90
  9.3.2. Docker Properties ..................................................................................................... 90
  9.3.3. rkt Properties ........................................................................................................... 91
  9.3.4. Bundle Network Properties ....................................................................................... 92
  9.3.5. Bundle Storage Properties ........................................................................................ 93
  9.3.6. Bundle Primitive ....................................................................................................... 94
  9.3.7. Bundle Node Attributes ............................................................................................. 95
  9.3.8. Bundle Meta-Attributes ............................................................................................. 96
  9.3.9. Limitations of Bundles .............................................................................................. 96

9.1. Groups - A Syntactic Shortcut

One of the most common elements of a cluster is a set of resources that need to be located together, start sequentially, and stop in the reverse order. To simplify this configuration, we support the concept of groups.

Example 9.1. A group of two primitive resources

```xml
<group id="shortcut">
  <primitive id="Public-IP" class="ocf" type="IPaddr" provider="heartbeat">
    <instance_attributes id="params-public-ip">
      <nvpair id="public-ip-addr" name="ip" value="192.0.2.2"/>
    </instance_attributes>
  </primitive>
  <primitive id="Email" class="lsb" type="exim"/>
</group>
```
Chapter 9. Advanced Resource Types

Although the example above contains only two resources, there is no limit to the number of resources a group can contain. The example is also sufficient to explain the fundamental properties of a group:

- Resources are started in the order they appear in (Public-IP first, then Email)
- Resources are stopped in the reverse order to which they appear in (Email first, then Public-IP)

If a resource in the group can't run anywhere, then nothing after that is allowed to run, too.
- If Public-IP can't run anywhere, neither can Email;
- but if Email can't run anywhere, this does not affect Public-IP in any way

The group above is logically equivalent to writing:

```
<configuration>
  <resources>
    <primitive id="Public-IP" class="ocf" type="IPaddr" provider="heartbeat">
      <instance_attributes id="params-public-ip">
        <nvpair id="public-ip-addr" name="ip" value="192.0.2.2"/>
      </instance_attributes>
    </primitive>
    <primitive id="Email" class="lsb" type="exim"/>
  </resources>
  <constraints>
    <rsc_colocation id="xxx" rsc="Email" with-rsc="Public-IP" score="INFINITY"/>
    <rsc_order id="yyy" first="Public-IP" then="Email"/>
  </constraints>
</configuration>
```

Obviously as the group grows bigger, the reduced configuration effort can become significant.

Another (typical) example of a group is a DRBD volume, the filesystem mount, an IP address, and an application that uses them.

### 9.1.1. Group Properties

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>A unique name for the group</td>
</tr>
</tbody>
</table>

### 9.1.2. Group Options

Groups inherit the `priority`, `target-role`, and `is-managed` properties from primitive resources. See Section 4.4, “Resource Options” for information about those properties.

### 9.1.3. Group Instance Attributes

Groups have no instance attributes. However, any that are set for the group object will be inherited by the group's children.

### 9.1.4. Group Contents

Groups may only contain a collection of cluster resources (see Section 4.3, “Resource Properties”). To refer to a child of a group resource, just use the child's `id` instead of the group's.
9.1.5. Group Constraints

Although it is possible to reference a group's children in constraints, it is usually preferable to reference the group itself.

Example 9.3. Some constraints involving groups

```xml
<constraints>
  <rsc_location id="group-prefers-node1" rsc="shortcut" node="node1" score="500"/>
  <rsc_colocation id="webserver-with-group" rsc="Webserver" with-rsc="shortcut"/>
  <rsc_order id="start-group-then-webserver" first="Webserver" then="shortcut"/>
</constraints>
```

9.1.6. Group Stickiness

Stickiness, the measure of how much a resource wants to stay where it is, is additive in groups. Every active resource of the group will contribute its stickiness value to the group's total. So if the default resource stickiness is 100, and a group has seven members, five of which are active, then the group as a whole will prefer its current location with a score of 500.

9.2. Clones - Resources That Can Have Multiple Active Instances

Clone resources are resources that can have more than one copy active at the same time. This allows you, for example, to run a copy of a daemon on every node. You can clone any primitive or group resource.

9.2.1. Anonymous versus Unique Clones

A clone resource is configured to be either anonymous or globally unique.

Anonymous clones are the simplest. These behave completely identically everywhere they are running. Because of this, there can be only one instance of an anonymous clone active per node.

The instances of globally unique clones are distinct entities. All instances are launched identically, but one instance of the clone is not identical to any other instance, whether running on the same node or a different node. As an example, a cloned IP address can use special kernel functionality such that each instance handles a subset of requests for the same IP address.

9.2.2. Promotable clones

If a clone is promotable, its instances can perform a special role that Pacemaker will manage via the promote and demote actions of the resource agent.

---

1 Of course, the service must support running multiple instances.
Chapter 9. Advanced Resource Types

Services that support such a special role have various terms for the special role and the default role: primary and secondary, master and replica, controller and worker, etc. Pacemaker uses the terms *master* and *slave*, but is agnostic to what the service calls them or what they do.

All that Pacemaker cares about is that an instance comes up in the default role when started, and the resource agent supports the *promote* and *demote* actions to manage entering and exiting the special role.

### 9.2.3. Clone Properties

Table 9.2. Properties of a Clone Resource

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>A unique name for the clone</td>
</tr>
</tbody>
</table>

### 9.2.4. Clone Options

*Options* inherited from primitive resources: *priority*, *target-role*, *is-managed*

Table 9.3. Clone-specific configuration options

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>globally-unique</td>
<td>false</td>
<td>If true, each clone instance performs a distinct function</td>
</tr>
<tr>
<td>clone-max</td>
<td>number of nodes in cluster</td>
<td>The maximum number of clone instances that can be started across the entire cluster</td>
</tr>
<tr>
<td>clone-node-max</td>
<td>1</td>
<td>If globally-unique is true, the maximum number of clone instances that can be started on a single node</td>
</tr>
<tr>
<td>clone-min</td>
<td>0</td>
<td>Require at least this number of clone instances to be runnable before allowing resources depending on the clone to be runnable. A value of 0 means require all clone instances to be runnable.</td>
</tr>
<tr>
<td>notify</td>
<td>false</td>
<td>Call the resource agent's notify action for all active instances, before and after starting or stopping any clone instance. The resource agent must support this action. Allowed values: false, true</td>
</tr>
<tr>
<td>ordered</td>
<td>false</td>
<td>If true, clone instances must be started sequentially instead of in parallel. Allowed values: false, true</td>
</tr>
<tr>
<td>interleave</td>
<td>false</td>
<td>When this clone is ordered relative to another clone, if this option is false (the default), the ordering is relative to all instances of the other clone, whereas if this option is true, the ordering is relative only to instances on the same node. Allowed values: false, true</td>
</tr>
<tr>
<td>promotable</td>
<td>false</td>
<td>If true, clone instances can perform a special role that Pacemaker will manage via the resource agent's promote and demote actions. The resource agent</td>
</tr>
</tbody>
</table>

---

2 These are historical terms that will eventually be replaced, but the extensive use of them and the need for backward compatibility makes it a long process. You may see examples using a master tag instead of a clone tag with the promotable meta-attribute set to true; the master tag is supported, but deprecated, and will be removed in a future version. You may also see such services referred to as multi-state or stateful; these means the same thing as promotable.
9.2.5. Clone Contents

Clones must contain exactly one primitive or group resource.

Example 9.4. A clone that runs a web server on all nodes

```xml
<clone id="apache-clone">
  <primitive id="apache" class="lsb" type="apache">
    <operations>
      <op id="apache-monitor" name="monitor" interval="30"/>
    </operations>
  </primitive>
</clone>
```

Warning

You should never reference the name of a clone’s child (the primitive or group resource being cloned). If you think you need to do this, you probably need to re-evaluate your design.

9.2.6. Clone Instance Attributes

Clones have no instance attributes; however, any that are set here will be inherited by the clone’s child.

9.2.7. Clone Constraints

In most cases, a clone will have a single instance on each active cluster node. If this is not the case, you can indicate which nodes the cluster should preferentially assign copies to with resource location constraints. These constraints are written no differently from those for primitive resources except that the clone’s `id` is used.

Example 9.5. Some constraints involving clones

```xml
<constraints>
  <rsc_location id="clone-prefers-node1" rsc="apache-clone" node="node1" score="500"/>
  <rsc_colocation id="stats-with-clone" rsc="apache-stats" with="apache-clone"/>
  <rsc_order id="start-clone-then-stats" first="apache-clone" then="apache-stats"/>
</constraints>
```
Chapter 9. Advanced Resource Types

Ordering constraints behave slightly differently for clones. In the example above, **apache-stats** will wait until all copies of **apache-clone** that need to be started have done so before being started itself. Only if no copies can be started will **apache-stats** be prevented from being active. Additionally, the clone will wait for **apache-stats** to be stopped before stopping itself.

Colocation of a primitive or group resource with a clone means that the resource can run on any node with an active instance of the clone. The cluster will choose an instance based on where the clone is running and the resource's own location preferences.

Colocation between clones is also possible. If one clone **A** is colocated with another clone **B**, the set of allowed locations for **A** is limited to nodes on which **B** is (or will be) active. Placement is then performed normally.

### 9.2.7.1. Promotable Clone Constraints

For promotable clone resources, the **first-action** and/or **then-action** fields for ordering constraints may be set to **promote** or **demote** to constrain the master role, and colocation constraints may contain **rsc-role** and/or **with-rsc-role** fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>rsc-role</strong></td>
<td>Started</td>
<td>An additional attribute of colocation constraints that specifies the role that rsc must be in. Allowed values: <strong>Started, Master, Slave</strong>.</td>
</tr>
<tr>
<td><strong>with-rsc-role</strong></td>
<td>Started</td>
<td>An additional attribute of colocation constraints that specifies the role that with-rsc must be in. Allowed values: <strong>Started, Master, Slave</strong>.</td>
</tr>
</tbody>
</table>

**Example 9.6. Constraints involving promotable clone resources**

```xml
<constraints>
  <rsc_location id="db-prefers-node1" rsc="database" node="node1" score="500"/>
  <rsc_colocation id="backup-with-db-slave" rsc="backup"
    with-rsc="database" with-rsc-role="Slave"/>
  <rsc_colocation id="myapp-with-db-master" rsc="myApp"
    with-rsc="database" with-rsc-role="Master"/>
  <rsc_order id="start-db-before-backup" first="database" then="backup"/>
  <rsc_order id="promote-db-then-app" first="database" first-action="promote"
    then="myApp" then-action="start"/>
</constraints>
```

In the example above, **myApp** will wait until one of the database copies has been started and promoted to master before being started itself on the same node. Only if no copies can be promoted will **myApp** be prevented from being active. Additionally, the cluster will wait for **myApp** to be stopped before demoting the database.

Colocation of a primitive or group resource with a promotable clone resource means that it can run on any node with an active instance of the promotable clone resource that has the specified role (**master** or **slave**). In the example above, the cluster will choose a location based on where database is running as a **master**, and if there are multiple **master** instances it will also factor in **myApp**'s own location preferences when deciding which location to choose.
Colocation with regular clones and other promotable clone resources is also possible. In such cases, the set of allowed locations for the \texttt{rsc} clone is (after role filtering) limited to nodes on which the \texttt{with-rsc} promotable clone resource is (or will be) in the specified role. Placement is then performed as normal.

### 9.2.7.2. Using Promotable Clone Resources in Colocation Sets

Table 9.5. Additional colocation set options relevant to promotable clone resources

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>role</td>
<td>Started</td>
<td>The role that \textit{all members} of the set must be in. Allowed values: Started, Master, Slave.</td>
</tr>
</tbody>
</table>

In the following example B’s master must be located on the same node as A’s master. Additionally resources C and D must be located on the same node as A’s and B’s masters.

Example 9.7. Colocate C and D with A’s and B’s master instances

```xml
<constraints>
  <rsc_colocation id="coloc-1" score="INFINITY" >
    <resource_set id="colocated-set-example-1" sequential="true" role="Master">
      <resource_ref id="A"/>
      <resource_ref id="B"/>
    </resource_set>
    <resource_set id="colocated-set-example-2" sequential="true">
      <resource_ref id="C"/>
      <resource_ref id="D"/>
    </resource_set>
  </rsc_colocation>
</constraints>
```

### 9.2.7.3. Using Promotable Clone Resources in Ordered Sets

Table 9.6. Additional ordered set options relevant to promotable clone resources

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>value of first-action</td>
<td>An additional attribute of ordering constraint sets that specifies the action that applies to \textit{all members} of the set. Allowed values: start, stop, promote, demote.</td>
</tr>
</tbody>
</table>

Example 9.8. Start C and D after first promoting A and B

```xml
<constraints>
  <rsc_order id="order-1" score="INFINITY" >
    <resource_set id="ordered-set-1" sequential="true" action="promote">
      <resource_ref id="A"/>
      <resource_ref id="B"/>
    </resource_set>
    <resource_set id="ordered-set-2" sequential="true" action="start">
      <resource_ref id="C"/>
      <resource_ref id="D"/>
    </resource_set>
  </rsc_order>
</constraints>
```

In the above example, B cannot be promoted to a master role until A has been promoted. Additionally, resources C and D must wait until A and B have been promoted before they can start.
9.2.8. Clone Stickiness

To achieve a stable allocation pattern, clones are slightly sticky by default. If no value for `resource-stickiness` is provided, the clone will use a value of 1. Being a small value, it causes minimal disturbance to the score calculations of other resources but is enough to prevent Pacemaker from needlessly moving copies around the cluster.

**Note**

For globally unique clones, this may result in multiple instances of the clone staying on a single node, even after another eligible node becomes active (for example, after being put into standby mode then made active again). If you do not want this behavior, specify a `resource-stickiness` of 0 for the clone temporarily and let the cluster adjust, then set it back to 1 if you want the default behavior to apply again.

**Important**

If `resource-stickiness` is set in the `rsc_defaults` section, it will apply to clone instances as well. This means an explicit `resource-stickiness` of 0 in `rsc_defaults` works differently from the implicit default used when `resource-stickiness` is not specified.

9.2.9. Clone Resource Agent Requirements

Any resource can be used as an anonymous clone, as it requires no additional support from the resource agent. Whether it makes sense to do so depends on your resource and its resource agent.

9.2.9.1. Resource Agent Requirements for Globally Unique Clones

Globally unique clones require additional support in the resource agent. In particular, it must only respond with `{OCF_SUCCESS}` if the node has that exact instance active. All other probes for instances of the clone should result in `{OCF_NOT_RUNNING}` (or one of the other OCF error codes if they are failed).

Individual instances of a clone are identified by appending a colon and a numerical offset, e.g. `apache:2`.

Resource agents can find out how many copies there are by examining the `OCF_RESKEY_CRM_meta_clone_max` environment variable and which instance it is by examining `OCF_RESKEY_CRM_meta_clone`.

The resource agent must not make any assumptions (based on `OCF_RESKEY_CRM_meta_clone`) about which numerical instances are active. In particular, the list of active copies will not always be an unbroken sequence, nor always start at 0.
9.2.9.2. Resource Agent Requirements for Promotable Clones

Promotable clone resources require two extra actions, demote and promote, which are responsible for changing the state of the resource. Like start and stop, they should return $\texttt{OCF\_SUCCESS}$ if they completed successfully or a relevant error code if they did not.

The states can mean whatever you wish, but when the resource is started, it must come up in the mode called slave. From there the cluster will decide which instances to promote to master.

In addition to the clone requirements for monitor actions, agents must also accurately report which state they are in. The cluster relies on the agent to report its status (including role) accurately and does not indicate to the agent what role it currently believes it to be in.

Table 9.7. Role implications of OCF return codes

<table>
<thead>
<tr>
<th>Monitor Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCF_NOT_RUNNING</td>
<td>Stopped</td>
</tr>
<tr>
<td>OCF_SUCCESS</td>
<td>Running (Slave)</td>
</tr>
<tr>
<td>OCF_RUNNING_MASTER</td>
<td>Running (Master)</td>
</tr>
<tr>
<td>OCF_FAILED_MASTER</td>
<td>Failed (Master)</td>
</tr>
<tr>
<td>Other</td>
<td>Failed (Slave)</td>
</tr>
</tbody>
</table>

9.2.9.3. Clone Notifications

If the clone has the notify meta-attribute set to true, and the resource agent supports the notify action, Pacemaker will call the action when appropriate, passing a number of extra variables which, when combined with additional context, can be used to calculate the current state of the cluster and what is about to happen to it.

Table 9.8. Environment variables supplied with Clone notify actions

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCF_RESKEY_CRM_meta_notify_type</td>
<td>Allowed values: pre, post</td>
</tr>
<tr>
<td>OCF_RESKEY_CRM_meta_notify_operation</td>
<td>Allowed values: start, stop</td>
</tr>
<tr>
<td>OCF_RESKEY_CRM_meta_notify_start_resource</td>
<td>Resources to be started</td>
</tr>
<tr>
<td>OCF_RESKEY_CRM_meta_notify_stop_resource</td>
<td>Resources to be stopped</td>
</tr>
<tr>
<td>OCF_RESKEY_CRM_meta_notify_active_resource</td>
<td>Resources that are running</td>
</tr>
<tr>
<td>OCF_RESKEY_CRM_meta_notify_inactive_resource</td>
<td>Resources that are not running</td>
</tr>
<tr>
<td>OCF_RESKEY_CRM_meta_notify_start_uname</td>
<td>Nodes on which resources will be started</td>
</tr>
<tr>
<td>OCF_RESKEY_CRM_meta_notify_stop_uname</td>
<td>Nodes on which resources will be stopped</td>
</tr>
<tr>
<td>OCF_RESKEY_CRM_meta_notify_active_uname</td>
<td>Nodes on which resources are running</td>
</tr>
</tbody>
</table>

The variables come in pairs, such as OCF\_RESKEY\_CRM\_meta\_notify\_start\_resource and OCF\_RESKEY\_CRM\_meta\_notify\_start\_uname, and should be treated as an array of whitespace-separated elements.

OCF\_RESKEY\_CRM\_meta\_notify\_inactive\_resource is an exception, as the matching uname variable does not exist since inactive resources are not running on any node.

Thus, in order to indicate that clone:0 will be started on sles-1, clone:2 will be started on sles-3, and clone:3 will be started on sles-2, the cluster would set:
### Example 9.9. Notification variables

```plaintext
OCF_RESKEY_CRM_meta_notify_start_resource="clone:0 clone:2 clone:3"
OCF_RESKEY_CRM_meta_notify_start_uname="sles-1 sles-3 sles-2"
```

### Note

Pacemaker will log but otherwise ignore failures of notify actions.

#### 9.2.9.4. Interpretation of Notification Variables

**Pre-notification (stop):**
- Active resources: `$OCF_RESKEY_CRM_meta_notify_active_resource`
- Inactive resources: `$OCF_RESKEY_CRM_meta_notify_inactive_resource`
- Resources to be started: `$OCF_RESKEY_CRM_meta_notify_start_resource`
- Resources to be stopped: `$OCF_RESKEY_CRM_meta_notify_stop_resource`

**Post-notification (stop) / Pre-notification (start):**
- Active resources:
  - `$OCF_RESKEY_CRM_meta_notify_active_resource`
  - minus `$OCF_RESKEY_CRM_meta_notify_stop_resource`
- Inactive resources:
  - `$OCF_RESKEY_CRM_meta_notify_inactive_resource`
  - plus `$OCF_RESKEY_CRM_meta_notify_stop_resource`
  - Resources that were started: `$OCF_RESKEY_CRM_meta_notify_start_resource`
  - Resources that were stopped: `$OCF_RESKEY_CRM_meta_notify_stop_resource`

**Post-notification (start):**
- Active resources:
  - `$OCF_RESKEY_CRM_meta_notify_active_resource`
  - minus `$OCF_RESKEY_CRM_meta_notify_stop_resource`
  - plus `$OCF_RESKEY_CRM_meta_notify_start_resource`
- Inactive resources:
  - `$OCF_RESKEY_CRM_meta_notify_inactive_resource`
  - plus `$OCF_RESKEY_CRM_meta_notify_stop_resource`
  - minus `$OCF_RESKEY_CRM_meta_notify_start_resource`
• Resources that were started: $OCF\_RESKEY\_CRM\_meta\_notify\_start\_resource$

• Resources that were stopped: $OCF\_RESKEY\_CRM\_meta\_notify\_stop\_resource$

### 9.2.9.5. Extra Notifications for Promotable Clones

Table 9.9. Extra environment variables supplied for promotable clones

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCF_RESKEY_CRM_meta_notify_master_resource</td>
<td>Resources that are running in <strong>Master</strong> mode</td>
</tr>
<tr>
<td>OCF_RESKEY_CRM_meta_notify_slave_resource</td>
<td>Resources that are running in <strong>Slave</strong> mode</td>
</tr>
<tr>
<td>OCF_RESKEY_CRM_meta_notify_promote_resource</td>
<td>Resources to be promoted</td>
</tr>
<tr>
<td>OCF_RESKEY_CRM_meta_notify_demote_resource</td>
<td>Resources to be demoted</td>
</tr>
<tr>
<td>OCF_RESKEY_CRM_meta_notify_promote_uname</td>
<td>Nodes on which resources will be promoted</td>
</tr>
<tr>
<td>OCF_RESKEY_CRM_meta_notify_demote_uname</td>
<td>Nodes on which resources will be demoted</td>
</tr>
<tr>
<td>OCF_RESKEY_CRM_meta_notify_master_uname</td>
<td>Nodes on which resources are running in <strong>Master</strong> mode</td>
</tr>
<tr>
<td>OCF_RESKEY_CRM_meta_notify_slave_uname</td>
<td>Nodes on which resources are running in <strong>Slave</strong> mode</td>
</tr>
</tbody>
</table>

### 9.2.9.6. Interpretation of Promotable Notification Variables

**Pre-notification (demote):**
- **Active** resources: $OCF\_RESKEY\_CRM\_meta\_notify\_active\_resource$
- **Master** resources: $OCF\_RESKEY\_CRM\_meta\_notify\_master\_resource$
- **Slave** resources: $OCF\_RESKEY\_CRM\_meta\_notify\_slave\_resource$
- **Inactive** resources: $OCF\_RESKEY\_CRM\_meta\_notify\_inactive\_resource$
- Resources to be started: $OCF\_RESKEY\_CRM\_meta\_notify\_start\_resource$
- Resources to be promoted: $OCF\_RESKEY\_CRM\_meta\_notify\_promote\_resource$
- Resources to be demoted: $OCF\_RESKEY\_CRM\_meta\_notify\_demote\_resource$
- Resources to be stopped: $OCF\_RESKEY\_CRM\_meta\_notify\_stop\_resource$

**Post-notification (demote) / Pre-notification (stop):**
- **Active** resources: $OCF\_RESKEY\_CRM\_meta\_notify\_active\_resource$
- **Master** resources: $OCF\_RESKEY\_CRM\_meta\_notify\_master\_resource$
  - minus $OCF\_RESKEY\_CRM\_meta\_notify\_demote\_resource$
- **Slave** resources: $OCF\_RESKEY\_CRM\_meta\_notify\_slave\_resource$
Chapter 9. Advanced Resource Types

- Inactive resources: $OCF_RESKEY_CRM_meta_notify_inactive_resource
- Resources to be started: $OCF_RESKEY_CRM_meta_notify_start_resource
- Resources to be promoted: $OCF_RESKEY_CRM_meta_notify_promote_resource
- Resources to be demoted: $OCF_RESKEY_CRM_meta_notify_demote_resource
- Resources to be stopped: $OCF_RESKEY_CRM_meta_notify_stop_resource
- Resources that were demoted: $OCF_RESKEY_CRM_meta_notify_demote_resource

Post-notification (stop) / Pre-notification (start)

- **Active** resources:
  - $OCF_RESKEY_CRM_meta_notify_active_resource
  - minus $OCF_RESKEY_CRM_meta_notify_stop_resource
- **Master** resources:
  - $OCF_RESKEY_CRM_meta_notify_master_resource
  - minus $OCF_RESKEY_CRM_meta_notify_demote_resource
- **Slave** resources:
  - $OCF_RESKEY_CRM_meta_notify_slave_resource
  - minus $OCF_RESKEY_CRM_meta_notify_stop_resource
- Inactive resources:
  - $OCF_RESKEY_CRM_meta_notify_inactive_resource
  - plus $OCF_RESKEY_CRM_meta_notify_stop_resource
  - Resources to be started: $OCF_RESKEY_CRM_meta_notify_start_resource
  - Resources to be promoted: $OCF_RESKEY_CRM_meta_notify_promote_resource
  - Resources to be demoted: $OCF_RESKEY_CRM_meta_notify_demote_resource
  - Resources to be stopped: $OCF_RESKEY_CRM_meta_notify_stop_resource
  - Resources that were demoted: $OCF_RESKEY_CRM_meta_notify_demote_resource
  - Resources that were stopped: $OCF_RESKEY_CRM_meta_notify_stop_resource

Post-notification (start) / Pre-notification (promote)

- **Active** resources:
  - $OCF_RESKEY_CRM_meta_notify_active_resource
  - minus $OCF_RESKEY_CRM_meta_notify_stop_resource
  - plus $OCF_RESKEY_CRM_meta_notify_start_resource
- **Master** resources:
• $OCF\_RESKEY\_CRM\_meta\_notify\_master\_resource
• minus $OCF\_RESKEY\_CRM\_meta\_notify\_demote\_resource

• **Slave** resources:
  • $OCF\_RESKEY\_CRM\_meta\_notify\_slave\_resource
  • minus $OCF\_RESKEY\_CRM\_meta\_notify\_stop\_resource
  • plus $OCF\_RESKEY\_CRM\_meta\_notify\_start\_resource

• Inactive resources:
  • $OCF\_RESKEY\_CRM\_meta\_notify\_inactive\_resource
  • plus $OCF\_RESKEY\_CRM\_meta\_notify\_stop\_resource
  • minus $OCF\_RESKEY\_CRM\_meta\_notify\_start\_resource

• Resources to be started: $OCF\_RESKEY\_CRM\_meta\_notify\_start\_resource
• Resources to be promoted: $OCF\_RESKEY\_CRM\_meta\_notify\_promote\_resource
• Resources to be demoted: $OCF\_RESKEY\_CRM\_meta\_notify\_demote\_resource
• Resources to be stopped: $OCF\_RESKEY\_CRM\_meta\_notify\_stop\_resource
• Resources that were started: $OCF\_RESKEY\_CRM\_meta\_notify\_start\_resource
• Resources that were demoted: $OCF\_RESKEY\_CRM\_meta\_notify\_demote\_resource
• Resources that were stopped: $OCF\_RESKEY\_CRM\_meta\_notify\_stop\_resource

**Post-notification (promote)**

• **Active** resources:
  • $OCF\_RESKEY\_CRM\_meta\_notify\_active\_resource
  • minus $OCF\_RESKEY\_CRM\_meta\_notify\_stop\_resource
  • plus $OCF\_RESKEY\_CRM\_meta\_notify\_start\_resource

• **Master** resources:
  • $OCF\_RESKEY\_CRM\_meta\_notify\_master\_resource
  • minus $OCF\_RESKEY\_CRM\_meta\_notify\_demote\_resource
  • plus $OCF\_RESKEY\_CRM\_meta\_notify\_promote\_resource

• **Slave** resources:
  • $OCF\_RESKEY\_CRM\_meta\_notify\_slave\_resource
  • minus $OCF\_RESKEY\_CRM\_meta\_notify\_stop\_resource
  • plus $OCF\_RESKEY\_CRM\_meta\_notify\_start\_resource
• minus $OCF\_RESKEY\_CRM\_meta\_notify\_promote\_resource

• Inactive resources:
  • $OCF\_RESKEY\_CRM\_meta\_notify\_inactive\_resource
  • plus $OCF\_RESKEY\_CRM\_meta\_notify\_stop\_resource
  • minus $OCF\_RESKEY\_CRM\_meta\_notify\_start\_resource

• Resources to be started: $OCF\_RESKEY\_CRM\_meta\_notify\_start\_resource

• Resources to be promoted: $OCF\_RESKEY\_CRM\_meta\_notify\_promote\_resource

• Resources to be demoted: $OCF\_RESKEY\_CRM\_meta\_notify\_demote\_resource

• Resources to be stopped: $OCF\_RESKEY\_CRM\_meta\_notify\_stop\_resource

• Resources that were started: $OCF\_RESKEY\_CRM\_meta\_notify\_start\_resource

• Resources that were promoted: $OCF\_RESKEY\_CRM\_meta\_notify\_promote\_resource

• Resources that were demoted: $OCF\_RESKEY\_CRM\_meta\_notify\_demote\_resource

• Resources that were stopped: $OCF\_RESKEY\_CRM\_meta\_notify\_stop\_resource

9.2.10. Monitoring Promotable Clone Resources

The usual monitor actions are insufficient to monitor a promotable clone resource, because Pacemaker needs to verify not only that the resource is active, but also that its actual role matches its intended one.

Define two monitoring actions: the usual one will cover the slave role, and an additional one with role="master" will cover the master role.

Example 9.10. Monitoring both states of a promotable clone resource

```xml
<clone id="myMasterRsc">
  <meta_attributes id="myMasterRsc-meta">
    <nvpair name="promotable" value="true"/>
  </meta_attributes>
  <primitive id="myRsc" class="ocf" type="myApp" provider="myCorp">
    <operations>
      <op id="public-ip-slave-check" name="monitor" interval="60"/>
      <op id="public-ip-master-check" name="monitor" interval="61" role="Master"/>
    </operations>
  </primitive>
</clone>
```
Important

It is crucial that every monitor operation has a different interval! Pacemaker currently differentiates between operations only by resource and interval; so if (for example) a promotable clone resource had the same monitor interval for both roles, Pacemaker would ignore the role when checking the status — which would cause unexpected return codes, and therefore unnecessary complications.

9.2.11. Determining Which Instance is Promoted

Pacemaker can choose a promotable clone instance to be promoted in one of two ways:

- Promotion scores: These are node attributes set via the `crm_master` utility, which generally would be called by the resource agent's start action if it supports promotable clones. This tool automatically detects both the resource and host, and should be used to set a preference for being promoted. Based on this, `promoted-max`, and `promoted-node-max`, the instance(s) with the highest preference will be promoted.

- Constraints: Location constraints can indicate which nodes are most preferred as masters.

Example 9.11. Explicitly preferring node1 to be promoted to master

```xml
<rsc_location id="master-location" rsc="myMasterRsc">
  <rule id="master-rule" score="100" role="Master">
    <expression id="master-exp" attribute="#uname" operation="eq" value="node1"/>
  </rule>
</rsc_location>
```

9.3. Bundles - Isolated Environments

Pacemaker supports a special syntax for launching a `container` with any infrastructure it requires: the `bundle`.

Pacemaker bundles support `Docker` and `rkt` container technologies.  

Example 9.12. A bundle for a containerized web server

```xml
<bundle id="httpd-bundle">
  <docker image="pcmk:http" replicas="3"/>
  <network ip-range-start="192.168.122.131" host-netmask="24" host-interface="eth0">
    <port-mapping id="httpd-port" port="80"/>
  </network>
</bundle>
```

---

3 https://en.wikipedia.org/wiki/Operating-system-level_virtualization
4 https://www.docker.com/
5 https://coreos.com/rkt/
6 Docker is a trademark of Docker, Inc. No endorsement by or association with Docker, Inc. is implied.
9.3.1. Bundle Properties

Table 9.10. Properties of a Bundle

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>A unique name for the bundle (required)</td>
</tr>
<tr>
<td>description</td>
<td>Arbitrary text (not used by Pacemaker)</td>
</tr>
</tbody>
</table>

A bundle must contain exactly one `<docker>` or `<rkt>` element.

9.3.2. Docker Properties

Before configuring a Docker bundle in Pacemaker, the user must install Docker and supply a fully configured Docker image on every node allowed to run the bundle.

Pacemaker will create an implicit `ocf:heartbeat:docker` resource to manage a bundle’s Docker container. The user must ensure that resource agent is installed on every node allowed to run the bundle.

Table 9.11. Properties of a Bundle’s Docker Element

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td></td>
<td>Docker image tag (required)</td>
</tr>
<tr>
<td>replicas</td>
<td></td>
<td>A positive integer specifying the number of container instances to launch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value of <code>promoted-max</code> if that is positive, else 1</td>
</tr>
<tr>
<td>replicas-per-host</td>
<td>1</td>
<td>A positive integer specifying the number of container instances allowed to run on a single node</td>
</tr>
<tr>
<td>promoted-max</td>
<td>0</td>
<td>A non-negative integer that, if positive, indicates that the containerized service should be treated as a promotable service, with this many replicas allowed to run the service in the master role</td>
</tr>
</tbody>
</table>
### 9.3.3. rkt Properties

Before configuring a rkt bundle in Pacemaker, the user must install rkt and supply a fully configured container image on every node allowed to run the bundle.

Pacemaker will create an implicit `ocf:heartbeat:rkt` resource to manage a bundle’s rkt container. The user must ensure that resource agent is installed on every node allowed to run the bundle.

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td></td>
<td>Container image tag (required)</td>
</tr>
<tr>
<td>replicas</td>
<td>Value of <code>promoted-max</code> if that is positive, else 1</td>
<td>A positive integer specifying the number of container instances to launch</td>
</tr>
<tr>
<td>replicas-per-host</td>
<td>1</td>
<td>A positive integer specifying the number of container instances allowed to run on a single node</td>
</tr>
<tr>
<td>promoted-max</td>
<td>0</td>
<td>A non-negative integer that, if positive, indicates that the containerized service should be treated as a promotable service, with this many replicas allowed to run the service in the master role</td>
</tr>
<tr>
<td>network</td>
<td></td>
<td>If specified, this will be passed to <code>rkt run</code> as the network setting for the rkt container.</td>
</tr>
</tbody>
</table>

---

For backward compatibility, `masters` is accepted as an alias for `promoted-max`, but is deprecated since 2.0.0, and support for it will be removed in a future version.

---

### Table 9.12. Properties of a Bundle’s rkt Element

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>network</td>
<td></td>
<td>If specified, this will be passed to <code>docker run</code> as the network setting for the Docker container.</td>
</tr>
<tr>
<td>run-command</td>
<td>/usr/sbin/pacemaker-remoted if bundle contains a primitive, otherwise none</td>
<td>This command will be run inside the container when launching it (&quot;PID 1&quot;). If the bundle contains a <em>primitive</em>, this command <em>must</em> start pacemaker-remoted (but could, for example, be a script that does other stuff, too). If the container image has a pre-2.0.0 version of Pacemaker, set this to <code>/usr/sbin/pacemaker_remoted</code> (note the underbar instead of dash).</td>
</tr>
<tr>
<td>options</td>
<td>Extra command-line options to pass to <code>docker run</code></td>
<td>Extra command-line options to pass to <code>docker run</code></td>
</tr>
</tbody>
</table>

---

7 https://docs.docker.com/engine/reference/run/#network-settings
### Chapter 9. Advanced Resource Types

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>run-command</td>
<td>/usr/sbin/pacemaker-remoted if bundle contains a primitive, otherwise none</td>
<td>This command will be run inside the container when launching it (&quot;PID 1&quot;). If the bundle contains a primitive, this command must start pacemaker-remoted (but could, for example, be a script that does other stuff, too). If the container image has a pre-2.0.0 version of Pacemaker, set this to /usr/sbin/pacemaker_remoted (note the underbar instead of dash).</td>
</tr>
<tr>
<td>options</td>
<td>Extra command-line options to pass to rkt run</td>
<td></td>
</tr>
</tbody>
</table>

For backward compatibility, masters is accepted as an alias for promoted-max, but is deprecated since 2.0.0, and support for it will be removed in a future version.

#### 9.3.4. Bundle Network Properties

A bundle may optionally contain one `<network>` element.

**Table 9.13. Properties of a Bundle’s Network Element**

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add-host</td>
<td>TRUE</td>
<td>If TRUE, and ip-range-start is used, Pacemaker will automatically ensure that /etc/hosts inside the containers has entries for each replica name and its assigned IP.</td>
</tr>
<tr>
<td>ip-range-start</td>
<td></td>
<td>If specified, Pacemaker will create an implicit ocf:heartbeat:IPaddr2 resource for each container instance, starting with this IP address, using up to replicas sequential addresses. These addresses can be used from the host's network to reach the service inside the container, though it is not visible within the container itself. Only IPv4 addresses are currently supported.</td>
</tr>
<tr>
<td>host-netmask</td>
<td>32</td>
<td>If ip-range-start is specified, the IP addresses are created with this CIDR netmask (as a number of bits).</td>
</tr>
<tr>
<td>host-interface</td>
<td></td>
<td>If ip-range-start is specified, the IP addresses are created on this host interface (by default, it will be determined from the IP address).</td>
</tr>
<tr>
<td>control-port</td>
<td>3121</td>
<td>If the bundle contains a primitive, the cluster will use this integer TCP port for communication with Pacemaker Remote inside the container. Changing this is useful when the container is unable to listen on the default port, for example, when the container uses the host's network rather than ip-range-start (in which case replicas-per-host must be 1), or when the bundle may run on a Pacemaker Remote node that is already listening on the default port. Any PCMK_remote_port environment variable</td>
</tr>
</tbody>
</table>
Note

Replicas are named by the bundle id plus a dash and an integer counter starting with zero. For example, if a bundle named `httpd-bundle` has `replicas=2`, its containers will be named `httpd-bundle-0` and `httpd-bundle-1`.

Additionally, a `<network>` element may optionally contain one or more `<port-mapping>` elements.

Table 9.14. Properties of a Bundle’s Port-Mapping Element

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td></td>
<td>A unique name for the port mapping (required)</td>
</tr>
<tr>
<td>port</td>
<td></td>
<td>If this is specified, connections to this TCP port number on the host network (on the container’s assigned IP address, if <code>ip-range-start</code> is specified) will be forwarded to the container network. Exactly one of <code>port</code> or <code>range</code> must be specified in a port-mapping.</td>
</tr>
<tr>
<td>internal-port</td>
<td>value of port</td>
<td>If <code>port</code> and this are specified, connections to <code>port</code> on the host’s network will be forwarded to this port on the container network.</td>
</tr>
<tr>
<td>range</td>
<td></td>
<td>If this is specified, connections to these TCP port numbers (expressed as <code>first_port-last_port</code>) on the host network (on the container’s assigned IP address, if <code>ip-range-start</code> is specified) will be forwarded to the same ports in the container network. Exactly one of <code>port</code> or <code>range</code> must be specified in a port-mapping.</td>
</tr>
</tbody>
</table>

Note

If the bundle contains a `primitive`, Pacemaker will automatically map the `control-port`, so it is not necessary to specify that port in a `port-mapping`.

9.3.5. Bundle Storage Properties

A bundle may optionally contain one `<storage>` element. A `<storage>` element has no properties of its own, but may contain one or more `<storage-mapping>` elements.

Table 9.15. Properties of a Bundle’s Storage-Mapping Element

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td></td>
<td>A unique name for the storage mapping (required)</td>
</tr>
</tbody>
</table>
### Field | Default | Description
--- | --- | ---
source-dir | | The absolute path on the host's filesystem that will be mapped into the container. Exactly one of source-dir and source-dir-root must be specified in a storage-mapping.
source-dir-root | | The start of a path on the host's filesystem that will be mapped into the container, using a different subdirectory on the host for each container instance. The subdirectory will be named the same as the replica name. Exactly one of source-dir and source-dir-root must be specified in a storage-mapping.
target-dir | | The path name within the container where the host storage will be mapped (required)
options | | File system mount options to use when mapping the storage

**Note**

Pacemaker does not define the behavior if the source directory does not already exist on the host. However, it is expected that the container technology and/or its resource agent will create the source directory in that case.

**Note**

If the bundle contains a primitive, Pacemaker will automatically map the equivalent of source-dir=/etc/pacemaker/authkey target-dir=/etc/pacemaker/authkey and source-dir-root=/var/log/pacemaker/bundles target-dir=/var/log into the container, so it is not necessary to specify those paths in a storage-mapping.

**Important**

The PCMK_authkey_location environment variable must not be set to anything other than the default of /etc/pacemaker/authkey on any node in the cluster.

### 9.3.6. Bundle Primitive

A bundle may optionally contain one `<primitive>` resource (see Section 4.1, "What is a Cluster Resource?"). The primitive may have operations, instance attributes and meta-attributes defined, as usual.

If a bundle contains a primitive resource, the container image must include the Pacemaker Remote daemon, and at least one of `ip-range-start` or `control-port` must be configured in the
bundle. Pacemaker will create an implicit `ocf:pacemaker:remote` resource for the connection, launch Pacemaker Remote within the container, and monitor and manage the primitive resource via Pacemaker Remote.

If the bundle has more than one container instance (replica), the primitive resource will function as an implicit clone (see Section 9.2, “Clones - Resources That Can Have Multiple Active Instances”) — a promotable clone if the bundle has `masters` greater than zero (see Section 9.2.2, “Promotable clones”).

### Important

Containers in bundles with a primitive must have an accessible networking environment, so that Pacemaker on the cluster nodes can contact Pacemaker Remote inside the container. For example, the Docker option `--net=none` should not be used with a primitive. The default (using a distinct network space inside the container) works in combination with `ip-range-start`. If the Docker option `--net=host` is used (making the container share the host's network space), a unique control-port should be specified for each bundle. Any firewall must allow access to the control-port.

9.3.7. Bundle Node Attributes

If the bundle has a primitive, the primitive’s resource agent may want to set node attributes such as promotion scores. However, with containers, it is not apparent which node should get the attribute.

If the container uses shared storage that is the same no matter which node the container is hosted on, then it is appropriate to use the promotion score on the bundle node itself.

On the other hand, if the container uses storage exported from the underlying host, then it may be more appropriate to use the promotion score on the underlying host.

Since this depends on the particular situation, the `container-attribute-target` resource meta-attribute allows the user to specify which approach to use. If it is set to `host`, then user-defined node attributes will be checked on the underlying host. If it is anything else, the local node (in this case the bundle node) is used as usual.

This only applies to user-defined attributes; the cluster will always check the local node for cluster-defined attributes such as `#uname`.

If `container-attribute-target` is `host`, the cluster will pass additional environment variables to the primitive’s resource agent that allow it to set node attributes appropriately: `CRM_meta_container_attribute_target` (identical to the meta-attribute value) and `CRM_meta_physical_host` (the name of the underlying host).

### Note

When called by a resource agent, the `attrd_updater` and `crm_attribute` commands will automatically check those environment variables and set attributes appropriately.
Chapter 9. Advanced Resource Types

9.3.8. Bundle Meta-Attributes
Any meta-attribute set on a bundle will be inherited by the bundle’s primitive and any resources implicitly created by Pacemaker for the bundle.

This includes options such as priority, target-role, and is-managed. See Section 4.4, “Resource Options” for more information.

9.3.9. Limitations of Bundles
Restarting pacemaker while a bundle is unmanaged or the cluster is in maintenance mode may cause the bundle to fail.

Bundles may not be explicitly cloned or included in groups. This includes the bundle’s primitive and any resources implicitly created by Pacemaker for the bundle. (If replicas is greater than 1, the bundle will behave like a clone implicitly.)

Bundles do not have instance attributes, utilization attributes, or operations, though a bundle’s primitive may have them.

A bundle with a primitive can run on a Pacemaker Remote node only if the bundle uses a distinct control-port.
Chapter 10.

Reusing Parts of the Configuration

Table of Contents

10.1. Reusing Resource Definitions ................................................................. 97
  10.1.1. Configuring Resources with Templates .............................................. 97
  10.1.2. Using Templates in Constraints ....................................................... 99
  10.1.3. Using Templates in Resource Sets .................................................... 99
10.2. Reusing Rules, Options and Sets of Operations ...................................... 100
10.3. Tagging Configuration Elements .......................................................... 101
  10.3.1. Configuring Tags ............................................................................. 102
  10.3.2. Using Tags in Constraints and Resource Sets .................................. 102

Pacemaker provides multiple ways to simplify the configuration XML by reusing parts of it in multiple places.

Besides simplifying the XML, this also allows you to manipulate multiple configuration elements with a single reference.

10.1. Reusing Resource Definitions

If you want to create lots of resources with similar configurations, defining a resource template simplifies the task. Once defined, it can be referenced in primitives or in certain types of constraints.

10.1.1. Configuring Resources with Templates

The primitives referencing the template will inherit all meta-attributes, instance attributes, utilization attributes and operations defined in the template. And you can define specific attributes and operations for any of the primitives. 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.

Hence, resource templates help to reduce the amount of configuration work. If any changes are needed, they can be done to the template definition and will take effect globally in all resource definitions referencing that template.

Resource templates have a syntax similar to that of primitives.

Example 10.1. Resource template for a migratable Xen virtual machine

```xml
<template id="vm-template" class="ocf" provider="heartbeat" type="Xen">
  <meta_attributes id="vm-template-meta_attributes">
    <nvpair id="vm-template-meta_attributes-allow-migrate" name="allow-migrate" value="true"/>
  </meta_attributes>
  <utilization id="vm-template-utilization">
    <nvpair id="vm-template-utilization-memory" name="memory" value="512"/>
  </utilization>
  <operations>
    <op id="vm-template-monitor-15s" interval="15s" name="monitor" timeout="60s"/>
    <op id="vm-template-start-0" interval="0" name="start" timeout="60s"/>
  </operations>
</template>
```
Chapter 10. Reusing Parts of the Configuration

Once you define a resource template, you can use it in primitives by specifying the `template` property.

Example 10.2. Xen primitive resource using a resource template

```xml
<primitive id="vm1" template="vm-template">
    <instance_attributes id="vm1-instance_attributes">
        <nvpair id="vm1-instance_attributes-name" name="name" value="vm1"/>
        <nvpair id="vm1-instance_attributes-xmfile" name="xmfile" value="/etc/xen/shared-vm/vm1"/>
    </instance_attributes>
</primitive>
```

In the example above, the new primitive `vm1` will inherit everything from `vm-template`. For example, the equivalent of the above two examples would be:

Example 10.3. Equivalent Xen primitive resource not using a resource template

```xml
<primitive id="vm1" class="ocf" provider="heartbeat" type="Xen">
    <meta_attributes id="vm-template-meta_attributes">
        <nvpair id="vm-template-meta_attributes-allow-migrate" name="allow-migrate" value="true"/>
    </meta_attributes>
    <utilization id="vm-template-utilization">
        <nvpair id="vm-template-utilization-memory" name="memory" value="512"/>
    </utilization>
    <operations>
        <op id="vm-template-monitor-15s" interval="15s" name="monitor" timeout="60s"/>
        <op id="vm-template-start-0" interval="0" name="start" timeout="60s"/>
    </operations>
    <instance_attributes id="vm1-instance_attributes">
        <nvpair id="vm1-instance_attributes-name" name="name" value="vm1"/>
        <nvpair id="vm1-instance_attributes-xmfile" name="xmfile" value="/etc/xen/shared-vm/vm1"/>
    </instance_attributes>
</primitive>
```

If you want to overwrite some attributes or operations, add them to the particular primitive’s definition.

Example 10.4. Xen resource overriding template values

```xml
<primitive id="vm2" template="vm-template">
    <meta_attributes id="vm2-meta_attributes">
        <nvpair id="vm2-meta_attributes-allow-migrate" name="allow-migrate" value="false"/>
    </meta_attributes>
    <utilization id="vm2-utilization">
        <nvpair id="vm2-utilization-memory" name="memory" value="1024"/>
    </utilization>
    <instance_attributes id="vm2-instance_attributes">
        <nvpair id="vm2-instance_attributes-name" name="name" value="vm2"/>
        <nvpair id="vm2-instance_attributes-xmfile" name="xmfile" value="/etc/xen/shared-vm/vm2"/>
    </instance_attributes>
    <operations>
        <op id="vm2-monitor-30s" interval="30s" name="monitor" timeout="120s"/>
        <op id="vm2-stop-0" interval="0" name="stop" timeout="60s"/>
    </operations>
</primitive>
```
In the example above, the new primitive `vm2` has special attribute values. Its `monitor` operation has a longer `timeout` and `interval`, and the primitive has an additional `stop` operation.

To see the resulting definition of a resource, run:

```
# crm_resource --query-xml --resource vm2
```

To see the raw definition of a resource in the CIB, run:

```
# crm_resource --query-xml-raw --resource vm2
```

### 10.1.2. Using Templates in Constraints

A resource template can be referenced in the following types of constraints:

- **order** constraints (see Section 5.3, “Specifying the Order in which Resources Should Start/Stop”)
- **colocation** constraints (see Section 5.4, “Placing Resources Relative to other Resources”)
- **rsc_ticket** constraints (for multi-site clusters as described in Section 14.3, “Configuring Ticket Dependencies”)

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 example, given the example templates earlier in this section:

```xml
<rsc_colocation id="vm-template-colo-base-rsc" rsc="vm-template" rsc-role="Started" with-rsc="base-rsc" score="INFINITY"/>
```

would colocate all VMs with `base-rsc` and is the equivalent of the following constraint configuration:

```xml
<rsc_colocation id="vm-colo-base-rsc" score="INFINITY">
    <resource_set id="vm-colo-base-rsc-0" sequential="false" role="Started">
        <resource_ref id="vm1"/>
        <resource_ref id="vm2"/>
    </resource_set>
    <resource_set id="vm-colo-base-rsc-1">
        <resource_ref id="base-rsc"/>
    </resource_set>
</rsc_colocation>
```

**Note**

In a colocation constraint, only one template may be referenced from either `rsc` or `with-rsc`; the other reference must be a regular resource.

### 10.1.3. Using Templates in Resource Sets

Resource templates can also be referenced in resource sets.
For example, given the example templates earlier in this section, then:

```xml
<rsc_order id="order1" score="INFINITY">
  <resource_set id="order1-0">
    <resource_ref id="base-rsc"/>
    <resource_ref id="vm-template"/>
    <resource_ref id="top-rsc"/>
  </resource_set>
</rsc_order>
```

is the equivalent of the following constraint using a sequential resource set:

```xml
<rsc_order id="order1" score="INFINITY">
  <resource_set id="order1-0">
    <resource_ref id="base-rsc"/>
    <resource_ref id="vm1"/>
    <resource_ref id="vm2"/>
    <resource_ref id="top-rsc"/>
  </resource_set>
</rsc_order>
```

Or, if the resources referencing the template can run in parallel, then:

```xml
<rsc_order id="order2" score="INFINITY">
  <resource_set id="order2-0">
    <resource_ref id="base-rsc"/>
  </resource_set>
  <resource_set id="order2-1" sequential="false">
    <resource_ref id="vm-template"/>
  </resource_set>
  <resource_set id="order2-2">
    <resource_ref id="top-rsc"/>
  </resource_set>
</rsc_order>
```

is the equivalent of the following constraint configuration:

```xml
<rsc_order id="order2" score="INFINITY">
  <resource_set id="order2-0">
    <resource_ref id="base-rsc"/>
  </resource_set>
  <resource_set id="order2-1" sequential="false">
    <resource_ref id="vm1"/>
    <resource_ref id="vm2"/>
  </resource_set>
  <resource_set id="order2-2">
    <resource_ref id="top-rsc"/>
  </resource_set>
</rsc_order>
```

### 10.2. Reusing Rules, Options and Sets of Operations

Sometimes a number of constraints need to use the same set of rules, and resources need to set the same options and parameters. To simplify this situation, you can refer to an existing object using an `id-ref` instead of an `id`.

So if for one resource you have

```xml
<rsc_location id="WebServer-connectivity" rsc="Webserver">
  <rule id="ping-prefer-rule" score-attribute="pingd"/>
</rsc_location>
```
Then instead of duplicating the rule for all your other resources, you can instead specify:

**Example 10.5. Referencing rules from other constraints**

```xml
<rsc_location id="WebDB-connectivity" rsc="WebDB">
  <rule id-ref="ping-prefer-rule"/>
</rsc_location>
```

### Important

The cluster will insist that the **rule** exists somewhere. Attempting to add a reference to a non-existing rule will cause a validation failure, as will attempting to remove a **rule** that is referenced elsewhere.

The same principle applies for **meta_attributes** and **instance_attributes** as illustrated in the example below:

**Example 10.6. Referencing attributes, options, and operations from other resources**

```xml
<primitive id="mySpecialRsc" class="ocf" type="Special" provider="me">
  <instance_attributes id="mySpecialRsc-attrs" score="1">
    <nvpair id="default-interface" name="interface" value="eth0"/>
    <nvpair id="default-port" name="port" value="9999"/>
  </instance_attributes>
  <meta_attributes id="mySpecialRsc-options">
    <nvpair id="failure-timeout" name="failure-timeout" value="5m"/>
    <nvpair id="migration-threshold" name="migration-threshold" value="1"/>
    <nvpair id="stickiness" name="resource-stickiness" value="0"/>
  </meta_attributes>
  <operations id="health-checks">
    <op id="health-check" name="monitor" interval="60s"/>
    <op id="health-check" name="monitor" interval="30min"/>
  </operations>
</primitive>

<primitive id="myOther1Rsc" class="ocf" type="Other" provider="me">
  <instance_attributes id-ref="mySpecialRsc-attrs"/>
  <meta_attributes id-ref="mySpecialRsc-options"/>
  <operations id-ref="health-checks"/>
</primitive>
```

**id-ref** can similarly be used with **resource_set** (in any constraint type), **nvpair**, and **operations**.

### 10.3. Tagging Configuration Elements

Pacemaker allows you to *tag* any configuration element that has an XML ID.

The main purpose of tagging is to support higher-level user interface tools; Pacemaker itself only uses tags within constraints. Therefore, what you can do with tags mostly depends on the tools you use.
Chapter 10. Reusing Parts of the Configuration

10.3.1. Configuring Tags
A tag is simply a named list of XML IDs.

Example 10.7. Tag referencing three resources

```xml
<tags>
  <tag id="all-vms">
    <obj_ref id="vm1"/>
    <obj_ref id="vm2"/>
    <obj_ref id="vm3"/>
  </tag>
</tags>
```

What you can do with this new tag depends on what your higher-level tools support. For example, a tool might allow you to enable or disable all of the tagged resources at once, or show the status of just the tagged resources.

A single configuration element can be listed in any number of tags.

10.3.2. Using Tags in Constraints and Resource Sets
Pacemaker itself only uses tags in constraints. If you supply a tag name instead of a resource name in any constraint, the constraint will apply to all resources listed in that tag.

Example 10.8. Constraint using a tag

```xml
<rsc_order id="order1" first="storage" then="all-vms" kind="Mandatory" />
```

In the example above, assuming the `all-vms` tag is defined as in the previous example, the constraint will behave the same as:

Example 10.9. Equivalent constraints without tags

```xml
<rsc_order id="order1-1" first="storage" then="vm1" kind="Mandatory" />
<rsc_order id="order1-2" first="storage" then="vm2" kind="Mandatory" />
<rsc_order id="order1-3" first="storage" then="vm2" kind="Mandatory" />
```

A tag may be used directly in the constraint, or indirectly by being listed in a resource set used in the constraint. When used in a resource set, an expanded tag will honor the set's `sequential` property.
Chapter 11.

Utilization and Placement Strategy

Table of Contents

11.1. Utilization attributes ................................................................. 103
11.2. Placement Strategy ................................................................. 104
11.3. Allocation Details ................................................................. 105
  11.3.1. Which node is preferred to get consumed first when allocating resources?  105
  11.3.2. Which node has more free capacity?  ........................................ 105
  11.3.3. Which resource is preferred to be assigned first? ......................... 105
11.4. Limitations and Workarounds ................................................. 106

Pacemaker decides where to place a resource according to the resource allocation scores on every node. The resource will be allocated to the node where the resource has the highest score.

If the resource allocation scores on all the nodes are equal, by the default placement strategy, Pacemaker will choose a node with the least number of allocated resources for balancing the load. If the number of resources on each node is equal, the first eligible node listed in the CIB will be chosen to run the resource.

Often, in real-world situations, different resources use significantly different proportions of a node’s capacities (memory, I/O, etc.). We cannot balance the load ideally just according to the number of resources allocated to a node. Besides, if resources are placed such that their combined requirements exceed the provided capacity, they may fail to start completely or run with degraded performance.

To take these factors into account, Pacemaker allows you to configure:

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

11.1. Utilization attributes

To configure the capacity that a node provides or a resource requires, you can use utilization attributes in node and resource objects. You can name utilization attributes according to your preferences and define as many name/value pairs as your configuration needs. However, the attributes’ values must be integers.

Example 11.1. Specifying CPU and RAM capacities of two nodes

```xml
<node id="node1" type="normal" uname="node1">
  <utilization id="node1-utilization">
    <nvpair id="node1-utilization-cpu" name="cpu" value="2"/>
    <nvpair id="node1-utilization-memory" name="memory" value="2048"/>
  </utilization>
</node>

<node id="node2" type="normal" uname="node2">
  <utilization id="node2-utilization">
    <nvpair id="node2-utilization-cpu" name="cpu" value="4"/>
    <nvpair id="node2-utilization-memory" name="memory" value="4096"/>
  </utilization>
</node>
```
Example 11.2. Specifying CPU and RAM consumed by several resources

```xml
<primitive id="rsc-small" class="ocf" provider="pacemaker" type="Dummy">
  <utilization id="rsc-small-utilization">
    <nvpair id="rsc-small-utilization-cpu" name="cpu" value="1"/>
    <nvpair id="rsc-small-utilization-memory" name="memory" value="1024"/>
  </utilization>
</primitive>

<primitive id="rsc-medium" class="ocf" provider="pacemaker" type="Dummy">
  <utilization id="rsc-medium-utilization">
    <nvpair id="rsc-medium-utilization-cpu" name="cpu" value="2"/>
    <nvpair id="rsc-medium-utilization-memory" name="memory" value="2048"/>
  </utilization>
</primitive>

<primitive id="rsc-large" class="ocf" provider="pacemaker" type="Dummy">
  <utilization id="rsc-large-utilization">
    <nvpair id="rsc-large-utilization-cpu" name="cpu" value="3"/>
    <nvpair id="rsc-large-utilization-memory" name="memory" value="3072"/>
  </utilization>
</primitive>
```

A node is considered eligible for a resource if it has sufficient free capacity to satisfy the resource’s requirements. The nature of the required or provided capacities is completely irrelevant to Pacemaker — it just makes sure that all capacity requirements of a resource are satisfied before placing a resource to a node.

### 11.2. Placement Strategy

After you have configured the capacities your nodes provide and the capacities your resources require, you need to set the `placement-strategy` in the global cluster options, otherwise the capacity configurations have no effect.

Four values are available for the **placement-strategy**:

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

- **utilization**
  - Utilization values are taken into account only when deciding whether a node is considered eligible (i.e. whether it has sufficient free capacity to satisfy the resource’s requirements). Load-balancing is still done based on the number of resources allocated to a node.

- **balanced**
  - Utilization values are taken into account when deciding whether a node is eligible to serve a resource and when load-balancing, so an attempt is made to spread the resources in a way that optimizes resource performance.

- **minimal**
  - Utilization values are taken into account only when deciding whether a node is eligible to serve a resource. For load-balancing, an attempt is made to concentrate the resources on as few nodes as possible, thereby enabling possible power savings on the remaining nodes.

Set **placement-strategy** with `crm_attribute`:

```
# crm_attribute --name placement-strategy --update balanced
```
Now Pacemaker will ensure the load from your resources will be distributed evenly throughout the cluster, without the need for convoluted sets of colocation constraints.

11.3. Allocation Details

11.3.1. Which node is preferred to get consumed first when allocating resources?

- The node with the highest node weight gets consumed first. Node weight is a score maintained by the cluster to represent node health.

- If multiple nodes have the same node weight:
  
  - If the placement-strategy is default or utilization, the node that has the least number of allocated resources gets consumed first.
  
  - If their numbers of allocated resources are equal, the first eligible node listed in the CIB gets consumed first.

  - If the placement-strategy is balanced, the node that has the most free capacity gets consumed first.

  - If the free capacities of the nodes are equal, the node that has the least number of allocated resources gets consumed first.

  - If their numbers of allocated resources are equal, the first eligible node listed in the CIB gets consumed first.

  - If the placement-strategy is minimal, the first eligible node listed in the CIB gets consumed first.

11.3.2. Which node has more free capacity?

If only one type of utilization attribute has been defined, free capacity is a simple numeric comparison.

If multiple types of utilization attributes have been defined, then the node that is numerically highest in the the most attribute types has the most free capacity. For example:

- If nodeA has more free cpus, and nodeB has more free memory, then their free capacities are equal.

- If nodeA has more free cpus, while nodeB has more free memory and storage, then nodeB has more free capacity.

11.3.3. Which resource is preferred to be assigned first?

- The resource that has the highest priority (see Section 4.4, “Resource Options”) gets allocated first.

- If their priorities are equal, check whether they are already running. The resource that has the highest score on the node where it’s running gets allocated first, to prevent resource shuffling.

- If the scores above are equal or the resources are not running, the resource has the highest score on the preferred node gets allocated first.

- If the scores above are equal, the first runnable resource listed in the CIB gets allocated first.
11.4. Limitations and Workarounds

The type of problem Pacemaker is dealing with here is known as the knapsack problem\(^1\) and falls into the \textit{NP-complete}\(^2\) category of computer science problems — a fancy way of saying "it takes a really long time to solve".

Clearly in a HA cluster, it’s not acceptable to spend minutes, let alone hours or days, finding an optimal solution while services remain unavailable.

So instead of trying to solve the problem completely, Pacemaker uses a \textit{best effort} algorithm for determining which node should host a particular service. This means it arrives at a solution much faster than traditional linear programming algorithms, but by doing so at the price of leaving some services stopped.

In the contrived example at the start of this section:

- \texttt{rsc-small} would be allocated to \texttt{node1}
- \texttt{rsc-medium} would be allocated to \texttt{node2}
- \texttt{rsc-large} would remain inactive

Which is not ideal.

There are various approaches to dealing with the limitations of pacemaker's placement strategy:

* **Ensure you have sufficient physical capacity.**
  
  It might sound obvious, but if the physical capacity of your nodes is (close to) maxed out by the cluster under normal conditions, then failover isn’t going to go well. Even without the utilization feature, you’ll start hitting timeouts and getting secondary failures.

* **Build some buffer into the capabilities advertised by the nodes.**
  
  Advertise slightly more resources than we physically have, on the (usually valid) assumption that a resource will not use 100% of the configured amount of CPU, memory and so forth \textit{all} the time. This practice is sometimes called \textit{overcommit}.

* **Specify resource priorities.**
  
  If the cluster is going to sacrifice services, it should be the ones you care about (comparatively) the least. Ensure that resource priorities are properly set so that your most important resources are scheduled first.

---


STONITH

Table of Contents

12.1. What Is STONITH? ................................................................. 107
12.2. What STONITH Device Should You Use? ............................... 107
12.3. Special Treatment of STONITH Resources ............................ 107
12.4. Unfencing ............................................................................. 111
12.5. Configuring STONITH ........................................................... 112
   12.5.1. Example STONITH Configuration .................................. 113
12.6. Advanced STONITH Configurations ........................................ 115
   12.6.1. Example Dual-Layer, Dual-Device Fencing Topologies .......... 116
12.7. Remapping Reboots ............................................................... 122

12.1. What Is STONITH?
STONITH (an acronym for “Shoot The Other Node In The Head”), also called fencing, protects your data from being corrupted by rogue nodes or concurrent access.

Just because a node is unresponsive, this doesn’t mean it isn’t accessing your data. The only way to be 100% sure that your data is safe, is to use STONITH so we can be certain that the node is truly offline, before allowing the data to be accessed from another node.

STONITH also has a role to play in the event that a clustered service cannot be stopped. In this case, the cluster uses STONITH to force the whole node offline, thereby making it safe to start the service elsewhere.

12.2. What STONITH Device Should You Use?
It is crucial that the STONITH device can allow the cluster to differentiate between a node failure and a network one.

The biggest mistake people make in choosing a STONITH device is to use a remote power switch (such as many on-board IPMI controllers) that shares power with the node it controls. In such cases, the cluster cannot be sure if the node is really offline, or active and suffering from a network fault.

Likewise, any device that relies on the machine being active (such as SSH-based “devices” used during testing) are inappropriate.

12.3. Special Treatment of STONITH Resources
STONITH resources are somewhat special in Pacemaker.

STONITH may be initiated by pacemaker or by other parts of the cluster (such as resources like DRBD or DLM). To accommodate this, pacemaker does not require the STONITH resource to be in the started state in order to be used, thus allowing reliable use of STONITH devices in such a case.

All nodes have access to STONITH devices’ definitions and instantiate them on-the-fly when needed, but preference is given to verified instances, which are the ones that are started according to the cluster’s knowledge.
In the case of a cluster split, the partition with a verified instance will have a slight advantage, because the STONITH daemon in the other partition will have to hear from all its current peers before choosing a node to perform the fencing.

Fencing resources do work the same as regular resources in some respects:

- **target-role** can be used to enable or disable the resource
- Location constraints can be used to prevent a specific node from using the resource

**Important**

Currently there is a limitation that fencing resources may only have one set of meta-attributes and one set of instance attributes. This can be revisited if it becomes a significant limitation for people.

See the table below or run `man pacemaker-fenced` to see special instance attributes that may be set for any fencing resource, regardless of fence agent.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stonith-timeout</td>
<td>NA</td>
<td>NA</td>
<td>Older versions used this to override the default period to wait for a STONITH (reboot, on, off) action to complete for this device. It has been replaced by the <code>pcmk_reboot_timeout</code> and <code>pcmk_off_timeout</code> properties.</td>
</tr>
<tr>
<td>provides</td>
<td>string</td>
<td></td>
<td>Any special capability provided by the fence device. Currently, only one such capability is meaningful: <code>unfencing</code> (see Section 12.4, <em>Unfencing</em>).</td>
</tr>
<tr>
<td>pcmk_host_map</td>
<td>string</td>
<td></td>
<td>A mapping of host names to ports numbers for devices that do not support host names. Example: <code>node1:1;node2:2,3</code> tells the cluster to use port 1 for node1 and ports 2 and 3 for node2.</td>
</tr>
<tr>
<td>pcmk_host_list</td>
<td>string</td>
<td></td>
<td>A list of machines controlled by this device (optional unless <code>pcmk_host_check</code> is <code>static-list</code>).</td>
</tr>
<tr>
<td>pcmk_host_check</td>
<td>string</td>
<td>dynamic-list</td>
<td>How to determine which machines are controlled by the device. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>dynamic-list</strong>: query the device via the &quot;list&quot; command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>static-list</strong>: check the <code>pcmk_host_list</code> attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>status</strong>: query the device via the &quot;status&quot; command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>none</strong>: assume every device can fence every machine</td>
</tr>
</tbody>
</table>
### Special Treatment of STONITH Resources

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pcmk_delay_max</code></td>
<td>time</td>
<td>0s</td>
<td>Enable a random delay of up to the time specified before executing stonith actions. This is sometimes used in two-node clusters to ensure that the nodes don’t fence each other at the same time. The overall delay introduced by pacemaker is derived from this random delay value adding a static delay so that the sum is kept below the maximum delay.</td>
</tr>
<tr>
<td><code>pcmk_delay_base</code></td>
<td>time</td>
<td>0s</td>
<td>Enable a static delay before executing stonith actions. This can be used e.g. in two-node clusters to ensure that the nodes don’t fence each other, by having separate fencing resources with different values. The node that is fenced with the shorter delay will lose a fencing race. The overall delay introduced by pacemaker is derived from this value plus a random delay such that the sum is kept below the maximum delay.</td>
</tr>
<tr>
<td><code>pcmk_action_limit</code></td>
<td>integer</td>
<td>1</td>
<td>The maximum number of actions that can be performed in parallel on this device, if the cluster option <code>concurrent-fencing</code> is <code>true</code>. -1 is unlimited.</td>
</tr>
<tr>
<td><code>pcmk_host_argument</code></td>
<td>string</td>
<td><code>port</code></td>
<td>Advanced use only. Which parameter should be supplied to the resource agent to identify the node to be fenced. Some devices do not support the standard <code>port</code> parameter or may provide additional ones. Use this to specify an alternate, device-specific parameter. A value of <code>none</code> tells the cluster not to supply any additional parameters.</td>
</tr>
<tr>
<td><code>pcmk_reboot_action</code></td>
<td>string</td>
<td><code>reboot</code></td>
<td>Advanced use only. The command to send to the resource agent in order to reboot a node. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific command.</td>
</tr>
<tr>
<td><code>pcmk_reboot_timeout</code></td>
<td>time</td>
<td>60s</td>
<td>Advanced use only. Specify an alternate timeout to use for <code>reboot</code> actions instead of the value of <code>stonith-timeout</code>. Some devices need much more or less time to complete than normal. Use this to specify an alternate, device-specific timeout.</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>---------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>pcmk_reboot_retries</td>
<td>integer</td>
<td>2</td>
<td>Advanced use only. The maximum number of times to retry the reboot command within the timeout period. Some devices do not support multiple connections, and operations may fail if the device is busy with another task, so Pacemaker will automatically retry the operation, if there is time remaining. Use this option to alter the number of times Pacemaker retries before giving up.</td>
</tr>
<tr>
<td>pcmk_off_action</td>
<td>string</td>
<td>off</td>
<td>Advanced use only. The command to send to the resource agent in order to shut down a node. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific command.</td>
</tr>
<tr>
<td>pcmk_off_timeout</td>
<td>time</td>
<td>60s</td>
<td>Advanced use only. Specify an alternate timeout to use for off actions instead of the value of stonith-timeout. Some devices need much more or less time to complete than normal. Use this to specify an alternate, device-specific timeout.</td>
</tr>
<tr>
<td>pcmk_off_retries</td>
<td>integer</td>
<td>2</td>
<td>Advanced use only. The maximum number of times to retry the off command within the timeout period. Some devices do not support multiple connections, and operations may fail if the device is busy with another task, so Pacemaker will automatically retry the operation, if there is time remaining. Use this option to alter the number of times Pacemaker retries before giving up.</td>
</tr>
<tr>
<td>pcmk_list_action</td>
<td>string</td>
<td>list</td>
<td>Advanced use only. The command to send to the resource agent in order to list nodes. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific command.</td>
</tr>
<tr>
<td>pcmk_list_timeout</td>
<td>time</td>
<td>60s</td>
<td>Advanced use only. Specify an alternate timeout to use for list actions instead of the value of stonith-timeout. Some devices need much more or less time to complete than normal. Use this to specify an alternate, device-specific timeout.</td>
</tr>
<tr>
<td>pcmk_list_retries</td>
<td>integer</td>
<td>2</td>
<td>Advanced use only. The maximum number of times to retry the list command within the timeout period. Some devices do not support multiple connections, and operations may fail if the device is busy with another task, so Pacemaker will automatically retry the operation, if there is time remaining. Use this option to alter the number of times Pacemaker retries before giving up.</td>
</tr>
</tbody>
</table>
### Unfencing

Most fence devices cut the power to the target. By contrast, fence devices that perform fabric fencing cut off a node’s access to some critical resource, such as a shared disk or a network switch.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pcmk_monitor_action</td>
<td>string</td>
<td>monitor</td>
<td>Advanced use only. The command to send to the resource agent in order to report extended status. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific command.</td>
</tr>
<tr>
<td>pcmk_monitor_timeout</td>
<td>time</td>
<td>60s</td>
<td>Advanced use only. Specify an alternate timeout to use for monitor actions instead of the value of stonith-timeout. Some devices need much more or less time to complete than normal. Use this to specify an alternate, device-specific timeout.</td>
</tr>
<tr>
<td>pcmk_monitor_retries</td>
<td>integer</td>
<td>2</td>
<td>Advanced use only. The maximum number of times to retry the monitor command within the timeout period. Some devices do not support multiple connections, and operations may fail if the device is busy with another task, so Pacemaker will automatically retry the operation, if there is time remaining. Use this option to alter the number of times Pacemaker retries before giving up.</td>
</tr>
<tr>
<td>pcmk_status_action</td>
<td>string</td>
<td>status</td>
<td>Advanced use only. The command to send to the resource agent in order to report status. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific command.</td>
</tr>
<tr>
<td>pcmk_status_timeout</td>
<td>time</td>
<td>60s</td>
<td>Advanced use only. Specify an alternate timeout to use for status actions instead of the value of stonith-timeout. Some devices need much more or less time to complete than normal. Use this to specify an alternate, device-specific timeout.</td>
</tr>
<tr>
<td>pcmk_status_retries</td>
<td>integer</td>
<td>2</td>
<td>Advanced use only. The maximum number of times to retry the status command within the timeout period. Some devices do not support multiple connections, and operations may fail if the device is busy with another task, so Pacemaker will automatically retry the operation, if there is time remaining. Use this option to alter the number of times Pacemaker retries before giving up.</td>
</tr>
</tbody>
</table>
With fabric fencing, it is expected that the cluster will fence the node, and then a system administrator must manually investigate what went wrong, correct any issues found, then reboot (or restart the cluster services on) the node.

Once the node reboots and rejoins the cluster, some fabric fencing devices require that an explicit command to restore the node’s access to the critical resource. This capability is called unfencing and is typically implemented as the fence agent's on command.

If any cluster resource has requires set to unfencing, then that resource will not be probed or started on a node until that node has been unfenced.

### 12.5. Configuring STONITH

1. Find the correct driver:

   ```
   # stonith_admin --list-installed
   ```

2. Find the required parameters associated with the device (replacing $AGENT_NAME with the name obtained from the previous step):

   ```
   # stonith_admin --metadata --agent $AGENT_NAME
   ```

3. Create a file called stonith.xml containing a primitive resource with a class of stonith, a type equal to the agent name obtained earlier, and a parameter for each of the values returned in the previous step.

4. If the device does not know how to fence nodes based on their uname, you may also need to set the special pcmk_host_map parameter. See man pacemaker-fenced for details.

5. If the device does not support the list command, you may also need to set the special pcmk_host_list and/or pcmk_host_check parameters. See man pacemaker-fenced for details.

6. If the device does not expect the victim to be specified with the port parameter, you may also need to set the special pcmk_host_argument parameter. See man pacemaker-fenced for details.

7. Upload it into the CIB using cibadmin:

---

8. Set `stonith-enabled` to true:

   ```bash
   # crm_attribute -t crm_config -n stonith-enabled -v true
   ```

9. Once the stonith resource is running, you can test it by executing the following (although you might want to stop the cluster on that machine first):

   ```bash
   # stonith_admin --reboot nodename
   ```

### 12.5.1. Example STONITH Configuration

Assume we have an chassis containing four nodes and an IPMI device active on 192.0.2.1. We would choose the `fence_ipmilan` driver, and obtain the following list of parameters:

**Example 12.1. Obtaining a list of STONITH Parameters**

```bash
# stonith_admin --metadata -a fence_ipmilan
```

```xml
<resource-agent name="fence_ipmilan" shortdesc="Fence agent for IPMI over LAN">
    <symlink name="fence_ilo3" shortdesc="Fence agent for HP iLO3"/>
    <symlink name="fence_ilo4" shortdesc="Fence agent for HP iLO4"/>
    <symlink name="fence_idrac" shortdesc="Fence agent for Dell iDRAC"/>
    <symlink name="fence_imm" shortdesc="Fence agent for IBM Integrated Management Module"/>
    <longdesc>
        <vendor-url>
        </vendor-url>
    <parameters>
        <parameter name="auth" unique="0" required="0">
            <getopt mixed="-A"/>
            <content type="string"/>
            <shortdesc>
            </shortdesc>
        </parameter>
        <parameter name="ipaddr" unique="0" required="1">
            <getopt mixed="-a"/>
            <content type="string"/>
            <shortdesc>
            </shortdesc>
        </parameter>
        <parameter name="passwd" unique="0" required="0">
            <getopt mixed="-p"/>
            <content type="string"/>
            <shortdesc>
            </shortdesc>
        </parameter>
        <parameter name="passwd_script" unique="0" required="0">
            <getopt mixed="-S"/>
            <content type="string"/>
            <shortdesc>
            </shortdesc>
        </parameter>
        <parameter name="lanplus" unique="0" required="0">
            <getopt mixed="-P"/>
            <content type="boolean"/>
            <shortdesc>
            </shortdesc>
```
Chapter 12. STONITH

```xml
<shortdesc>
</shortdesc>
<parameter name="login" unique="0" required="0">
  <getopt mixed="-l"/>
  <content type="string"/>
  <shortdesc>
  </shortdesc>
</parameter>
<parameter name="action" unique="0" required="0">
  <getopt mixed="-o"/>
  <content type="string" default="reboot"/>
  <shortdesc>
  </shortdesc>
</parameter>
<parameter name="timeout" unique="0" required="0">
  <getopt mixed="-t"/>
  <content type="string"/>
  <shortdesc>
  </shortdesc>
</parameter>
<parameter name="cipher" unique="0" required="0">
  <getopt mixed="-C"/>
  <content type="string"/>
  <shortdesc>
  </shortdesc>
</parameter>
<parameter name="method" unique="0" required="0">
  <getopt mixed="-M"/>
  <content type="string" default="onoff"/>
  <shortdesc>
  </shortdesc>
</parameter>
<parameter name="power_wait" unique="0" required="0">
  <getopt mixed="-T"/>
  <content type="string" default="2"/>
  <shortdesc>
  </shortdesc>
</parameter>
<parameter name="delay" unique="0" required="0">
  <getopt mixed="-f"/>
  <content type="string"/>
  <shortdesc>
  </shortdesc>
</parameter>
<parameter name="privlvl" unique="0" required="0">
  <getopt mixed="-L"/>
  <content type="string"/>
  <shortdesc>
  </shortdesc>
</parameter>
<parameter name="verbose" unique="0" required="0">
  <getopt mixed="-v"/>
  <content type="boolean"/>
  <shortdesc>
  </shortdesc>
</parameter>
</parameters>
<actions>
  <action name="on"/>
  <action name="off"/>
  <action name="reboot"/>
  <action name="status"/>
  <action name="diag"/>
  <action name="list"/>
  <action name="monitor"/>
  <action name="metadata"/>
  <action name="stop" timeout="20s"/>
</actions>
```
Based on that, we would create a STONITH resource fragment that might look like this:

Example 12.2. An IPMI-based STONITH Resource

```xml
<primitive id="Fencing" class="stonith" type="fence_ipmilan">
  <instance_attributes id="Fencing-params">
    <nvpair id="Fencing-passwd" name="passwd" value="testuser"/>
    <nvpair id="Fencing-login" name="login" value="abc123"/>
    <nvpair id="Fencing-ipaddr" name="ipaddr" value="192.0.2.1"/>
    <nvpair id="Fencing-pcmk_host_list" name="pcmk_host_list" value="pcmk-1 pcmk-2"/>
  </instance_attributes>
  <operations>
    <op id="Fencing-monitor-10m" interval="10m" name="monitor" timeout="300s"/>
  </operations>
</primitive>
```

Finally, we need to enable STONITH:

```
# crm_attribute -t crm_config -n stonith-enabled -v true
```

### 12.6. Advanced STONITH Configurations

Some people consider that having one fencing device is a single point of failure; others prefer removing the node from the storage and network instead of turning it off.

Whatever the reason, Pacemaker supports fencing nodes with multiple devices through a feature called **fencing topologies**.

Simply create the individual devices as you normally would, then define one or more **fencing-level** entries in the **fencing-topology** section of the configuration.

- Each fencing level is attempted in order of ascending **index**. Allowed values are 1 through 9.
- If a device fails, processing terminates for the current level. No further devices in that level are exercised, and the next level is attempted instead.
- If the operation succeeds for all the listed devices in a level, the level is deemed to have passed.
- The operation is finished when a level has passed (success), or all levels have been attempted (failed).
- If the operation failed, the next step is determined by the scheduler and/or the controller.

Some possible uses of topologies include:

- Try poison-pill and fail back to power
- Try disk and network, and fall back to power if either fails

---

2 Not true, since a node or resource must fail before fencing even has a chance to
Chapter 12. STONITH

- Initiate a kdump and then poweroff the node

Table 12.2. Properties of Fencing Levels

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>A unique name for the level</td>
</tr>
<tr>
<td>target</td>
<td>The name of a single node to which this level applies</td>
</tr>
<tr>
<td>target-pattern</td>
<td>An extended regular expression (as defined in POSIX) matching the names of nodes to which this level applies</td>
</tr>
<tr>
<td>target-attribute</td>
<td>The name of a node attribute that is set (to target-value) for nodes to which this level applies</td>
</tr>
<tr>
<td>target-value</td>
<td>The node attribute value (of target-attribute) that is set for nodes to which this level applies</td>
</tr>
<tr>
<td>index</td>
<td>The order in which to attempt the levels. Levels are attempted in ascending order until one succeeds. Valid values are 1 through 9.</td>
</tr>
<tr>
<td>devices</td>
<td>A comma-separated list of devices that must all be tried for this level</td>
</tr>
</tbody>
</table>

Example 12.3. Fencing topology with different devices for different nodes

```xml
<cib crm_feature_set="3.0.6" validate-with="pacemaker-1.2" admin_epoch="1" epoch="0" num_updates="0">
  <configuration>
    ...
    <fencing-topology>
      <!-- For pcmk-1, try poison-pill and fail back to power -->
      <fencing-level id="f-p1.1" target="pcmk-1" index="1" devices="poison-pill"/>
      <fencing-level id="f-p1.2" target="pcmk-1" index="2" devices="power"/>

      <!-- For pcmk-2, try disk and network, and fail back to power -->
      <fencing-level id="f-p2.1" target="pcmk-2" index="1" devices="disk,network"/>
      <fencing-level id="f-p2.2" target="pcmk-2" index="2" devices="power"/>
    </fencing-topology>
    ...
  </configuration>
</cib>
```

12.6.1. Example Dual-Layer, Dual-Device Fencing Topologies

The following example illustrates an advanced use of fencing-topology in a cluster with the following properties:

- 3 nodes (2 active prod-mysql nodes, 1 prod_mysql-rep in standby for quorum purposes)
- the active nodes have an IPMI-controlled power board reached at 192.0.2.1 and 192.0.2.2
- the active nodes also have two independent PSUs (Power Supply Units) connected to two independent PDUs (Power Distribution Units) reached at 198.51.100.1 (port 10 and port 11) and 203.0.113.1 (port 10 and port 11)
- the first fencing method uses the fence_ipmi agent

3 http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap09.html#tag_09_04
• the second fencing method uses the **fence_apc_snmp** agent targetting 2 fencing devices (one per PSU, either port 10 or 11)

• fencing is only implemented for the active nodes and has location constraints

• fencing topology is set to try IPMI fencing first then default to a "sure-kill" dual PDU fencing

In a normal failure scenario, STONITH will first select **fence_ipmi** to try to kill the faulty node. Using a fencing topology, if that first method fails, STONITH will then move on to selecting **fence_apc_snmp** twice:

• once for the first PDU

• again for the second PDU

The fence action is considered successful only if both PDUs report the required status. If any of them fails, STONITH loops back to the first fencing method, **fence_ipmi**, and so on until the node is fenced or fencing action is cancelled.

**First fencing method: single IPMI device**

Each cluster node has its own dedicated IPMI channel that can be called for fencing using the following primitives:

```
<primitive class="stonith" id="fence_prod-mysql1_ipmi" type="fence_ipmilan">
  <instance_attributes id="fence_prod-mysql1_ipmi-instance_attributes">
    <nvpair id="fence_prod-mysql1_ipmi-instance_attributes-ipaddr" name="ipaddr" value="192.0.2.1"/>
    <nvpair id="fence_prod-mysql1_ipmi-instance_attributes-action" name="action" value="off"/>
    <nvpair id="fence_prod-mysql1_ipmi-instance_attributes-login" name="login" value="fencing"/>
    <nvpair id="fence_prod-mysql1_ipmi-instance_attributes-passwd" name="passwd" value="finishme"/>
    <nvpair id="fence_prod-mysql1_ipmi-instance_attributes-verbose" name="verbose" value="true"/>
    <nvpair id="fence_prod-mysql1_ipmi-instance_attributes-pcmk_host_list" name="pcmk_host_list" value="prod-mysql1"/>
    <nvpair id="fence_prod-mysql1_ipmi-instance_attributes-lanplus" name="lanplus" value="true"/>
  </instance_attributes>
</primitive>
```

```
<primitive class="stonith" id="fence_prod-mysql2_ipmi" type="fence_ipmilan">
  <instance_attributes id="fence_prod-mysql2_ipmi-instance_attributes">
    <nvpair id="fence_prod-mysql2_ipmi-instance_attributes-ipaddr" name="ipaddr" value="192.0.2.2"/>
    <nvpair id="fence_prod-mysql2_ipmi-instance_attributes-action" name="action" value="off"/>
    <nvpair id="fence_prod-mysql2_ipmi-instance_attributes-login" name="login" value="fencing"/>
    <nvpair id="fence_prod-mysql2_ipmi-instance_attributes-passwd" name="passwd" value="finishme"/>
    <nvpair id="fence_prod-mysql2_ipmi-instance_attributes-verbose" name="verbose" value="true"/>
    <nvpair id="fence_prod-mysql2_ipmi-instance_attributes-pcmk_host_list" name="pcmk_host_list" value="prod-mysql2"/>
    <nvpair id="fence_prod-mysql2_ipmi-instance_attributes-lanplus" name="lanplus" value="true"/>
  </instance_attributes>
</primitive>
```
Second fencing method: dual PDU devices

Each cluster node also has two distinct power channels controlled by two distinct PDUs. That means a total of 4 fencing devices configured as follows:

- Node 1, PDU 1, PSU 1 @ port 10
- Node 1, PDU 2, PSU 2 @ port 10
- Node 2, PDU 1, PSU 1 @ port 11
- Node 2, PDU 2, PSU 2 @ port 11

The matching fencing agents are configured as follows:

```xml
<primitive class="stonith" id="fence_prod-mysql1_apc1" type="fence_apc_snmp">
  <instance_attributes id="fence_prod-mysql1_apc1-instance_attributes">
    <nvpair id="fence_prod-mysql1_apc1-instance_attributes-ipaddr" name="ipaddr" value="198.51.100.1"/>
    <nvpair id="fence_prod-mysql1_apc1-instance_attributes-action" name="action" value="off"/>
    <nvpair id="fence_prod-mysql1_apc1-instance_attributes-port" name="port" value="10"/>
    <nvpair id="fence_prod-mysql1_apc1-instance_attributes-login" name="login" value="fencing"/>
    <nvpair id="fence_prod-mysql1_apc1-instance_attributes-passwd" name="passwd" value="fencing"/>
    <nvpair id="fence_prod-mysql1_apc1-instance_attributes-pcmk_host_list" name="pcmk_host_list" value="prod-mysql1"/>
  </instance_attributes>
</primitive>

<primitive class="stonith" id="fence_prod-mysql1_apc2" type="fence_apc_snmp">
  <instance_attributes id="fence_prod-mysql1_apc2-instance_attributes">
    <nvpair id="fence_prod-mysql1_apc2-instance_attributes-ipaddr" name="ipaddr" value="203.0.113.1"/>
    <nvpair id="fence_prod-mysql1_apc2-instance_attributes-action" name="action" value="off"/>
    <nvpair id="fence_prod-mysql1_apc2-instance_attributes-port" name="port" value="10"/>
    <nvpair id="fence_prod-mysql1_apc2-instance_attributes-login" name="login" value="fencing"/>
    <nvpair id="fence_prod-mysql1_apc2-instance_attributes-passwd" name="passwd" value="fencing"/>
    <nvpair id="fence_prod-mysql1_apc2-instance_attributes-pcmk_host_list" name="pcmk_host_list" value="prod-mysql1"/>
  </instance_attributes>
</primitive>

<primitive class="stonith" id="fence_prod-mysql2_apc1" type="fence_apc_snmp">
  <instance_attributes id="fence_prod-mysql2_apc1-instance_attributes">
    <nvpair id="fence_prod-mysql2_apc1-instance_attributes-ipaddr" name="ipaddr" value="198.51.100.1"/>
    <nvpair id="fence_prod-mysql2_apc1-instance_attributes-action" name="action" value="off"/>
    <nvpair id="fence_prod-mysql2_apc1-instance_attributes-port" name="port" value="11"/>
    <nvpair id="fence_prod-mysql2_apc1-instance_attributes-login" name="login" value="fencing"/>
    <nvpair id="fence_prod-mysql2_apc1-instance_attributes-passwd" name="passwd" value="fencing"/>
    <nvpair id="fence_prod-mysql2_apc1-instance_attributes-pcmk_host_list" name="pcmk_host_list" value="prod-mysql2"/>
  </instance_attributes>
</primitive>

<primitive class="stonith" id="fence_prod-mysql2_apc2" type="fence_apc_snmp">
  <instance_attributes id="fence_prod-mysql2_apc2-instance_attributes">
    <nvpair id="fence_prod-mysql2_apc2-instance_attributes-ipaddr" name="ipaddr" value="203.0.113.1"/>
    <nvpair id="fence_prod-mysql2_apc2-instance_attributes-action" name="action" value="off"/>
  </instance_attributes>
</primitive>
```
Example Dual-Layer, Dual-Device Fencing Topologies

Location Constraints
To prevent STONITH from trying to run a fencing agent on the same node it is supposed to fence, constraints are placed on all the fencing primitives:

```xml
<constraints>
  <rsc_location id="l_fence_prod-mysql1_ipmi" node="prod-mysql1" rsc="fence_prod-mysql1_ipmi" score="-INFINITY"/>
  <rsc_location id="l_fence_prod-mysql2_ipmi" node="prod-mysql2" rsc="fence_prod-mysql2_ipmi" score="-INFINITY"/>
  <rsc_location id="l_fence_prod-mysql1_apc2" node="prod-mysql1" rsc="fence_prod-mysql1_apc2" score="-INFINITY"/>
  <rsc_location id="l_fence_prod-mysql1_apc1" node="prod-mysql1" rsc="fence_prod-mysql1_apc1" score="-INFINITY"/>
  <rsc_location id="l_fence_prod-mysql2_apc1" node="prod-mysql2" rsc="fence_prod-mysql2_apc1" score="-INFINITY"/>
  <rsc_location id="l_fence_prod-mysql2_apc2" node="prod-mysql2" rsc="fence_prod-mysql2_apc2" score="-INFINITY"/>
</constraints>
```

Fencing topology
Now that all the fencing resources are defined, it's time to create the right topology. We want to first fence using IPMI and if that does not work, fence both PDUs to effectively and surely kill the node.

```xml
<fencing-topology>
  <fencing-level devices="fence_prod-mysql1_ipmi" id="fencing-2" index="1" target="prod-mysql1"/>
  <fencing-level devices="fence_prod-mysql1_apc1,fence_prod-mysql1_apc2" id="fencing-3" index="2" target="prod-mysql1"/>
  <fencing-level devices="fence_prod-mysql2_ipmi" id="fencing-0" index="1" target="prod-mysql2"/>
  <fencing-level devices="fence_prod-mysql2_apc1,fence_prod-mysql2_apc2" id="fencing-1" index="2" target="prod-mysql2"/>
</fencing-topology>
```

Please note, in `fencing-topology`, the lowest `index` value determines the priority of the first fencing method.

Final configuration
Put together, the configuration looks like this:

```xml
<configuration>
  <cluster_property_set id="cib-bootstrap-options">
    <nvpair id="cib-bootstrap-options-stonith-enabled" name="stonith-enabled" value="true"/>
  </cluster_property_set>
</configuration>
```

```xml
<cib admin_epoch="0" crm_feature_set="3.0.7" epoch="292" have-quorum="1" num_updates="29" validate-with="pacemaker-1.2">
  <configuration>
    <crm_config>
      <cluster_property_set id="cib-bootstrap-options">
        <nvpair id="cib-bootstrap-options-stonith-enabled" name="stonith-enabled" value="true"/>
        <nvpair id="cib-bootstrap-options-stonith-action" name="stonith-action" value="off"/>
      </cluster_property_set>
    </crm_config>
  </configuration>
</cib>
```
<nvpair id="cib-bootstrap-options-expected-quorum-votes" name="expected-quorum-votes" value="3"/>
...
</cluster_property_set>
</crm_config>
<nodes>
<node id="prod-mysql1" uname="prod-mysql1">
<node id="prod-mysql2" uname="prod-mysql2"/>
<node id="prod-mysql-rep1" uname="prod-mysql-rep1"/>
<instance_attributes id="prod-mysql-rep1"/>
<nvpair id="prod-mysql-rep1-standby" name="standby" value="on"/>
</instance_attributes>
</node>
</nodes>
<resources>
<primitive class="stonith" id="fence_prod-mysql1_ipmi" type="fence_ipmilan">
<instance_attributes id="fence_prod-mysql1_ipmi-instance_attributes">
<nvpair id="fence_prod-mysql1_ipmi-instance_attributes-ipaddr" name="ipaddr" value="192.0.2.1"/>
<nvpair id="fence_prod-mysql1_ipmi-instance_attributes-action" name="action" value="off"/>
<nvpair id="fence_prod-mysql1_ipmi-instance_attributes-login" name="login" value="fencing"/>
<nvpair id="fence_prod-mysql1_ipmi-instance_attributes-passwd" name="passwd" value="finishme"/>
<nvpair id="fence_prod-mysql1_ipmi-instance_attributes-pcmk_host_list" name="pcmk_host_list" value="prod-mysql1"/>
<nvpair id="fence_prod-mysql1_ipmi-instance_attributes-lanplus" name="lanplus" value="true"/>
</instance_attributes>
</primitive>
<primitive class="stonith" id="fence_prod-mysql2_ipmi" type="fence_ipmilan">
<instance_attributes id="fence_prod-mysql2_ipmi-instance_attributes">
<nvpair id="fence_prod-mysql2_ipmi-instance_attributes-ipaddr" name="ipaddr" value="192.0.2.2"/>
<nvpair id="fence_prod-mysql2_ipmi-instance_attributes-action" name="action" value="off"/>
<nvpair id="fence_prod-mysql2_ipmi-instance_attributes-login" name="login" value="fencing"/>
<nvpair id="fence_prod-mysql2_ipmi-instance_attributes-passwd" name="passwd" value="finishme"/>
<nvpair id="fence_prod-mysql2_ipmi-instance_attributes-pcmk_host_list" name="pcmk_host_list" value="prod-mysql2"/>
<nvpair id="fence_prod-mysql2_ipmi-instance_attributes-lanplus" name="lanplus" value="true"/>
</instance_attributes>
</primitive>
<primitive class="stonith" id="fence_prod-mysql1_apc1" type="fence_apc_snmp">
<instance_attributes id="fence_prod-mysql1_apc1-instance_attributes">
<nvpair id="fence_prod-mysql1_apc1-instance_attributes-ipaddr" name="ipaddr" value="198.51.100.1"/>
<nvpair id="fence_prod-mysql1_apc1-instance_attributes-action" name="action" value="off"/>
<nvpair id="fence_prod-mysql1_apc1-instance_attributes-port" name="port" value="10"/>
<nvpair id="fence_prod-mysql1_apc1-instance_attributes-login" name="login" value="fencing"/>
<nvpair id="fence_prod-mysql1_apc1-instance_attributes-passwd" name="passwd" value="fencing"/>
<nvpair id="fence_prod-mysql1_apc1-instance_attributes-pcmk_host_list" name="pcmk_host_list" value="prod-mysql1"/>
</instance_attributes>
</primitive>
Example Dual-Layer, Dual-Device Fencing Topologies

```
<primitive class="stonith" id="fence_prod-mysql1_apc2" type="fence_apc_snmp">
  <instance_attributes id="fence_prod-mysql1_apc2-instance_attributes">
    <nvpair id="fence_prod-mysql1_apc2-instance_attributes-ipaddr" name="ipaddr" value="203.0.113.1"/>
    <nvpair id="fence_prod-mysql1_apc2-instance_attributes-action" name="action" value="off"/>
    <nvpair id="fence_prod-mysql1_apc2-instance_attributes-port" name="port" value="10"/>
    <nvpair id="fence_prod-mysql1_apc2-instance_attributes-login" name="login" value="fencing"/>
    <nvpair id="fence_prod-mysql1_apc2-instance_attributes-passwd" name="passwd" value="fencing"/>
    <nvpair id="fence_prod-mysql1_apc2-instance_attributes-pcmk_host_list" name="pcmk_host_list" value="prod-mysql1"/>
  </instance_attributes>
</primitive>

<primitive class="stonith" id="fence_prod-mysql2_apc1" type="fence_apc_snmp">
  <instance_attributes id="fence_prod-mysql2_apc1-instance_attributes">
    <nvpair id="fence_prod-mysql2_apc1-instance_attributes-ipaddr" name="ipaddr" value="198.51.100.1"/>
    <nvpair id="fence_prod-mysql2_apc1-instance_attributes-action" name="action" value="off"/>
    <nvpair id="fence_prod-mysql2_apc1-instance_attributes-port" name="port" value="11"/>
    <nvpair id="fence_prod-mysql2_apc1-instance_attributes-login" name="login" value="fencing"/>
    <nvpair id="fence_prod-mysql2_apc1-instance_attributes-passwd" name="passwd" value="fencing"/>
    <nvpair id="fence_prod-mysql2_apc1-instance_attributes-pcmk_host_list" name="pcmk_host_list" value="prod-mysql2"/>
  </instance_attributes>
</primitive>

<primitive class="stonith" id="fence_prod-mysql2_apc2" type="fence_apc_snmp">
  <instance_attributes id="fence_prod-mysql2_apc2-instance_attributes">
    <nvpair id="fence_prod-mysql2_apc2-instance_attributes-ipaddr" name="ipaddr" value="203.0.113.1"/>
    <nvpair id="fence_prod-mysql2_apc2-instance_attributes-action" name="action" value="off"/>
    <nvpair id="fence_prod-mysql2_apc2-instance_attributes-port" name="port" value="11"/>
    <nvpair id="fence_prod-mysql2_apc2-instance_attributes-login" name="login" value="fencing"/>
    <nvpair id="fence_prod-mysql2_apc2-instance_attributes-passwd" name="passwd" value="fencing"/>
    <nvpair id="fence_prod-mysql2_apc2-instance_attributes-pcmk_host_list" name="pcmk_host_list" value="prod-mysql2"/>
  </instance_attributes>
</primitive>
```

```
<constraints>
  <rsc_location id="l_fence_prod-mysql1_ipmi" node="prod-mysql1" rsc="fence_prod-mysql1_ipmi" score="-INFINITY"/>
  <rsc_location id="l_fence_prod-mysql2_ipmi" node="prod-mysql2" rsc="fence_prod-mysql2_ipmi" score="-INFINITY"/>
  <rsc_location id="l_fence_prod-mysql1_apc2" node="prod-mysql1" rsc="fence_prod-mysql1_apc2" score="-INFINITY"/>
  <rsc_location id="l_fence_prod-mysql2_apc1" node="prod-mysql2" rsc="fence_prod-mysql2_apc1" score="-INFINITY"/>
  <rsc_location id="l_fence_prod-mysql2_apc2" node="prod-mysql2" rsc="fence_prod-mysql2_apc2" score="-INFINITY"/>
</constraints>
```

```
<fencing-topology>
  <fencing-level devices="fence_prod-mysql1_ipmi" id="fencing-2" index="1" target="prod-mysql1"/>
</fencing-topology>
```
12.7. Remapping Reboots

When the cluster needs to reboot a node, whether because `stonith-action` is `reboot` or because a reboot was manually requested (such as by `stonith_admin --reboot`), it will remap that to other commands in two cases:

1. If the chosen fencing device does not support the `reboot` command, the cluster will ask it to perform `off` instead.

2. If a fencing topology level with multiple devices must be executed, the cluster will ask all the devices to perform `off`, then ask the devices to perform `on`.

To understand the second case, consider the example of a node with redundant power supplies connected to intelligent power switches. Rebooting one switch and then the other would have no effect on the node. Turning both switches off, and then on, actually reboots the node.

In such a case, the fencing operation will be treated as successful as long as the `off` commands succeed, because then it is safe for the cluster to recover any resources that were on the node. Timeouts and errors in the `on` phase will be logged but ignored.

When a reboot operation is remapped, any action-specific timeout for the remapped action will be used (for example, `pcmk_off_timeout` will be used when executing the `off` command, not `pcmk_reboot_timeout`).

```xml
<fencing-level devices="fence_prod-mysql1_apc1,fence_prod-mysql1_apc2" id="fencing-3" index="2" target="prod-mysql1" />
<fencing-level devices="fence_prod-mysql2_ipmi" id="fencing-0" index="1" target="prod-mysql2" />
<fencing-level devices="fence_prod-mysql2_apc1,fence_prod-mysql2_apc2" id="fencing-1" index="2" target="prod-mysql2" />
</fencing-topology>
...<configuration>
</cib>
```
Status — Here be dragons

Table of Contents

13.1. Node Status .................................................................................................................... 123
13.2. Transient Node Attributes ........................................................................................... 124
13.3. Operation History ......................................................................................................... 124
   13.3.1. Simple Operation History Example ................................................................. 126
   13.3.2. Complex Operation History Example ............................................................. 127

Most users never need to understand the contents of the status section and can be happy with the output from `crm_mon`.

However for those with a curious inclination, this section attempts to provide an overview of its contents.

### 13.1. Node Status

In addition to the cluster's configuration, the CIB holds an up-to-date representation of each cluster node in the status section.

**Example 13.1. A bare-bones status entry for a healthy node cl-virt-1**

```
<node_state id="1" uname="cl-virt-1" in_ccm="true" crmd="online" crm-debug-origin="do_update_resource" join="member" expected="member">
  <transient_attributes id="1"/>
  <lrm id="1"/>
</node_state>
```

Users are highly recommended *not* to modify any part of a node's state directly. The cluster will periodically regenerate the entire section from authoritative sources, so any changes should be done with the tools appropriate to those sources.

**Table 13.1. Authoritative Sources for State Information**

<table>
<thead>
<tr>
<th>CIB Object</th>
<th>Authoritative Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>node_state</td>
<td>pacemaker-controld</td>
</tr>
<tr>
<td>transient_attributes</td>
<td>pacemaker-atrd</td>
</tr>
<tr>
<td>lrm</td>
<td>pacemaker-execd</td>
</tr>
</tbody>
</table>

The fields used in the node_state objects are named as they are largely for historical reasons and are rooted in Pacemaker's origins as the resource manager for the older Heartbeat project. They have remained unchanged to preserve compatibility with older versions.

**Table 13.2. Node Status Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique identifier for the node. Corosync-based clusters use a numeric counter.</td>
</tr>
<tr>
<td>uname</td>
<td>The node's name as known by the cluster</td>
</tr>
</tbody>
</table>
Chapter 13. Status — Here be dragons

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in_ccm</td>
<td>Is the node a member at the cluster communication layer? Allowed values: true, false.</td>
</tr>
<tr>
<td>crmd</td>
<td>Is the node a member at the pacemaker layer? Allowed values: online, offline.</td>
</tr>
<tr>
<td>crm-debug-origin</td>
<td>The name of the source function that made the most recent change (for debugging purposes).</td>
</tr>
<tr>
<td>join</td>
<td>Does the node participate in hosting resources? Allowed values: down, pending, member, banned.</td>
</tr>
<tr>
<td>expected</td>
<td>Expected value for join.</td>
</tr>
</tbody>
</table>

The cluster uses these fields to determine whether, at the node level, the node is healthy or is in a failed state and needs to be fenced.

13.2. Transient Node Attributes

Like regular node attributes, the name/value pairs listed in the transient_attributes section help to describe the node. However they are forgotten by the cluster when the node goes offline. This can be useful, for instance, when you want a node to be in standby mode (not able to run resources) just until the next reboot.

In addition to any values the administrator sets, the cluster will also store information about failed resources here.

Example 13.2. A set of transient node attributes for node cl-virt-1

```
<transient_attributes id="cl-virt-1">
  <instance_attributes id="status-cl-virt-1">
    <nvpair id="status-cl-virt-1-pingd" name="pingd" value="3"/>
    <nvpair id="status-cl-virt-1-probe_complete" name="probe_complete" value="true"/>
    <nvpair id="status-cl-virt-1-fail-count-pingd:0#monitor_30000" name="fail-count-pingd:0#monitor_30000" value="1"/>
    <nvpair id="status-cl-virt-1-last-failure-pingd:0" name="last-failure-pingd:0" value="1239009742"/>
  </instance_attributes>
</transient_attributes>
```

In the above example, we can see that a monitor on the pingd:0 resource has failed once, at 09:22:22 UTC 6 April 2009. We also see that the node is connected to three pingd peers and that all known resources have been checked for on this machine (probe_complete).

13.3. Operation History

A node's resource history is held in the lrm_resources tag (a child of the lrm tag). The information stored here includes enough information for the cluster to stop the resource safely if it is removed from the configuration section. Specifically, the resource's id, class, type and provider are stored.

---

¹ You can use the standard date command to print a human-readable version of any seconds-since-epoch value, for example date -d @1239009742.
Example 13.3. A record of the `apcstonith` resource

```xml
<lrm_resource id="apcstonith" type="apcmastersnmp" class="stonith"/>
```

Additionally, we store the last job for every combination of resource, action and interval. The concatenation of the values in this tuple are used to create the id of the `lrn_rsc_op` object.

Table 13.3. Contents of an `lrn_rsc_op` job

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Identifier for the job constructed from the resource's id, operation and interval.</td>
</tr>
<tr>
<td>call-id</td>
<td>The job's ticket number. Used as a sort key to determine the order in which the jobs were executed.</td>
</tr>
<tr>
<td>operation</td>
<td>The action the resource agent was invoked with.</td>
</tr>
<tr>
<td>interval</td>
<td>The frequency, in milliseconds, at which the operation will be repeated. A one-off job is indicated by 0.</td>
</tr>
<tr>
<td>op-status</td>
<td>The job's status. Generally this will be either 0 (done) or -1 (pending). Rarely used in favor of rc-code.</td>
</tr>
<tr>
<td>rc-code</td>
<td>The job's result. Refer to the Resource Agents section of Pacemaker Administration for details on what the values here mean and how they are interpreted.</td>
</tr>
<tr>
<td>last-run</td>
<td>Machine-local date/time, in seconds since epoch, at which the job was executed. For diagnostic purposes.</td>
</tr>
<tr>
<td>last-rc-change</td>
<td>Machine-local date/time, in seconds since epoch, at which the job first returned the current value of rc-code. For diagnostic purposes.</td>
</tr>
<tr>
<td>exec-time</td>
<td>Time, in milliseconds, that the job was running for. For diagnostic purposes.</td>
</tr>
<tr>
<td>queue-time</td>
<td>Time, in seconds, that the job was queued for in the LRMd. For diagnostic purposes.</td>
</tr>
</tbody>
</table>
### Field Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>crm_feature_set</strong></td>
<td>The version which this job description conforms to. Used when processing <strong>op-digest</strong>.</td>
</tr>
<tr>
<td><strong>transition-key</strong></td>
<td>A concatenation of the job's graph action number, the graph number, the expected result and the UUID of the controller instance that scheduled it. This is used to construct <strong>transition-magic</strong> (below).</td>
</tr>
<tr>
<td><strong>transition-magic</strong></td>
<td>A concatenation of the job's <strong>op-status</strong>, <strong>rc-code</strong> and <strong>transition-key</strong>. Guaranteed to be unique for the life of the cluster (which ensures it is part of CIB update notifications) and contains all the information needed for the controller to correctly analyze and process the completed job. Most importantly, the decomposed elements tell the controller if the job entry was expected and whether it failed.</td>
</tr>
<tr>
<td><strong>op-digest</strong></td>
<td>An MD5 sum representing the parameters passed to the job. Used to detect changes to the configuration, to restart resources if necessary.</td>
</tr>
<tr>
<td><strong>crm-debug-origin</strong></td>
<td>The origin of the current values. For diagnostic purposes.</td>
</tr>
</tbody>
</table>

### 13.3.1. Simple Operation History Example

#### Example 13.4. A monitor operation (determines current state of the *apcstonith* resource)

```xml
<lrm_resource id="apcstonith" type="apcmastersnmp" class="stonith">
  <lrm_rsc_op id="apcstonith_monitor_0" operation="monitor" call-id="2"
    rc-code="7" op-status="0" interval="0"
    crm-debug-origin="do_update_resource" crm_feature_set="3.0.1"
    op-digest="2e3da9274d3550dc6526fb24bfcbbca0"
    transition-key="22:2:7:2668bbeb-06d5-40f9-936d-24cb7f87006a"
    transition-magic="0:7;22:2:7:2668bbeb-06d5-40f9-936d-24cb7f87006a"
    last-run="1239008085" last-rc-change="1239008085" exec-time="10" queue-time="0"/>
</lrm_resource>
```

In the above example, the job is a non-recurring monitor operation often referred to as a “probe” for the *apcstonith* resource.

The cluster schedules probes for every configured resource on a node when the node first starts, in order to determine the resource’s current state before it takes any further action.

From the **transition-key**, we can see that this was the 22nd action of the 2nd graph produced by this instance of the controller (2668bbeb-06d5-40f9-936d-24cb7f87006a).

The third field of the **transition-key** contains a 7, which indicates that the job expects to find the resource inactive. By looking at the **rc-code** property, we see that this was the case.
As that is the only job recorded for this node, we can conclude that the cluster started the resource elsewhere.

### 13.3.2. Complex Operation History Example

#### Example 13.5. Resource history of a `pingd` clone with multiple jobs

```xml
<lrm_resource id="pingd:0" type="pingd" class="ocf" provider="pacemaker">
  <lrm_rsc_op id="pingd:0_monitor_30000" operation="monitor" call-id="34"
    rc-code="0" op-status="0" interval="30000"
    crm-debug-origin="do_update_resource" crm_feature_set="3.0.1"
    transition-key="10:11:0:2668bbeb-06d5-40f9-936d-24cb7f87006a"
    ...
    last-run="1239009741" last-rc-change="1239009741" exec-time="10" queue-time="0"/>
  <lrm_rsc_op id="pingd:0_stop_0" operation="stop"
    crm-debug-origin="do_update_resource" crm_feature_set="3.0.1" call-id="32"
    rc-code="0" op-status="0" interval="0"
    transition-key="11:11:0:2668bbeb-06d5-40f9-936d-24cb7f87006a"
    ...
    last-run="1239009741" last-rc-change="1239009741" exec-time="10" queue-time="0"/>
  <lrm_rsc_op id="pingd:0_start_0" operation="start" call-id="33"
    rc-code="0" op-status="0" interval="0"
    crm-debug-origin="do_update_resource" crm_feature_set="3.0.1"
    transition-key="31:11:0:2668bbeb-06d5-40f9-936d-24cb7f87006a"
    ...
    last-run="1239009741" last-rc-change="1239009741" exec-time="10" queue-time="0"/>
  <lrm_rsc_op id="pingd:0_monitor_0" operation="monitor" call-id="3"
    rc-code="0" op-status="0" interval="0"
    crm-debug-origin="do_update_resource" crm_feature_set="3.0.1"
    transition-key="23:2:7:2668bbeb-06d5-40f9-936d-24cb7f87006a"
    ...
    last-run="1239008085" last-rc-change="1239008085" exec-time="20" queue-time="0"/>
</lrm_resource>
```

When more than one job record exists, it is important to first sort them by `call-id` before interpreting them.

Once sorted, the above example can be summarized as:

1. A non-recurring monitor operation returning 7 (not running), with a `call-id` of 3
2. A stop operation returning 0 (success), with a `call-id` of 32
3. A start operation returning 0 (success), with a `call-id` of 33
4. A recurring monitor returning 0 (success), with a `call-id` of 34

The cluster processes each job record to build up a picture of the resource's state. After the first and second entries, it is considered stopped, and after the third it considered active.

Based on the last operation, we can tell that the resource is currently active.

Additionally, from the presence of a `stop` operation with a lower `call-id` than that of the `start` operation, we can conclude that the resource has been restarted. Specifically this occurred as part of actions 11 and 31 of transition 11 from the controller instance with the key 2668bbeb... This information can be helpful for locating the relevant section of the logs when looking for the source of a failure.
Multi-Site Clusters and Tickets

Table of Contents

14.1. Challenges for Multi-Site Clusters ................................................................. 129
14.2. Conceptual Overview ....................................................................................... 129
  14.2.1. Ticket ........................................................................................................ 129
  14.2.2. Dead Man Dependency ............................................................................. 130
  14.2.3. Cluster Ticket Registry ............................................................................ 130
  14.2.4. Configuration Replication ....................................................................... 130
14.3. Configuring Ticket Dependencies ................................................................. 131
14.4. Managing Multi-Site Clusters .................................................................... 132
  14.4.1. Granting and Revoking Tickets Manually .............................................. 132
  14.4.2. Granting and Revoking Tickets via a Cluster Ticket Registry ................... 132
  14.4.3. General Management of Tickets .............................................................. 133
14.5. For more information ................................................................................. 134

Apart from local clusters, Pacemaker also supports multi-site clusters. That means you can have multiple, geographically dispersed sites, each with a local cluster. Failover between these clusters can be coordinated manually by the administrator, or automatically by a higher-level entity called a Cluster Ticket Registry (CTR).

14.1. Challenges for Multi-Site Clusters

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

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

In the following sections, learn how to meet these challenges.

14.2. Conceptual Overview

Multi-site clusters can be considered as “overlay” clusters where each cluster site corresponds to a cluster node in a traditional cluster. The overlay cluster can be managed by a CTR in order to guarantee that any cluster resource will be active on no more than one cluster site. This is achieved by using tickets that are treated as failover domain between cluster sites, in case a site should be down.

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

14.2.1. Ticket

Tickets are, essentially, cluster-wide attributes. A ticket grants the right to run certain resources on a specific cluster site. Resources can be bound to a certain ticket by \texttt{rsc\_ticket} constraints. Only if...
the ticket is available at a site can the respective resources be started there. Vice versa, if the ticket is revoked, the resources depending on that ticket must be stopped.

The ticket thus is similar to a site quorum, i.e. the permission to manage/own resources associated with that site. (One can also think of the current have-quorum flag as a special, cluster-wide ticket that is granted in case of node majority.)

Tickets can be granted and revoked either manually by administrators (which could be the default for classic enterprise clusters), or via the automated CTR mechanism described below.

A ticket can only be owned by one site at a time. Initially, none of the sites has a ticket. Each ticket must be granted once by the cluster administrator.

The presence or absence of tickets for a site is stored in the CIB as a cluster status. With regards to a certain ticket, there are only two states for a site: true (the site has the ticket) or false (the site does not have the ticket). The absence of a certain ticket (during the initial state of the multi-site cluster) is the same as the value false.

### 14.2.2. Dead Man Dependency

A site can only activate resources safely if it can be sure that the other site has deactivated them. However after a ticket is revoked, it can take a long time until all resources depending on that ticket are stopped “cleanly”, especially in case of cascaded resources. To cut that process short, the concept of a Dead Man Dependency was introduced.

If a dead man dependency is in force, if a ticket is revoked from a site, the nodes that are hosting dependent resources are fenced. This considerably speeds up the recovery process of the cluster and makes sure that resources can be migrated more quickly.

This can be configured by specifying a loss-policy="fence" in rsc_ticket constraints.

### 14.2.3. Cluster Ticket Registry

A CTR is a coordinated group of network daemons that automatically handles granting, revoking, and timing out tickets (instead of the administrator revoking the ticket somewhere, waiting for everything to stop, and then granting it on the desired site).

Pacemaker does not implement its own CTR, but interoperates with external software designed for that purpose (similar to how resource and fencing agents are not directly part of pacemaker).

Participating clusters run the CTR daemons, which connect to each other, exchange information about their connectivity, and vote on which sites gets which tickets.

A ticket is granted to a site only once the CTR is sure that the ticket has been relinquished by the previous owner, implemented via a timer in most scenarios. If a site loses connection to its peers, its tickets time out and recovery occurs. After the connection timeout plus the recovery timeout has passed, the other sites are allowed to re-acquire the ticket and start the resources again.

This can also be thought of as a “quorum server”, except that it is not a single quorum ticket, but several.

### 14.2.4. Configuration Replication

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

The **rsc_ticket** constraint lets you specify the resources depending on a certain ticket. Together with the constraint, you can set a **loss-policy** that defines what should happen to the respective resources if the ticket is revoked.

The attribute **loss-policy** can have the following values:

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

**Example 14.1. Constraint that fences node if ticketA is revoked**

```xml
<rsc_ticket id="rsc1-req-ticketA" rsc="rsc1" ticket="ticketA" loss-policy="fence"/>
```

The example above creates a constraint with the ID **rsc1-req-ticketA**. It defines that the resource **rsc1** depends on **ticketA** and that the node running the resource should be fenced if **ticketA** is revoked.

If resource **rsc1** were a promotable resource (i.e. it could run in master or slave mode), you might want to configure that only master mode depends on **ticketA**. With the following configuration, **rsc1** will be demoted to slave mode if **ticketA** is revoked:

**Example 14.2. Constraint that demotes rsc1 if ticketA is revoked**

```xml
<rsc_ticket id="rsc1-req-ticketA" rsc="rsc1" rsc-role="Master" ticket="ticketA" loss-policy="demote"/>
```

You can create multiple **rsc_ticket** constraints to let multiple resources depend on the same ticket. However, **rsc_ticket** also supports resource sets (see **Section 5.5, “Resource Sets”**), so one can easily list all the resources in one **rsc_ticket** constraint instead.

**Example 14.3. Ticket constraint for multiple resources**

```xml
<rsc_ticket id="resources-dep-ticketA" ticket="ticketA" loss-policy="fence">
  <resource_set id="resources-dep-ticketA-0" role="Started">
    <resource_ref id="rsc1"/>
    <resource_ref id="group1"/>
    <resource_ref id="clone1"/>
  </resource_set>
  <resource_set id="resources-dep-ticketA-1" role="Master">
    <resource_ref id="ms1"/>
  </resource_set>
</rsc_ticket>
```

In the example above, there are two resource sets, so we can list resources with different roles in a single **rsc_ticket** constraint. There’s no dependency between the two resource sets, and there’s no dependency among the resources within a resource set. Each of the resources just depends on **ticketA**.
Referencing resource templates in `rsc_ticket` constraints, and even referencing them within resource sets, is also supported.

If you want other resources to depend on further tickets, create as many constraints as necessary with `rsc_ticket`.

14.4. Managing Multi-Site Clusters

14.4.1. Granting and Revoking Tickets Manually

You can grant tickets to sites or revoke them from sites manually. If you want to re-distribute a ticket, you should wait for the dependent resources to stop cleanly at the previous site before you grant the ticket to the new site.

Use the `crm_ticket` command line tool to grant and revoke tickets.

To grant a ticket to this site:

```bash
# crm_ticket --ticket ticketA --grant
```

To revoke a ticket from this site:

```bash
# crm_ticket --ticket ticketA --revoke
```

**Important**

If you are managing tickets manually, use the `crm_ticket` command with great care, because it cannot check whether the same ticket is already granted elsewhere.

14.4.2. Granting and Revoking Tickets via a Cluster Ticket Registry

We will use Booth\(^1\) here as an example of software that can be used with pacemaker as a Cluster Ticket Registry. Booth implements the Raft\(^2\) algorithm to guarantee the distributed consensus among different cluster sites, and manages the ticket distribution (and thus the failover process between sites).

Each of the participating clusters and arbitrators runs the Booth daemon `boothd`.

An arbitrator is the multi-site equivalent of a quorum-only node in a local cluster. If you have a setup with an even number of sites, you need an additional instance to reach consensus about decisions such as failover of resources across sites. In this case, add one or more arbitrators running at additional sites. Arbitrators are single machines that run a booth instance in a special mode. An arbitrator is especially important for a two-site scenario, otherwise there is no way for one site to distinguish between a network failure between it and the other site, and a failure of the other site.

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

---

\(^1\) [https://github.com/ClusterLabs/booth](https://github.com/ClusterLabs/booth)

Boothd at each site connects to its peers running at the other sites and exchanges connectivity details. Once a ticket is granted to a site, the booth mechanism will manage the ticket automatically: If the site which holds the ticket is out of service, the booth daemons will vote which of the other sites will get the ticket. To protect against brief connection failures, sites that lose the vote (either explicitly or implicitly by being disconnected from the voting body) need to relinquish the ticket after a time-out. Thus, it is made sure that a ticket will only be re-distributed after it has been relinquished by the previous site. The resources that depend on that ticket will fail over to the new site holding the ticket. The nodes that have run the resources before will be treated according to the loss-policy you set within the rsc_ticket constraint.

Before the booth can manage a certain ticket within the multi-site cluster, you initially need to grant it to a site manually via the booth command-line tool. After you have initially granted a ticket to a site, boothd will take over and manage the ticket automatically.

**Important**

The booth command-line tool can be used to grant, list, or revoke tickets and can be run on any machine where boothd is running. If you are managing tickets via Booth, use only booth for manual intervention, not crm_ticket. That ensures the same ticket will only be owned by one cluster site at a time.

14.4.2.1. Booth Requirements

- All clusters that will be part of the multi-site cluster must be based on Pacemaker.
- Booth must be installed on all cluster nodes and on all arbitrators that will be part of the multi-site cluster.
- Nodes belonging to the same cluster site should be synchronized via NTP. However, time synchronization is not required between the individual cluster sites.

14.4.3. General Management of Tickets

Display the information of tickets:

```
# crm_ticket --info
```

Or you can monitor them with:

```
# crm_mon --tickets
```

Display the rsc_ticket constraints that apply to a ticket:

```
# crm_ticket --ticket ticketA --constraints
```

When you want to do maintenance or manual switch-over of a ticket, revoking the ticket would trigger the loss policies. If loss-policy="fence", the dependent resources could not be gracefully stopped/demoted, and other unrelated resources could even be affected.

The proper way is making the ticket standby first with:

```
# crm_ticket --ticket ticketA --standby
```
Chapter 14. Multi-Site Clusters and Tickets

Then the dependent resources will be stopped or demoted gracefully without triggering the loss policies.

If you have finished the maintenance and want to activate the ticket again, you can run:

```
# crm_ticket --ticket ticketA --activate
```

14.5. For more information

- SUSE's Geo Clustering quick start
- Booth

---

3 https://www.suse.com/documentation/sle-ha-geo-12/art_ha_geo_quick/data/art_ha_geo_quick.html
4 https://github.com/ClusterLabs/booth
Appendix A. Sample Configurations

Table of Contents

A.1. Empty ........................................................................................................................................ 135
A.2. Simple ..................................................................................................................................... 135
A.3. Advanced Configuration ......................................................................................................... 136

A.1. Empty

Example A.1. An Empty Configuration

```xml
<cib crm_feature_set="3.0.7" validate-with="pacemaker-1.2" admin_epoch="1" epoch="0"
    num_updates="0">
    <configuration>
        <crm_config/>
        <nodes/>
        <resources/>
        <constraints/>
    </configuration>
    <status/>
</cib>
```

A.2. Simple

Example A.2. A simple configuration with two nodes, some cluster options and a resource

```xml
<cib crm_feature_set="3.0.7" validate-with="pacemaker-1.2" admin_epoch="1" epoch="0"
    num_updates="0">
    <configuration>
        <cluster_property_set id="cib-bootstrap-options">
            <nvpair id="option-1" name="symmetric-cluster" value="true"/>
            <nvpair id="option-2" name="no-quorum-policy" value="stop"/>
            <nvpair id="option-3" name="stonith-enabled" value="0"/>
        </cluster_property_set>
        <nodes>
            <node id="xxx" uname="c001n01" type="normal"/>
            <node id="yyy" uname="c001n02" type="normal"/>
        </nodes>
        <resources>
            <primitive id="myAddr" class="ocf" provider="heartbeat" type="IPaddr">
                <operations>
                    <op id="myAddr-monitor" name="monitor" interval="300s"/>
                </operations>
                <instance_attributes id="myAddr-params">
                    <nvpair id="myAddr-ip" name="ip" value="192.0.2.10"/>
                </instance_attributes>
            </primitive>
            <rsc_location id="myAddr-prefer" rsc="myAddr" node="c001n01" score="INFINITY"/>
        </resources>
    </configuration>
</cib>
```
Appendix A. Sample Configurations

In the above example, we have one resource (an IP address) that we check every five minutes and will run on host \textit{c001n01} until either the resource fails 10 times or the host shuts down.

A.3. Advanced Configuration

Example A.3. An advanced configuration with groups, clones and STONITH

```
<cib crm_feature_set="3.0.7" validate-with="pacemaker-1.2" admin_epoch="1" epoch="0"
    num_updates="0">
  <configuration>
    <crm_config>
      <cluster_property_set id="cib-bootstrap-options">
        <nvpair id="option-1" name="symmetric-cluster" value="true"/>
        <nvpair id="option-2" name="no-quorum-policy" value="stop"/>
        <nvpair id="option-3" name="stonith-enabled" value="true"/>
      </cluster_property_set>
      <nodes>
        <node id="xxx" uname="c001n01" type="normal"/>
        <node id="yyy" uname="c001n02" type="normal"/>
        <node id="zzz" uname="c001n03" type="normal"/>
      </nodes>
      <resources>
        <primitive id="myAddr" class="ocf" provider="heartbeat" type="IPaddr">
          <operations>
            <op id="myAddr-monitor" name="monitor" interval="300s"/>
          </operations>
          <instance_attributes id="myAddr-attrs">
            <nvpair id="myAddr-attr-1" name="ip" value="192.0.2.10"/>
          </instance_attributes>
        </primitive>
        <group id="myGroup">
          <primitive id="database" class="lsb" type="oracle">
            <operations>
              <op id="database-monitor" name="monitor" interval="300s"/>
            </operations>
          </primitive>
          <primitive id="webserver" class="lsb" type="apache">
            <operations>
              <op id="webserver-monitor" name="monitor" interval="300s"/>
            </operations>
          </primitive>
        </group>
        <clone id="STONITH">
          <meta_attributes id="stonith-options">
            <nvpair id="stonith-option-1" name="globally-unique" value="false"/>
          </meta_attributes>
          <primitive id="stonithclone" class="stonith" type="external/ssh">
            <operations>
            </operations>
          </primitive>
        </clone>
      </resources>
    </crm_config>
  </configuration>
</cib>
```
<op id="stonith-op-mon" name="monitor" interval="5s"/>
</operations>

<instance_attributes id="stonith-attrs">
    <nvpair id="stonith-attr-1" name="hostlist" value="c001n01,c001n02"/>
</instance_attributes>
</primitive>
</clone>
</resources>

<constraints>
    <rsc_location id="myAddr-prefer" rsc="myAddr" node="c001n01" score="INFINITY"/>
    <rsc_colocation id="group-with-ip" rsc="myGroup" with-rsc="myAddr" score="INFINITY"/>
</constraints>

<op_defaults>
    <meta_attributes id="op_defaults-options">
        <nvpair id="op-default-1" name="timeout" value="30s"/>
    </meta_attributes>
</op_defaults>

<rsc_defaults>
    <meta_attributes id="rsc_defaults-options">
        <nvpair id="rsc-default-1" name="resource-stickiness" value="100"/>
        <nvpair id="rsc-default-2" name="migration-threshold" value="10"/>
    </meta_attributes>
</rsc_defaults>
</configuration>

<status/>
</cib>
Appendix B. Revision History

Revision 1-0  19 Oct 2009  Andrew Beekhof andrew@beekhof.net
Import from Pages.app

Revision 2-0  26 Oct 2009  Andrew Beekhof andrew@beekhof.net
Cleanup and reformatting of docbook xml complete

Revision 3-0  Tue Nov 12 2009  Andrew Beekhof andrew@beekhof.net
Split book into chapters and pass validation
Re-organize book for use with Publican

Revision 4-0  Mon Oct 8 2012  Andrew Beekhof andrew@beekhof.net
Converted to asciidoc (which is converted to docbook for use with Publican)

Revision 5-0  Mon Feb 23 2015  Ken Gaillot kgaillot@redhat.com
Update for clarity, stylistic consistency and current command-line syntax

Revision 6-0  Tue Dec 8 2015  Ken Gaillot kgaillot@redhat.com
Update for Pacemaker 1.1.14

Revision 7-0  Tue May 3 2016  Ken Gaillot kgaillot@redhat.com
Update for Pacemaker 1.1.15

Revision 7-1  Fri Oct 28 2016  Ken Gaillot kgaillot@redhat.com
Overhaul upgrade documentation, and document node health strategies

Revision 8-0  Tue Oct 25 2016  Ken Gaillot kgaillot@redhat.com
Update for Pacemaker 1.1.16

Revision 9-0  Tue Jul 11 2017  Ken Gaillot kgaillot@redhat.com
Update for Pacemaker 1.1.17

Revision 10-0  Fri Oct 6 2017  Ken Gaillot kgaillot@redhat.com

1 https://fedorahosted.org/publican/
2 http://www.methods.co.nz/asciidoc
3 https://fedorahosted.org/publican/
Appendix B. Revision History

Update for Pacemaker 1.1.18

Revision 11-0  Fri Jan 12 2018  Ken Gaillot kgaillot@redhat.com
Update for Pacemaker 2.0.0

Revision 12-0  Mon Jan 28 2019  Ken Gaillot kgaillot@redhat.com, Reid Wahl nwahl@redhat.com, Jan Pokorný jpkorny@redhat.com
Update for Pacemaker 2.0.1, remove "Further Reading" and "FAQ" sections, and add minor clarifications and reformatting
Symbols
#digests-, 15
   Node attribute, 15
#node-unfenced, 15
   Node attribute, 15

A
Action, 26
   Property
      enabled, 27, 27
      id, 26
      interval, 26
      name, 26
      on-fail, 27
      role, 27
      timeout, 26
Status
   call-id, 125
   crm-debug-origin, 126
   crm_feature_set, 126
   exec-time, 125
   id, 125
   interval, 125
   last-fc-change, 125
   last-run, 125
   op-digest, 126
   op-status, 125
   operation, 125
   queue-time, 125
   rc-code, 125
   transition-key, 126
   transition-magic, 126
action, 39, 81
   Ordering Constraints, 81
   Resource Sets, 39
Action Property, 26, 26, 26, 26, 27, 27, 27, 27, 27
Action Status, 125, 125, 125, 125, 125, 125, 125, 125, 125, 125, 125, 125, 125, 126, 126, 126, 126, 126, 126, 126, 126, 126
active_resource, 83
   Notification Environment Variable, 83
activeuname, 83
   Notification Environment Variable, 83
add-host, 92
   network, 92
admin_epoch, 8
   Cluster Option, 8
Alert
   Option
      timeout, 48
      timestamp-format, 48
Alerts, 47

Asymmetrical Opt-In, 33
Asymmetrical Opt-In Clusters, 33
attribute, 13, 54
   #digests-, 15
   #node-unfenced, 15
Constraint Expression, 54
fail-count-, 14
last-failure-, 15
maintenance, 15
probe_complete, 15
resource-discovery-enabled, 15
shutdown, 15
site-name, 15
standby, 15
terminate, 15
Attribute Expression, 54
   attribute, 54
   id, 54
   operation, 54
   type, 54
   value, 54, 54
   attribute_name, 51
   attribute_value, 51

B
batch-limit, 9
   Cluster Option, 9
boolean-op, 53
   Constraint Rule, 53
bundle, 89, 89, 89, 89, 90, 90
   network, 92
      port-mapping, 93
Property
   description, 90
   id, 90
   storage
      storage-mapping, 93

C
call-id, 125
   Action Status, 125
cib-last-written, 8
   Cluster Property, 8
class, 17, 21
   Resource, 21
Clone
   Option
      clone-max, 78
      clone-min, 78
      clone-node-max, 78
      globally-unique, 78
      interleave, 78
      notify, 78
<table>
<thead>
<tr>
<th>Property</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>78, 79</td>
</tr>
<tr>
<td>ordered</td>
<td>78</td>
</tr>
<tr>
<td>promotable</td>
<td>79</td>
</tr>
<tr>
<td>promoted-max</td>
<td>79</td>
</tr>
<tr>
<td>promoted-node-max</td>
<td>79</td>
</tr>
<tr>
<td>Clone Property</td>
<td>78, 78, 78, 78, 78, 78, 78, 79, 79, 79, 79</td>
</tr>
<tr>
<td>Clone Option</td>
<td>78, 78, 78, 78, 78, 78, 78, 78, 79, 79, 79</td>
</tr>
<tr>
<td>Clone Resources</td>
<td>77</td>
</tr>
<tr>
<td>clone-max</td>
<td>78</td>
</tr>
<tr>
<td>Clone Option</td>
<td>78</td>
</tr>
<tr>
<td>clone-min</td>
<td>78</td>
</tr>
<tr>
<td>Clone Option</td>
<td>78</td>
</tr>
<tr>
<td>clone-node-max</td>
<td>78</td>
</tr>
<tr>
<td>Clone Option</td>
<td>78</td>
</tr>
<tr>
<td>Clones</td>
<td>77, 82</td>
</tr>
<tr>
<td>Cluster</td>
<td>8</td>
</tr>
<tr>
<td>Option</td>
<td></td>
</tr>
<tr>
<td>admin_epoch</td>
<td>8</td>
</tr>
<tr>
<td>batch-limit</td>
<td>9</td>
</tr>
<tr>
<td>cluster-delay</td>
<td>10</td>
</tr>
<tr>
<td>cluster-ipc-limit</td>
<td>11</td>
</tr>
<tr>
<td>concurrent-fencing</td>
<td>10</td>
</tr>
<tr>
<td>Configuration Version</td>
<td>8</td>
</tr>
<tr>
<td>dc-deadtime</td>
<td>10</td>
</tr>
<tr>
<td>election-timeout</td>
<td>11</td>
</tr>
<tr>
<td>enable-startup-probes</td>
<td>9</td>
</tr>
<tr>
<td>epoch</td>
<td>8</td>
</tr>
<tr>
<td>join-finalization-timeout</td>
<td>12</td>
</tr>
<tr>
<td>join-integration-timeout</td>
<td>12</td>
</tr>
<tr>
<td>maintenance-mode</td>
<td>10</td>
</tr>
<tr>
<td>migration-limit</td>
<td>9</td>
</tr>
<tr>
<td>no-quorum-policy</td>
<td>9</td>
</tr>
<tr>
<td>node-health-base</td>
<td>11</td>
</tr>
<tr>
<td>node-health-green</td>
<td>11</td>
</tr>
<tr>
<td>node-health-red</td>
<td>11</td>
</tr>
<tr>
<td>node-health-strategy</td>
<td>11</td>
</tr>
<tr>
<td>node-health-yellow</td>
<td>11</td>
</tr>
<tr>
<td>num_updates</td>
<td>8</td>
</tr>
<tr>
<td>pe-error-series-max</td>
<td>11</td>
</tr>
<tr>
<td>pe-input-series-max</td>
<td>11</td>
</tr>
<tr>
<td>pe-warn-series-max</td>
<td>11</td>
</tr>
<tr>
<td>placement-strategy</td>
<td>11</td>
</tr>
<tr>
<td>remove-after-stop</td>
<td>11</td>
</tr>
<tr>
<td>shutdown-escalation</td>
<td>11</td>
</tr>
<tr>
<td>start-failure-is-fatal</td>
<td>9</td>
</tr>
<tr>
<td>startup-fencing</td>
<td>11</td>
</tr>
<tr>
<td>stonith-action</td>
<td>10</td>
</tr>
<tr>
<td>stonith-enabled</td>
<td>10</td>
</tr>
<tr>
<td>stonith-max-attempts</td>
<td>10</td>
</tr>
<tr>
<td>stonith-timeout</td>
<td>10</td>
</tr>
<tr>
<td>stonith-watchdog-timeout</td>
<td>10</td>
</tr>
<tr>
<td>stop-all-resources</td>
<td>9</td>
</tr>
<tr>
<td>stop-orphan-actions</td>
<td>9</td>
</tr>
<tr>
<td>stop-orphan-resources</td>
<td>9</td>
</tr>
<tr>
<td>symmetric-cluster</td>
<td>9</td>
</tr>
<tr>
<td>transition-delay</td>
<td>12</td>
</tr>
<tr>
<td>validate-with</td>
<td>8</td>
</tr>
<tr>
<td>Property</td>
<td></td>
</tr>
<tr>
<td>cib-last-written</td>
<td>8</td>
</tr>
<tr>
<td>cluster-infrastructure</td>
<td>9</td>
</tr>
<tr>
<td>dc-uuid</td>
<td>8</td>
</tr>
<tr>
<td>dc-version</td>
<td>9</td>
</tr>
<tr>
<td>have-quorum</td>
<td>8</td>
</tr>
<tr>
<td>Setting Options with Rules</td>
<td>60</td>
</tr>
<tr>
<td>Cluster Option</td>
<td>8, 8, 8, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 12, 12</td>
</tr>
<tr>
<td>Cluster Property</td>
<td>8, 8, 8, 9, 9</td>
</tr>
<tr>
<td>Cluster Type</td>
<td></td>
</tr>
<tr>
<td>Asymmetrical Opt-In</td>
<td>33</td>
</tr>
<tr>
<td>Symmetrical Opt-Out</td>
<td>34</td>
</tr>
<tr>
<td>cluster-delay</td>
<td>10</td>
</tr>
<tr>
<td>Cluster Option</td>
<td>10</td>
</tr>
<tr>
<td>cluster-infrastructure</td>
<td>9</td>
</tr>
<tr>
<td>Cluster Property</td>
<td>9</td>
</tr>
<tr>
<td>cluster-ipc-limit</td>
<td>11</td>
</tr>
<tr>
<td>Cluster Option</td>
<td>11</td>
</tr>
<tr>
<td>cluster-recheck-interval</td>
<td>10</td>
</tr>
<tr>
<td>Cluster Option</td>
<td>10</td>
</tr>
<tr>
<td>Colocation</td>
<td>36</td>
</tr>
<tr>
<td>id</td>
<td>37</td>
</tr>
<tr>
<td>node-attribute</td>
<td>37</td>
</tr>
<tr>
<td>rsc, 37</td>
<td></td>
</tr>
<tr>
<td>score, 37</td>
<td></td>
</tr>
<tr>
<td>with-rsc, 37</td>
<td></td>
</tr>
<tr>
<td>Colocation Constraints</td>
<td>37, 37, 37, 37, 37</td>
</tr>
<tr>
<td>concurrent-fencing</td>
<td>10</td>
</tr>
<tr>
<td>Cluster Option</td>
<td>10</td>
</tr>
<tr>
<td>Configuration</td>
<td>107</td>
</tr>
<tr>
<td>Configuration Version</td>
<td>8</td>
</tr>
<tr>
<td>Cluster</td>
<td>8</td>
</tr>
<tr>
<td>Constraint</td>
<td></td>
</tr>
<tr>
<td>Attribute Expression</td>
<td>54</td>
</tr>
<tr>
<td>attribute</td>
<td>54</td>
</tr>
<tr>
<td>id, 54</td>
<td></td>
</tr>
<tr>
<td>operation</td>
<td>54</td>
</tr>
<tr>
<td>type, 54</td>
<td></td>
</tr>
<tr>
<td>value, 54, 54</td>
<td></td>
</tr>
<tr>
<td>Date Specification</td>
<td>56</td>
</tr>
<tr>
<td>hours, 56</td>
<td></td>
</tr>
<tr>
<td>id, 56</td>
<td></td>
</tr>
<tr>
<td>monthdays, 56</td>
<td></td>
</tr>
<tr>
<td>months, 56</td>
<td></td>
</tr>
<tr>
<td>moon, 56</td>
<td></td>
</tr>
</tbody>
</table>
weekdays, 56
weeks, 56
weeklyyears, 56
yeardays, 56
years, 56
Date/Time Expression, 55
  end, 55
  operation, 55
  start, 55
Duration, 56
Rule, 53
  boolean-op, 53
  id, 53
  role, 53
  score, 53
  score-attribute, 53
Constraint Expression, 54, 54, 54, 54, 54, 54, 55, 55, 55
Constraint Rule, 53, 53, 53, 53, 53
Constraints, 31
Colocation, 36
  id, 37
  node-attribute, 37
  rsc, 37
  score, 37
  with-rsc, 37
Location, 32
  id, 32
  node, 32
  Resource Discovery, 33
  rsc, 32
  rsc-pattern, 32
  score, 32
Ordering, 34
  action, 35
  first, 35
  first-action, 35
  id, 35
  kind, 35
  role, 81
  rsc-role, 80
  then, 35
  then-action, 35
  with-rsc-role, 80
Resource Sets
  action, 39
  id, 38
  require-all, 39
  role, 39
  score, 39
  sequential, 38
control-port, 93
  network, 93
Controlling Cluster Options, 60
crm-debug-origin, 124, 126
  Action Status, 126
  Node Status, 124
crmd, 124
  Node Status, 124
CRM_alert_
  attribute_name, 51
  attribute_value, 51
  desc, 51
  exec_time, 51
  interval, 51
  kind, 50
  node, 51
  nodeid, 51
  rc, 51
  recipient, 50
  rsc, 51
  status, 51
  target_rc, 51
  task, 51
  timestamp, 51
  timestamp_epoch, 51
  timestamp_usec, 51
  version, 50
CRM_alert_node_
  sequence, 51
crm_feature_set, 126
  Action Status, 126
custom, 72
D
dampen, 68
  Ping Resource Option, 68
Date Specification, 56, 56, 56, 56, 56, 56, 56, 56, 56, 56, 56, 56
  hours, 56
  id, 56
  monthdays, 56
  months, 56
  moon, 56
  weekdays, 56
  weeks, 56
  weeklyyears, 56
  yeardays, 56
  years, 56
Date/Time Expression, 55
  end, 55
  operation, 55
  start, 55
dc-deadtime, 10
  Cluster Option, 10
dc-uuid, 8
  Cluster Property, 8
dc-version, 9
Index

Cluster Property, 9
demote_resource, 85
  Notification Environment Variable, 85
demote_uname, 85
  Notification Environment Variable, 85
desc, 51
description, 90
  bundle, 90
Determine by Rules, 58
Determine Resource Location, 58
devices, 116
  fencing-level, 116
Docker, 90, 90, 90, 90, 91, 91, 91
  bundle, 89
Property
  image, 90
  network, 91
  options, 91
  promoted-max, 90
  replicas, 90
  replicas-per-host, 90
  run-command, 91
Duration, 56, 56

E
election-timeout, 11
  Cluster Option, 11
enable-startup-probes, 9
  Cluster Option, 9
enabled, 27, 27
  Action Property, 27, 27
end, 55
  Constraint Expression, 55
Environment Variable
  CRM_alert_
    attribute_name, 51
    attribute_value, 51
    desc, 51
    exec_time, 51
    interval, 51
    kind, 50
    node, 51
    nodeid, 51
    rc, 51
    recipient, 50
    rsc, 51
    status, 51
    target_rc, 51
    task, 51
    timestamp, 51
    timestamp_epoch, 51
    timestamp_usec, 51
    version, 50
  CRM_alert_node_
    sequence, 51
OCF_RESKEY_CRM_meta_notify_
  active_resource, 83
  active_uname, 83
demote_resource, 85
demote_uname, 85
inactive_resource, 85
master_resource, 85
master_uname, 85
operation, 83
promote_resource, 85
promote_uname, 85
slave_resource, 85
slave_uname, 85
start_resource, 83
start_uname, 83
stop_resource, 83
stop_uname, 83
type, 83
epoch, 8
  Cluster Option, 8
exec-time, 125
  Action Status, 125
exec_time, 51
expected, 124
  Node Status, 124

F
fail-count-, 14
  Node attribute, 14
failure-timeout, 23
  Resource Option, 23
feedback
  contact information for this manual, xv
  fencing-level
    devices, 116
    id, 116
    index, 116
    target, 116
    target-attribute, 116, 116
    target-pattern, 116
Property
  pcmk_action_limit, 109
  pcmk_delay_base, 109
  pcmk_delay_max, 109
  pcmk_host_argument, 109
  pcmk_host_check, 109
  pcmk_host_list, 108
  pcmk_host_map, 108
  pcmk_list_action, 110
  pcmk_list_retries, 111
Action Status, 125
last-run, 125
Action Status, 125
Linux Standard Base Resources, 18
Location, 32
Determine by Rules, 58
id, 32
node, 32
Resource Discovery, 33
rsc, 32
rsc-pattern, 32
score, 32
Location Constraints, 32, 32, 32, 32, 32, 32, 33
Location Relative to other Resources, 36
LSB, 18
Resources, 18

M
maintenance, 15
Node attribute, 15
maintenance-mode, 10
Cluster Option, 10
master_resource, 85
Notification Environment Variable, 85
master_uname, 85
Notification Environment Variable, 85
migrate-on-red, 72
migration-limit, 9
Cluster Option, 9
migration-threshold, 23
Resource Option, 23
monthdays, 56
Date Specification, 56
months, 56
Date Specification, 56
moon, 56
Date Specification, 56
Moving, 66
Resources, 66
multiple-active, 23
Resource Option, 23
multiplier, 68
Ping Resource Option, 68

N
Nagios Plugins, 20
Resources, 20
name, 26
Action Property, 26
network, 91, 91, 92, 92, 92, 92, 92, 93
Docker, 91
port-mapping, 93

Property
add-host, 92
control-port, 93
host-interface, 92
host-netmask, 92
ip-range-start, 92
rkt, 91
no-quorum-policy, 9
Cluster Option, 9
Node
attribute, 13
#digests-, 15
#node-unfenced, 15
fail-count-, 14
last-failure-, 15
maintenance, 15
probe_complete, 15
resource-discovery-enabled, 15
shutdown, 15
site-name, 15
standby, 15
terminate, 15
Status, 123
crm-debug-origin, 124
crmd, 124
expected, 124
id, 123
in_ccm, 124
join, 124
uname, 123
node, 32, 51
Location Constraints, 32
Node attribute, 14, 15, 15, 15, 15, 15, 15, 15, 15
Node health
custom, 72
green, 71
migrate-on-red, 72
none, 71
only-green, 72
progressive, 72
red, 71
score, 71
yellow, 71
Node Status, 123, 123, 124, 124, 124, 124, 124
node-attribute, 37
Colocation Constraints, 37
node-health-base, 11
Cluster Option, 11
node-health-green, 11
Cluster Option, 11
node-health-red, 11
Cluster Option, 11
node-health-strategy, 11
Cluster Option, 11
node-health-yellow, 11
Cluster Option, 11
nodeid, 51
none, 71
Notification Environment Variable, 83, 83, 83, 83, 83, 83, 85, 85, 85, 85, 85, 85, 85, 85 notify, 78
Clone Option, 78
num_updates, 8
Cluster Option, 8

O
OCF, 18
Resources, 18
OCF_FAILED_MASTER, 83
OCF_NOT_RUNNING, 83
OCF_RESKEY_CRM_meta_notify_
active_resource, 83
active_uname, 83
demote_resource, 85
demote_uname, 85
inactive_resource, 83
master_resource, 85
master_uname, 85
operation, 83
promote_resource, 85
promote_uname, 85
slave_resource, 85
slave_uname, 85
start_resource, 83
start_uname, 83
stop_resource, 83
stop_uname, 83
type, 83
OCF_RUNNING_MASTER, 83
OCF_SUCCESS, 83
on-fail, 27
Action Property, 27
only-green, 72
op-digest, 126
Action Status, 126
op-status, 125
Action Status, 125
Open Cluster Framework
Resources, 18
operation, 54, 55, 83, 125
Action Status, 125
Constraint Expression, 54, 55
Notification Environment Variable, 83
Operation History, 124
Option
admin_epoch, 8
batch-limit, 9
clone-max, 78
clone-min, 78
clone-node-max, 78
cluster-delay, 10
cluster-ipc-limit, 11
cluster-recheck-interval, 10
concurrent-fencing, 10
Configuration Version, 8
dampen, 68
dc-deadtime, 10
election-timeout, 11
enable-startup-probes, 9
ePOCH, 8
failure-timeout, 23
globally-unique, 78
host_list, 68
interleave, 78
is-managed, 22
join-finalization-timeout, 12
join-integration-timeout, 12
maintenance-mode, 10
migration-limit, 9
migration-threshold, 23
multiple-active, 23
multiplier, 68
no-quorum-policy, 9
node-health-base, 11
node-health-green, 11
node-health-red, 11
node-health-strategy, 11
node-health-yellow, 11
notify, 78
num_updates, 8
ordered, 78
pe-error-series-max, 11
pe-input-series-max, 11
pe-warn-series-max, 11
placement-strategy, 11
priority, 22
promotable, 79
promoted-max, 79
promoted-node-max, 79
remove-after-stop, 11
requires, 22
resource-stickiness, 22
shutdown-escalation, 11
start-failure-is-fatal, 9
startup-fencing, 11
stonith-action, 10
stonith-enabled, 10
stonith-max-attempts, 10
stonith-timeout, 10
stonith-watchdog-timeout, 10
stop-all-resources, 9
Index

stop-orphan-actions, 9
stop-orphan-resources, 9
symmetric-cluster, 9
target-role, 22
timeout, 48
timestamp-format, 48
transition-delay, 12
validate-with, 8
options, 91, 92, 94
   Docker, 91
   rkt, 92
storage-mapping, 94
ordered, 78
   Clone Option, 78
Ordering, 34
   action, 81
   first, 35
   first-action, 35
   id, 35
   kind, 35
   role, 81
   rsc-role, 80
   then, 35
   then-action, 35
   with-rsc-role, 80
Ordering Constraints, 34, 35, 35, 35, 35, 35, 35, 35, 80, 80, 81, 81
   symmetrical, 35

P

pcmk_action_limit, 109
   Fencing, 109
pcmk_delay_base, 109
   Fencing, 109
pcmk_delay_max, 109
   Fencing, 109
pcmk_host_argument, 109
   Fencing, 109
pcmk_host_check, 109
   Fencing, 109
pcmk_host_list, 108
   Fencing, 108
pcmk_host_map, 108
   Fencing, 108
pcmk_list_action, 110
   Fencing, 110
pcmk_list_retries, 111
   Fencing, 111
pcmk_list_timeout, 110
   Fencing, 110
pcmk_monitor_action, 111
   Fencing, 111
pcmk_monitor_retries, 111
   Fencing, 111
pcmk_monitor_timeout, 111
   Fencing, 111
pcmk_off_action, 110
   Fencing, 110
pcmk_off_retries, 110
   Fencing, 110
pcmk_off_timeout, 110
   Fencing, 110
pcmk_reboot_action, 109
   Fencing, 109
pcmk_reboot_retries, 110
   Fencing, 110
pcmk_reboot_timeout, 109
   Fencing, 109
pcmk_status_action, 111
   Fencing, 111
pcmk_status_retries, 111
   Fencing, 111
pcmk_status_timeout, 111
   Fencing, 111
pe-error-series-max, 11
   Cluster Option, 11
pe-input-series-max, 11
   Cluster Option, 11
pe-warn-series-max, 11
   Cluster Option, 11
Ping Resource
   Option
dampen, 68
host_list, 68
multiplier, 68
Ping Resource Option, 68, 68, 68
   placement-strategy, 11
   Cluster Option, 11
port, 93
   port-mapping, 93
   port-mapping, 93, 93, 93, 93
Property
   id, 93
   internal-port, 93
   port, 93
   range, 93
priority, 22
   Resource Option, 22
probe_complete, 15
   Node attribute, 15
progressive, 72
Promotable, 77
promotable, 79
   Clone Option, 79
Promotable Clone Resources, 77
promoted-max, 79, 90, 91
   Clone Option, 79
   Docker, 90
rkt, 91
promoted-node-max, 79
Clone Option, 79
promote_resource, 85
Notification Environment Variable, 85
promote_uname, 85
Notification Environment Variable, 85
Property
    add-host, 92
cib-last-written, 8
class, 21
cluster-infrastructure, 9
control-port, 93
dc-uuid, 8
dc-version, 9
description, 90
enabled, 27, 27
have-quorum, 8
host-interface, 92
host-netmask, 92
id, 21, 26, 78, 90, 93, 93
image, 90, 91
internal-port, 93
interval, 26
ip-range-start, 92
name, 26
network, 91, 91
on-fail, 27
options, 91, 92, 94
pcmk_action_limit, 109
pcmk_delay_base, 109
pcmk_delay_max, 109
pcmk_host_argument, 109
pcmk_host_check, 109
pcmk_host_list, 108
pcmk_host_map, 108
pcmk_list_action, 110
pcmk_list_retries, 111
pcmk_list_timeout, 110
pcmk_monitor_action, 111
pcmk_monitor_retries, 111
pcmk_monitor_timeout, 111
pcmk_off_action, 110
pcmk_off_retries, 110
pcmk_off_timeout, 110
pcmk_reboot_action, 109
pcmk_reboot_retries, 110
pcmk_reboot_timeout, 109
pcmk_status_action, 111
pcmk_status_retries, 111
pcmk_status_timeout, 111
port, 93
promoted-max, 90, 91
provider, 21
provides, 108
range, 93
replicas, 90, 91
replicas-per-host, 90, 91
role, 27
run-command, 91, 92
source-dir, 94
source-dir-root, 94
stonith-timeout, 108, 109, 110
target-dir, 94
timeout, 26
type, 21
provider, 21
Resource, 21
provides, 108
Fencing, 108
Q
queue-time, 125
    Action Status, 125
R
range, 93
    port-mapping, 93
rc, 51
rc-code, 125
    Action Status, 125
recipient, 50
red, 71
remove-after-stop, 11
    Cluster Option, 11
replicas, 90, 91
    Docker, 90
    rkt, 91
replicas-per-host, 90, 91
    Docker, 90
    rkt, 91
require-all, 39
    Resource Sets, 39
requires, 22
    Resource Option, 22
Resource, 17, 21, 21, 21, 21
    Action, 26
    Alerts, 47
    bundle, 89
class, 17
Clones, 77
Constraint
    Attribute Expression, 54
    Date Specification, 56
    Date/Time Expression, 55
    Duration, 56
    Rule, 53
Index

Constraints, 31
  Colocation, 36
  Location, 32
  Ordering, 34
Group Property
  id, 76
Groups, 75
Location
  Determine by Rules, 58
Location Relative to other Resources, 36
LSB, 18
Moving, 66
Nagios Plugins, 20
OCF, 18
Option
  failure-timeout, 23
  is-managed, 22
  migration-threshold, 23
  multiple-active, 23
  priority, 22
  requires, 22
  resource-stickiness, 22
  target-role, 22
Promotable, 77
Property
  class, 21
  id, 21
  provider, 21
  type, 21
Start Order, 34
STONITH, 20
System Services, 20
Systemd, 19
Upstart, 19
Resource Discovery, 33
  Location Constraints, 33
Resource Option, 22, 22, 22, 22, 22, 22, 23, 23, 23
Resource Sets, 38, 39, 39, 39, 39, 39
  action, 39
  id, 38
  require-all, 39
  role, 39
  score, 39
  sequential, 38
  resource-discovery-enabled, 15
    Node attribute, 15
resource-stickiness, 22
  Clones, 82
  Groups, 77
  Resource Option, 22
Resources, 18, 18, 18, 18, 19, 19, 20, 20, 20, 20, 66
Return Code
  OCF_FAILED_MASTER, 83
  OCF_NOT_RUNNING, 83
OCF_RUNNING_MASTER, 83
OCF_SUCCESS, 83
rkt, 91, 91, 91, 91, 91, 92, 92
bundle, 89
Property
  image, 91
  network, 91
  options, 92
  promoted-max, 91
  replicas, 91
  replicas-per-host, 91
  run-command, 92
role, 27, 39, 53, 81
  Action Property, 27
  Constraint Rule, 53
  Ordering Constraints, 81
  Resource Sets, 39
rsc, 32, 37, 51
  Colocation Constraints, 37
  Location Constraints, 32
rsc-pattern, 32
  Location Constraints, 32
rsc-role, 80
  Ordering Constraints, 80
Rule, 53
  boolean-op, 53
  Controlling Cluster Options, 60
  Determine Resource Location, 58
  id, 53
  role, 53
  score, 53
  score-attribute, 53
  run-command, 91, 92
  Docker, 91
  rkt, 92
S
  score, 32, 37, 39, 53, 71
    Colocation Constraints, 37
    Constraint Rule, 53
    Location Constraints, 32
    Resource Sets, 39
    score-attribute, 53
    Constraint Rule, 53
  sequence, 51
  sequential, 38
    Resource Sets, 38
Setting Options with Rules, 60
shutdown, 15
  Node attribute, 15
shutdown-escalation, 11
  Cluster Option, 11
site-name, 15
  Node attribute, 15
slave_resource, 85
  Notification Environment Variable, 85
slave_uname, 85
  Notification Environment Variable, 85
source-dir, 94
  storage-mapping, 94
source-dir-root, 94
  storage-mapping, 94
standby, 15
  Node attribute, 15
start, 55
  Constraint Expression, 55
Start Order, 34
start-failure-is-fatal, 9
  Cluster Option, 9
startup-fencing, 11
  Cluster Option, 11
start_resource, 83
  Notification Environment Variable, 83
start_uname, 83
  Notification Environment Variable, 83
status, 51
Status, 123
  call-id, 125
crm-debug-origin, 124, 126
  crmd, 124
crm_feature_set, 126
eexec-time, 125
expected, 124
id, 123, 125
interval, 125
in_ccm, 124
join, 124
last-rc-change, 125
last-run, 125
op-digest, 126
op-status, 125
operation, 125
queue-time, 125
rc-code, 125
transition-key, 126
transition-magic, 126
uname, 123
Status of a Node, 123
STONITH, 20
  Configuration, 107
  Resources, 20
stonith-action, 10
  Cluster Option, 10
stonith-enabled, 10
  Cluster Option, 10
stonith-max-attempts, 10
  Cluster Option, 10
stonith-timeout, 10, 108, 109, 110
  Cluster Option, 10
  Fencing, 108, 109, 110
stonith-watchdog-timeout, 10
  Cluster Option, 10
stop-all-resources, 9
  Cluster Option, 9
stop-orphan-actions, 9
  Cluster Option, 9
stop-orphan-resources, 9
  Cluster Option, 9
stop_resource, 83
  Notification Environment Variable, 83
stop_uname, 83
  Notification Environment Variable, 83
storage
  storage-mapping, 93
storage-mapping, 93, 94, 94, 94, 94
  Property
    id, 93
    options, 94
    source-dir, 94
    source-dir-root, 94
    target-dir, 94
  symmetric-cluster, 9
    Cluster Option, 9
symmetrical, 35
  Ordering Constraints, 35
Symmetrical Opt-Out, 34
Symmetrical Opt-Out Clusters, 34
System Service
  Resources, 20
System Services, 20
Systemd, 19
    Resources, 19
T
  target, 116
    fencing-level, 116
    target-attribute, 116, 116
    fencing-level, 116, 116
target-dir, 94
  storage-mapping, 94
target-pattern, 116
  fencing-level, 116
target-role, 22
  Resource Option, 22
target_rc, 51
task, 51
terminate, 15
  Node attribute, 15
then, 35
  Ordering Constraints, 35
then-action, 35
  Ordering Constraints, 35
Index

Time Based Expressions, 55
timeout, 26, 48
  Action Property, 26
timestamp, 51
timestamp-format, 48
timestamp_epoch, 51
timestamp_usec, 51
transition-delay, 12
  Cluster Option, 12
transition-key, 126
  Action Status, 126
transition-magic, 126
  Action Status, 126
type, 21, 54, 83
  Constraint Expression, 54
  Notification Environment Variable, 83
  Resource, 21

U
uname, 123
  Node Status, 123
Upstart, 19
  Resources, 19

V
validate-with, 8
  Cluster Option, 8
value, 54, 54
  Constraint Expression, 54, 54
version, 50

W
weekdays, 56
  Date Specification, 56
weeks, 56
  Date Specification, 56
weekyears, 56
  Date Specification, 56
with-rsc, 37
  Colocation Constraints, 37
with-rsc-role, 80
  Ordering Constraints, 80

Y
yeardays, 56
  Date Specification, 56
years, 56
  Date Specification, 56
yellow, 71