Red Hat Enterprise Linux 6 Configuring the Red Hat High Availability Add-On with Pacemaker

Reference Document for the High Availability Add-On for Red Hat Enterprise Linux 6
Edition 1

Red Hat Enterprise Linux 6 Configuring the Red Hat High Availability Add-On with Pacemaker

---

Reference Document for the High Availability Add-On for Red Hat Enterprise Linux 6
Edition 1

Abstract

Red Hat High Availability Add-On Reference provides information on configuring the Red Hat High Availability Add-On using Pacemaker.

Table of Contents

Introduction

This document provides information about installing, configuring and managing Red Hat High Availability Add-On components. Red Hat High Availability Add-On components allow you to connect a group of computers (called *nodes* or *members*) to work together as a cluster. In this document, the use of the word *cluster* or *clusters* is used to refer to a group of computers running the Red Hat High Availability Add-On.

The audience of this document should have advanced working knowledge of Red Hat Enterprise Linux and understand the concepts of clusters, storage, and server computing.

For more information about Red Hat Enterprise Linux 6, refer to the following resources:

- *Red Hat Enterprise Linux Installation Guide* — Provides information regarding installation of Red Hat Enterprise Linux 6.
- *Red Hat Enterprise Linux Deployment Guide* — Provides information regarding the deployment, configuration and administration of Red Hat Enterprise Linux 6.

For more information about the High Availability Add-On and related products for Red Hat Enterprise Linux 6, refer to the following resources:

- *High Availability Add-On Overview* — Provides a high-level overview of the Red Hat High Availability Add-On.
- *Cluster Administration* — Provides information about installing, configuring and managing the High Availability Add-On.
- *Logical Volume Manager Administration* — Provides a description of the Logical Volume Manager (LVM), including information on running LVM in a clustered environment.
- *Global File System 2: Configuration and Administration* — Provides information about installing, configuring, and maintaining Red Hat GFS2 (Red Hat Global File System 2), which is included in the Resilient Storage Add-On.
- *DM Multipath* — Provides information about using the Device-Mapper Multipath feature of Red Hat Enterprise Linux 6.
- *Load Balancer Administration* — Provides information on configuring high-performance systems and services with the Load Balancer Add-On, a set of integrated software components that provide Linux Virtual Servers (LVS) for balancing IP load across a set of real servers.
- *Release Notes* — Provides information about the current release of Red Hat products.

Red Hat Cluster Suite documentation and other Red Hat documents are available in HTML, PDF, and RPM versions on the Red Hat Enterprise Linux Documentation CD and online at https://access.redhat.com/site/documentation/.

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 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:

> 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 submenu 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:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

```
static int kvm_vm_ioctl_deassign_device(struct kvm *kvm,
                  struct kvm_assigned_pci_dev *assigned_dev)
{
         int r = 0;
         struct kvm_assigned_dev_kernel *match;

         mutex_lock(&kvm->lock);

         match = kvm_find_assigned_dev(&kvm->arch.assigned_dev_head,
                                       assigned_dev->assigned_dev_id);
         if (!match) {
                 printk(KERN_INFO "%s: device hasn't been assigned before, "
                   "so cannot be deassigned\n", __func__);
                 r = -EINVAL;
                 goto out;
         }

         kvm_deassign_device(kvm, match);

         kvm_free_assigned_device(kvm, match);

out:
         mutex_unlock(&kvm->lock);
         return r;
}
```

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. Feedback

If you spot a typo, 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: http://bugzilla.redhat.com/bugzilla/. File the bug against the product **Red Hat Enterprise Linux 6** and the component **doc-Cluster_General**.

Be sure to mention the manual identifier:

```
Configuring_High_Availability_With_Pacemaker(EN)-6 (2013-11-13T16:26)
```

By mentioning this manual's identifier, we know exactly which version of the guide you have.

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

Chapter 1. Red Hat High Availability Add-On Configuration and Management Reference Overview

This document provides descriptions of the options and features that the Red Hat High Availability Add-On using Pacemaker supports.

This manual documents the use of the **pcs** configuration interface for the Red Hat Enterprixe Linux Release 6.5 and later.

Chapter 2. Pacemaker Cluster Properties

Cluster properties control how the cluster behaves when confronted with situations that may occur during cluster operation.

- Table 2.1, “Cluster Properties” describes the cluster properties options.
- Section 2.2, “Setting and Removing Cluster Properties” describes how to set cluster properties.
- Section 2.3, “Querying Cluster Property Settings” describes how to list the currently set cluster properties.

2.1. Summary of Cluster Properties and Options

Table 2.1, “Cluster Properties” summaries the Pacemaker cluster properties, showing the default values of the properties and the possible values you can set for those properties.

Table 2.1. Cluster Properties

Option	Default	Description
batch-limit	30	The number of jobs that the TE is allowed to execute in parallel. The "correct" value will depend on the speed and load of your network and cluster nodes.
migration-limit	-1 (unlimited)	The number of migration jobs that the TE is allowed to execute in parallel on a node.
no-quorum-policy	stop	What to do when the cluster does not have quorum. Allowed values: * ignore - continue all resource management * freeze - continue resource management, but do not recover resources from nodes not in the affected partition * stop - stop all resources in the affected cluster partition * suicide - fence all nodes in the affected cluster partition
symmetric-cluster	**true**	Indicates whether resources can run on any node by default.
stonith-enabled	**true**	Indicates that failed nodes and nodes with resources that can not be stopped should be fenced. Protecting your data requires that you set this **true**. If **true**, or unset, the cluster will refuse to start resources unless one or more STONITH resources have been configured also.
stonith-action	reboot	Action to send to STONITH device. Allowed values: **reboot**, **off**. The value **poweroff** is also allowed, but is only used for legacy devices.
cluster-delay	60s	Round trip delay over the network (excluding action execution). The "correct" value will depend on the speed and load of your network and cluster nodes.
stop-orphan-resources	**true**	Indicates whether deleted resources should be stopped.
stop-orphan-actions	**true**	Indicates whether deleted actions should be cancelled.
start-failure-is-fatal	**true**	When set to **false**, the cluster will instead use the resource's **failcount** and value for **resource-failure-stickiness**.
pe-error-series-max	-1 (all)	The number of PE inputs resulting in ERRORs to save. Used when reporting problems.
pe-warn-series-max	-1 (all)	The number of PE inputs resulting in WARNINGs to save. Used when reporting problems.
pe-input-series-max	-1 (all)	The number of "normal" PE inputs to save. Used when reporting problems.

2.2. Setting and Removing Cluster Properties

To set the value of a cluster property, use the following **pcs** command.

```
pcs property set property=value
```

For example, to set the value of **symmetric-cluster** to **false**, use the following command.

```
# pcs property set symmetric-cluster=false
```

You can remove a cluster property from the configuration with the following command.

```
pcs property unset property
```

Alternately, you can remove a cluster property from a configuration by leaving the value field of the **pcs property set** command blank. This restores that property to its default value. For example, if you have previously set the **symmetric-cluster** property to **false**, the following command removes the value you have set from the configuration and restores the value of **symmetric-cluster** to **true**, which is its default value.

```
# pcs property set stonith-enabled=
```

2.3. Querying Cluster Property Settings

In most cases, when you use the **pcs** command to display values of the various cluster components, you can use **pcs list** or **pcs show** interchangeably. In the following examples, **pcs list** is the format used to display an entire list of all settings for more than one property, while **pcs show** is the format used to display the values of a specific property.

To display the values of the property settings that have been set for the cluster, use the following **pcs** command.

```
pcs property list
```

To display all of the values of the property settings for the cluster, including the default values of the property settings that have not been explicitly set, use the following command.

```
pcs property list --all
```

To display the current value of a specific cluster property, use the following command.

```
pcs property show property
```

For example, to display the current value of the **no-quorum-policy**, execute the following command:

```
# pcs property list no-quorum-policy
Cluster Properties:
 no-quorum-policy: ignore
```

For informational purposes, you can display a list of all of the default values for the properties, whether they have been set to a value other than the default or not, by using the following command.

```
pcs property [list|show] --defaults
```

Chapter 3. Cluster Creation and Administration

This chapter describes how to perform basic cluster administration with Pacemaker, including creating the cluster, managing the cluster components, and displaying cluster status.

3.1. Cluster Creation

To create a running cluster, perform the following steps:

1. Configure and sync the cluster nodes.
2. Start cluster services on the cluster nodes.

The following sections described the commands that you use to perform these steps.

3.2. Configuring and Starting the Cluster Nodes

The format of the command that configures the cluster configuration file and syncs the configuration to the specified nodes is as follows. If you specify the **--start** option, the command will also start the cluster services on the local node. If necessary, you can also start the cluster services with a separate **pcs cluster start** command. The names of the nodes must match the hostnames associated with the IP address of the network interface that is used for cluster communication on each node.

```
pcs cluster setup [--start] cluster_name node1 [node2] [...]
```

For example, the following command configures the cluster configuration file for a cluster named **mycluster** that consists of the nodes **mynode1** and **mynode2** and syncs the configuration to the nodes.

```
# pcs cluster setup mycluster mynode1 mynode2
```

3.3. Managing Cluster Nodes

The following sections describe the commands you use to manage cluster nodes, including commands to start and stop cluster services and to add and remove cluster nodes.

3.3.1. Stopping Cluster Services

The following command stops cluster services on the local node.

```
pcs cluster stop
```

You can force a stop of cluster services on the local node with the following command, which performs a **kill -9** command.

```
pcs cluster kill
```

3.3.2. Enabling and Disabling Cluster Services

Use the following command to configure the cluster services to run on startup on the current node.

```
pcs cluster enable
```

Use the following command to configure the cluster services not to run on startup on the current node.

```
pcs cluster disable
```

3.3.3. Adding and Removing Cluster Nodes

To add and remove nodes you must run **pcs cluster setup** on all nodes.

3.3.4. Standby Mode

The following command puts the specified node into standby mode. The specified node is no longer able to host resources. Any resources currently active on the node will be moved to another node. If you specify the **--all**, this command puts all nodes into standby mode.

You can use this command when updating a resource's packages. You can also use this command when testing a configuration, to simulate recovery without actually shutting down a node.

```
pcs cluster standby node | --all
```

The following command removes the specified node from standby mode. After running this command, the specified node is then able to host resources. If you specify the **--all**, this command removes all nodes from standby mode.

```
pcs cluster unstandby node | --all
```

Note that when you execute the **pcs cluster standby** command, this adds constraints to the resources to prevent them from running on the indicated node. When you execute the **pcs cluster unstandby** command, this removes the constraints. This does not necessarily move the resources back to the indicated node; where the resources can run at that point depends on how you have configured your resources initially. For information on resource constraints, refer to Chapter 5, *Resource Constraints*.

3.4. Removing the Cluster Configuration

To remove all cluster configuration files and stop all cluster services on the local node, thus permanently destroying a cluster, use the following command. You must run this node seprately on each node in the cluster.

Warning

This command permantly removes any cluster configuration that has been created on the local node. It is recommended that you run **pcs cluster stop** before destroying the cluster.

```
pcs cluster destroy
```

3.5. Displaying Cluster Status

The following command displays the current status of the cluster and the cluster resources.

```
# pcs cluster status
```

You can display a subset of information about the current status of the cluster with the following commands.

The following command displays the status of the cluster, but not the cluster resources.

```
# pcs cluster status cluster
```

The following command displays the status of the cluster resources.

```
# pcs cluster status resources
```

Chapter 4. Configuring Cluster Resources

This chapter provides information on configuring resources in a cluster.

4.1. Resource Creation

Use the following command to create a cluster resource.

```
pcs resource create resource_id standard:provider:type|type [resource options]
```

For example, the following command creates a resource with the name ClusterIP of standard **ocf**, provider **heartbeat**, and type **IPaddr2**. The floating address of this resource is 192.168.0.120, the system will check whether the resource is running every 30 seconds.

```
# pcs resource create ClusterIP ocf:heartbeat:IPaddr2 ip=192.168.0.120
cidr_netmask=24 op monitor interval=30s
```

Use the following command to delete a configured resource.

```
pcs resource delete resource_id
```

For example, the following command deletes an existing resource with a resource ID of **ClusterIP**

```
# pcs resource delete ClusterIP
```

- For information on the ***resource_id***, ***standard***, ***provider***, and ***type*** fields of the **pcs resource create** command, refer to Section 4.2, “Resource Properties”.
- For information on defining resource parameters for individual resources, refer to Section 4.3, “Resource-Specific Parameters”.
- For information on defining resource meta options, which are used by the cluster to decide how a resource should behave, refer to Section 4.4, “Resource Meta Options”.
- For information on defining the operations to perform on a resource, refer to Section 4.5, “Resource Operations”.

4.2. Resource Properties

The properties that you define for a resource tell the cluster which script to use for the resource, where to find that script and what standards it conforms to. Table 4.1, “Resource Properties” describes these properties.

Table 4.1. Resource Properties

Field	Description
resource id	Your name for the resource
standard	The standard the script conforms to. Allowed values: **ocf**, **service**, **upstart**, **systemd**, **lsb**, **stonith**
type	The name of the Resource Agent you wish to use, for example **IPaddr** or **Filesystem**
provider	The OCF spec allows multiple vendors to supply the same ResourceAgent. Most of the agents shipped by Red Hat use **heartbeat** as the provider.

Table 4.2, “Commands to Display Resource Properties”. summarizes the commands that display the available resource properties. you can use to create a resource.

Table 4.2. Commands to Display Resource Properties

pcs Display Command	Output
pcs resource list	Displays a list of all available resources.
pcs resource standards	Displays a list of available resources agent standards.
pcs resource providers	Displays a list of available resources agent providers.
pcs resource list *string*	Displays a list of available resources filtered by the specified string. You can use this command to display resources filtered by the name of a standard, a provider, or a type.

4.3. Resource-Specific Parameters

For any individual resource, you can use the following command to display the parameters you can set for that resources.

```
# pcs resource describe standard:provider:type|type
```

For example, the following command display the parameters you can set for a resource of type **LVM**.

```
# pcs resource describe LVM
Resource options for: LVM
  volgrpname (required): The name of volume group.
  exclusive: If set, the volume group will be activated exclusively.
  partial_activation: If set, the volume group will be activated even
  only partial of the physicalvolumes available. It helps to set to
  true, when you are using mirroring logical volumes.
```

4.4. Resource Meta Options

In addition to the resource-specific parameters, you can configure additional resource options for any resource. These options are used by the cluster to decide how your resource should behave. Table 4.3, “Resource Meta Options” describes this options.

Table 4.3. Resource Meta Options

Field	Default	Description
priority	**0**	If not all resources can be active, the cluster will stop lower priority resources in order to keep higher priority ones active.
target-role	**Started**	What state should the cluster attempt to keep this resource in? Allowed values: * *Stopped* - Force the resource to be stopped * *Started* - Allow the resource to be started (In the case of multistate resources, they will not promoted to master) * *Master* - Allow the resource to be started and, if appropriate, promoted
is-managed	**true**	Is the cluster allowed to start and stop the resource? Allowed values: **true**, **false**
resource-stickiness	0	Value to indicate how much the resource prefers to stay where it is.
requires	Calculated	Inicates under what conditions can the resource be started. Defaults to **fencing** unless **stonith-enabled** is **false** or **standard** is **stonith** - under those conditions the default is **quorum**. Possible values: * **nothing** - can always be started * **quorum** - The cluster can only start this resource if a majority of the configured nodes are active * **fencing** - The cluster can only start this resource if a majority of the configured nodes are active *and* any failed or unknown nodes have been powered off. * **unfencing** - The cluster can only start this resource if a majority of the configured nodes are active *and* any failed or unknown nodes have been powered off *and* only on nodes that have been *unfenced*
migration-threshold	**INFINITY** (disabled)	How many failures may occur for this resource on a node, before this node is marked ineligible to host this resource. For information on configuring the **migration-threshold** option, refer to Section 8.2, “Moving Resources Due to Failure”.
failure-timeout	**0** (disabled)	Used in conjunction with the **migration-threshold** option, indicates 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. For information on configuring

		the **failure-timeout** option, refer to Section 8.2, "Moving Resources Due to Failure".
multiple-active	**stop_start**	What should the cluster do if it ever finds the resource active on more than one node. Allowed values: * **block** - mark the resource as unmanaged * **stop_only** - stop all active instances and leave them that way * **stop_start** - stop all active instances and start the resource in one location only

To change the default value of a resource meta option, use the following command.

```
pcs resource defaults options
```

For example, the following command resets the default value of **resource-stickiness** to 100.

```
# pcs resource defaults resource-stickiness=100
```

Omitting the ***options*** parameter from the **pcs resource defaults** displays a list of currently configured default values for resource meta options. The following example shows the output of this command after you have reset the default value of **resource-stickiness** to 100.

```
# pcs resource defaults
resource-stickiness: 100
```

Whether you have reset the default value of a resource meta option or not, you can set a resource option for a particular resource to a value other than the default when you create the resource. The following shows the format of the **pcs resource create** command you use when specifying a value for a resource meta option.

```
pcs resource create resource_id standard:provider:type|type [resource options]
[meta meta_options...]
```

For example, the following command creates a resource with a **resource-stickiness** value of 50.

```
# pcs resource create ClusterIP ocf:heartbeat:IPaddr2 ip=192.168.0.120
cidr_netmask=24 meta resource-stickiness=50
```

You can also set the value of a resource meta option for an existing resource, group, cloned resource, or master resource with the following command.

```
pcs resource meta resource_id | group_id | clone_id | master_id  meta_options
```

For information on resource clone meta options, see Section 9.1, "Resource Clones". For information on resource master meta options, see Section 9.2, "Multi-State Resources: Resources That Have Multiple Modes".

4.5. Resource Operations

To ensure that resources remain healthy, you can add a monitoring operation to a resource's definition. If you do not specify a monitoring operation for a resource, by default the **pcs** command will create a monitor operation with a 60 second interval.

Table 4.4, "Properties of an Operation" summarizes the properties of a resource monitoring operation.

Table 4.4. Properties of an Operation

Field	Description
id	Unique name for the action. The system assigns this when you configure an operation.
name	The action to perform. Common values: **monitor**, **start**, **stop**
interval	How frequently (in seconds) to perform the operation. Default value: **0**, meaning never.
timeout	How long to wait before declaring the action has failed. If you find that your system includes a resource that takes a long time to start or stop or perform a non-recurring monitor action at startup, and requires more time than the system allows before declaring that the start action has failed, you can increase this value from the default of 20 or the value of **timeout** in "op defaults".
on-fail	The action to take if this action ever fails. Allowed values: * **ignore** - Pretend the resource did not fail * **block** - Do not perform any further operations on the resource * **stop** - Stop the resource and do not start it elsewhere * **restart** - Stop the resource and start it again (possibly on a different node) * **fence** - STONITH the node on which the resource failed * **standby** - Move *all* resources away from the node on which the resource failed The default for the **stop** operation is **fence** when STONITH is enabled and **block** otherwise. All other operations default to **stop**.
enabled	If **false**, the operation is treated as if it does not exist. Allowed values: **true**, **false**

You can configure monitoring operations when you create a resource, using the following command.

```
pcs resource create resource_id standard:provider:type|type [resource_options]
op operation_action operation_options [operation_type operation_options]...]
```

For example, the following command creates an **IPaddr2** resource with a monitoring operation. The new resource is called **ClusterIP** with an IP address of 192.168.0.99 and a netmask of 24 on **eth2**. A monitoring operation will be performed every 30 seconds.

```
# pcs resource create ClusterIP ocf:heartbeat:IPaddr2 ip=192.168.0.99
cidr_netmask=24 nic=eth2 op monitor interval=30s
```

Alternately, you can add a monitoring operation to an existing resource with the following command.

```
pcs resource op add resource_id operation_action [operation_properties]
```

To set global default values for monitoring operations, use the following command.

```
pcs resource op defaults [options]
```

For example, the following command sets a global default of a **timeout** value of 240s for all monitoring operations.

```
# pcs resource op defaults timeout=240s
```

To display the currently configured default values for monitoring operations, do not specify any options when you execute the **pcs resource op defaults** commmand.

For example, following command displays the default monitoring operation values for a cluster which has been configured with a **timeout** value of 240s.

```
# pcs resource op defaults
timeout: 240s
```

Use the following command to delete a configured resource operation.

```
pcs resource op remove resource_id operation_name [operation_properties]
```

Note

You must specify the exact operation properties to properly remove an existing operation.

4.6. Displaying Configured Resources

To display a list of all configured resources, use the following command.

```
pcs resource show
```

For example, if your system is configured with a resource named **ClusterIP** and a resource named **WebSite**, the **pcs resource show** command yields the following output.

```
# pcs resource show
 ClusterIP (ocf::heartbeat:IPaddr2): Started
 WebSite (ocf::heartbeat:apache): Started
```

To display a list of all configured resources and the parameters configured for those resources, use the **--full** option of the the **pcs resource show** command, as in the following example.

```
# pcs resource show --full
 Resource: ClusterIP (type=IPaddr2 class=ocf provider=heartbeat)
  Attributes: ip=192.168.0.120 cidr_netmask=24
  Operations: monitor interval=30s
 Resource: WebSite (type=apache class=ocf provider=heartbeat)
  Attributes: statusurl=http://localhost/server-status
configfile=/etc/httpd/conf/httpd.conf
  Operations: monitor interval=1min
```

To display the configured parameters for a resource, use the following command.

```
pcs resource show resource_id
```

For example, the following command displays the currently configured parameters for resource **ClusterIP**.

```
# pcs resource show ClusterIP
 Resource: ClusterIP (type=IPaddr2 class=ocf provider=heartbeat)
  Attributes: ip=192.168.0.120 cidr_netmask=24
  Operations: monitor interval=30s
```

4.7. Modifying Resource Parameters

To modify the parameters of a configured resource, use the following command.

```
pcs resource update resource_id [resource_options]
```

The following sequence of commands show the initial values of the configured parameters for resource **ClusterIP**, the command to change the value of the **ip** parameter, and the values following the update command.

```
# pcs resource show ClusterIP
 Resource: ClusterIP (type=IPaddr2 class=ocf provider=heartbeat)
  Attributes: ip=192.168.0.120 cidr_netmask=24
  Operations: monitor interval=30s
# pcs resource update ClusterIP ip=192.169.0.120
# pcs resource show ClusterIP
 Resource: ClusterIP (type=IPaddr2 class=ocf provider=heartbeat)
  Attributes: ip=192.169.0.120 cidr_netmask=24
  Operations: monitor interval=30s
```

4.8. Multiple Monitoring Operations

You can configure a single resource with 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. When configuring multiple monitor operations, however, you must ensure that no two operations are performed at the same interval.

4.9. Enabling and Disabling Cluster Resources

The following command enables the resource specified by ***resource_id***.

```
pcs enable resource_id
```

The following command disables the resource specified by ***resource_id***.

```
pcs disable resource_id
```

Chapter 5. Resource Constraints

You can determine the behavior of a resource in a cluster by configuring constraints for that resource. You can configure the following categories of constraints:

- **location** constraints — A location constraint determines which nodes a resource can run on. Location constraints are described in Section 5.1, “Location Constraints”.
- **order** constraints — An order constraint determines the order in which the resources run. Order consraints are described in Section 5.2, “Order Constraints”.
- **colocation** constraints — A colocation constraint determines where resources will be placed relative to other resources. Colocation constraints are described in Section 5.3, “Colocation of Resources”.

As a shorthand for configuring a set of constraints that will locate a set of resources together and ensure that the resources start sequentially and stop in reverse order, Pacemaker supports the concept of resource groups. For information on resource groups, see Section 5.5, “Resource Groups”.

5.1. Location Constraints

Location constraints determine which nodes a resource can run on. You con configure location constraints to determine whether a resource will prefer or avoid a specified node.

Table 5.1, “Location Constraint Options”. summarizes the options for configuring location constraints.

Table 5.1. Location Constraint Options

Field	Description
id	A unique name for the constraint. This is set by the system when you configure a location constraint with **pcs**.
rsc	A resource name
node	A node’s name
score	Value to indicate the preference for whether a resource should run on or avoid a node.

The following command creates a location constraint for a resource to prefer the specified node or nodes.

```
pcs constraint location rsc prefers node[=score] ...
```

The following command creates a location constraint for a resource to avoid the specified node or nodes.

```
pcs constraint location rsc avoids node[=score] ...
```

There are two alternative strategies for specifying which nodes a resources can run on:

- Opt-In Clusters — Configure a cluster in which, by default, no resource can run anywhere and then selectively enable allowed nodes for specific resources. The procedure for configuring an opt-in cluster is described in Section 5.1.1, “Configuring an "Opt-In" Cluster”.
- Opt-Out Clusters — Configure a cluster in which, by default, all resources an run anywhere and then create location constraints for resources that are not allowed to run on specific nodes. The procedure for configuring an opt-out cluster is described in Section 5.1.2, “Configuring an "Opt-Out"

Cluster".

Whether you should choose to configure an opt-in or opt-out cluster depends both 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.

5.1.1. Configuring an "Opt-In" Cluster

To create an opt-in cluster, set the **symmetric-cluster** cluster property to **false** to prevent resources from running anywhere by default.

```
# pcs property set symmetric-cluster=false
```

Enable nodes for individual resources. The following commands configure location constraints so that the resource **Webserver** prefers node **example-1**, the resource **Database** prefers node **example-2**, and both resources can fail over to node **example-3** if their preferred node fails.

```
# pcs constraint location Webserver prefers example-1=200
# pcs constraint location Webserver prefers example-3=0
# pcs constraint location Database prefers example-2=200
# pcs constraint location Database prefers example-3=0
```

5.1.2. Configuring an "Opt-Out" Cluster

To create an opt-out cluster, set the **symmetric-cluster** cluster property to **true** to allow resources to run everywhere by default.

```
# pcs property set symmetric-cluster=true
```

The following commands will then yield a configuration that is equivalent to the example in Section 5.1.1, "Configuring an "Opt-In" Cluster".

```
# pcs constraint location Webserver prefers example-1=200
# pcs constraint location Webserver avoids example-2=INFINITY
# pcs constraint location Database avoids example-1=INFINITY
# pcs constraint location Database prefers example-2=200
```

Note that it is not necessary to specify a score of INFINITY in these commands, since that is the default value for the score.

5.2. Order Constraints

Order constraints determine the order in which the resources run. You can configure an order constraint to determine the order in which resources start and stop.

Use the following command to configure an order constraint.

```
pcs constraint order [action] resource_id then [action] resource_id [options]
```

Table 5.2, "Properties of an Order Constraint". summarizes the properties and options for configuring order constraints.

Table 5.2. Properties of an Order Constraint

Field	Description
resource_id	The name of a resource on which an action is performed.
action	The action to perform on the first resource specified an action is performed on the second resource specified. Possible values of the ***action*** property are as follows: * **start** - Start the resource. * **stop** - Stop the resource. * **promote** - Promote the resource from a slave resource to a master resource. * **demote** - Demote the resource from a master resource to a slave resource. If no action is specified, the default action is **start**. For information on master and slave resources, refer to Section 9.2, "Multi-State Resources: Resources That Have Multiple Modes".
kind option	How to enforce the constraint. The possible values of the **kind** option are as follows: * **Optional** - Only applies if both resources are starting and/or stopping. For information on optional ordering, refer to Section 5.2.2, "Advisory Ordering". * **Mandatory** - Always. If the first resource you specified is stopping or cannot be started, the second resource you specified must be stopped. For information on mandatory ordering, refer to Section 5.2.1, "Mandatory Ordering (Default)". This is the default value. * **Serialize** - Ensure that no two stop/start actions occur concurrently for a set of resources.
symmetrical options	If true, which is the default, stop the resources in the reverse order. Default value: **true**

5.2.1. Mandatory Ordering (Default)

Use mandatory constraints when the second resource you specify cannot run without the first resource you specify being active. To specify that a constraint is mandatory, specify the **kind=Mandatory** option for the **pcs constraint order** command. This will ensure that the second resource you specify will react when the first resource you specify changes state.

- If the first resource you specified resource was running and is stopped, the second resource you specified will also be stopped (if it is running).
- If the first resource you specified resource was not running and cannot be started, the second resource you specified will be stopped (if it is running).
- If the first resource you specified is (re)started while the second resource you specified is running, the second resource you specified will be stopped and restarted.

5.2.2. Advisory Ordering

When the **kind=Optional** option is specified for an order constraint, the constraint is considered optional and only has an effect when both resources are stopping and/or starting. Any change in state of the first resource you specified has no effect on the second resource you specified.

5.2.3. Ordered Resource Sets

A common situation is for an administrator to create a chain of ordered resources, where, for example, resource A starts before resource B which starts before resource C. You can configure a chain of ordered resources with the following command. The resources will start in the specified order.

```
pcs constraint order set resource1 resource2 [resourceN]... [options] [set
resource1 resource2 ...]
```

5.2.4. Removing Resources From Ordering Constraints

Use the following command to remove resources from any ordering constraint.

```
pcs constraint order remove resource1 [resourceN]...
```

5.3. Colocation of Resources

A colocation constraint determines that the location of one resource depends on the location of another resource.

There is an important side effect of creating a colocation constraint between two resources: it affects the order in which resources are assigned to a node. This is because you cannot place resource A relative to resource B unless you know where resource B is. So when you are creating colocation constraints, it is important to consider whether you should colocate resource A with resource B or resource B with resource A.

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

The following command creates a colocation constraint.

```
pcs constraint colocation add [master|slave] source_resource with [master|slave]
target_resource [score] [options]
```

For information on master and slave resources, see Section 9.2, “Multi-State Resources: Resources That Have Multiple Modes”.

Table 5.3, “Properties of a Colocation Constraint”. summarizes the properties and options for configuring colocation constraints.

Table 5.3. Properties of a Colocation Constraint

Field	Description
source_resource	The colocation source. If the constraint cannot be satisfied, the cluster may decide not to allow the resource to run at all.
target_resource	The colocation target. The cluster will decide where to put this resource first and then decide where to put the source resource.
score	Positive values indicate the resource should run on the same node. Negative values indicate the resources should not run on the same node. A value of + **INFINITY**, the default value, indicates that the ***source_resource*** must run on the same node as the ***target_resource***. A value of - **INFINITY** indicates that the ***source_resource*** must not run on the same node as the ***target_resource***.

5.3.1. Mandatory Placement

Mandatory placement occurs any time the constraint's score is **+INFINITY** or **-INFINITY**. In such cases, if the constraint cannot be satisfied, then the ***source_resource*** is not permitted to run. For **score=INFINITY**, this includes cases where the ***target_resource*** is not active.

If you need **myresource1** to always run on the same machine as **myresource2**, you would add the following constraint:

```
# pcs constraint colocation add myresource1 with myresource2 score=INFINITY
```

Because **INFINITY** was used, if **myresource2** cannot run on any of the cluster nodes (for whatever reason) then **myresource1** will not be allowed to run.

Alternatively, you may want to configure the opposite, a cluster in which **myresource1** cannot run on the same machine as **myresource2**. In this case use **score=-INFINITY**

```
# pcs constraint colocation add myresource1 myresource2 with score=-INFINITY
```

Again, by specifying **-INFINITY**, the constraint is binding. So if the only place left to run is where **myresource2** already is, then **myresource1** may not run anywhere.

5.3.2. 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 **-INFINITY** and less than **INFINITY**, the cluster will try and accommodate your wishes but may ignore them if the alternative is to stop some of the cluster resources. Advisory colocation constraints can combine with other elements of the configuration to behave as if they were mandatory.

5.3.3. Colocating Sets of Resources

Use the following command to create a colocation constraint on a set of resources. You can set the **sequential** option to **true** or **false** to indicate whether the set of colocated resources is an ordered set.

```
colocation set resource1 resource2 [resourceN]... [setoptions name=value] ... [set
resourceX resourceY ...] [setoptions name=value...]
```

You can set the **role** option for a colocation set to **master** or **slave**. For information on multi-state resources, see Section 9.2, "Multi-State Resources: Resources That Have Multiple Modes".

5.3.4. Removing Colocation Constraints

Use the following command to remove colocation constraints with *source_resource*.

```
pcs constraint colocation remove source_resource target_resource
```

5.4. Displaying Constraints

There are a several commands you can use to display constraints that have been configured.

The following command list all current location, order, and colocation constraints.

```
pcs constraint list|show
```

The following command lists all current location constraints.

- If **resources** is specified, location constraints are displayed per resource. This is the default behavior.
- If **nodes** is specified, location constraints are displayed per node.
- If specific resources or nodes are specified, then only information about those resources or nodes is displayed.
- If the **--full** option is specified, show the internal constraint IDs.

```
pcs constraint location [show resources|nodes [specific_nodes|resources]] [--full]
```

The following command lists all current ordering constraints. If the **--full** option is specified, show the internal constraint IDs.

```
pcs constraint order show [--full]
```

The following command lists all current colocation constraints. If the **--full** option is specified, show the internal constraint IDs.

```
pcs constraint colocation show [--full]
```

The following command lists the constraints that reference specific resources.

```
pcs constraint ref [resource] ...
```

5.5. Resource Groups

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, Pacemaker supports the concept of groups.

You create a resource group with the following command, specifying the resources to include in the group. If the group does not exist, this command creates the group. If the group exists, this command adds additional resources to the group. The resources will start in the order you specify them with this command, and will stop in the reverse order.

```
pcs resource group add group_name resource_id...
```

You can also add a new resource to an existing group when you create the resource, using the following command. The resource you create is added to the group named ***group_name***.

```
pcs resource create resource_id standard:provider:type|type [resource_options]
[op operation_action operation_options [operation_action operation_options --
group group_name
```

You remove a resource from a group with the following command. If there are no resources in the group, this command removes the group itself.

```
pcs resource group remove group_name resource_id...
```

The following command lists all currently configured resource groups.

```
pcs resource group list
```

The following example creates a resource group named **shortcut** that contains the existing resources **IPaddr** and **Email**.

```
# pcs resource group add shortcut IPaddr Email
```

There is no limit to the number of resources a group can contain. The fundamental properties of a group are as follows.

- Resources are started in the order in which you specify them (in this example, **Public-IP** first, then **Email**).
- Resources are stopped in the reverse order in which you specify them. (**Email** first, then **Public-IP**).

If a resource in the group cannot run anywhere, then no resource specified after that resource is allowed to run.

- If **Public-IP** cannot run anywhere, neither can **Email**.
- If **Email** cannot run anywhere, however, this does not affect **Public-IP** in any way.

5.5.1. Group Options

A resource group inherits the following options from the resources that it contains: **priority**, **target-role**, **is-managed**. For information on resource options, refer to Table 4.3, “Resource Meta Options”.

5.5.2. Group Constraints

Although it is possible to reference the individual resources that make of a group when you configure constraints, it is preferable to add constraints to the group as a whole.

5.5.3. 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.

Chapter 6. Fencing: Configuring STONITH

STONITH is an acronym for Shoot-The-Other-Node-In-The-Head and it protects your data from being corrupted by rogue nodes or concurrent access.

Just because a node is unresponsive, this does not mean it is not 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.

6.1. Available STONITH (Fencing) Agents

Use the following command to view of list of all available STONITH agents. You you specify a filter, then this command displays only the STONITH agents that match the filter.

```
pcs stonith list [filter]
```

6.2. General Properties of Fencing Devices

Table 6.1, “General Properties of Fencing Devices” describes the general properties you can set for fencing devices. Refer to Section 6.3, “Displaying Device-Specific Fencing Options” for information on fencing properties you can set for specific fencing devices.

Note

For information on more advanced fencing configuration properties, refer to Section 6.8, “Additional Fencing Configuration Options”

Table 6.1. General Properties of Fencing Devices

Field	Type	Default	Description
stonith-timeout	time	60s	How long to wait for the STONITH action to complete per a stonith device. Overrides the **stonith-timeout** cluster property.
priority	integer	0	The priority of the stonith resource. Devices are tried in order of highest priority to lowest.
pcmk_host_map	string		A mapping of host names to ports numbers for devices that do not support host names. For example: **node1:1;node2:2,3** tells the cluster to use port 1 for node1 and ports 2 and 3 for node2.
pcmk_host_list	string		A list of machines controlled by this device (Optional unless **pcmk_host_check=static-list**).
pcmk_host_check	string	**dynamic-list**	How to determine which machines are controlled by the device. Allowed values: **dynamic-list** (query the device), **static-list** (check the **pcmk_host_list** attribute), none (assume every device can fence every machine) .

6.3. Displaying Device-Specific Fencing Options

Use the following command to view the options for the specified STONITH agent.

```
pcs stonith describe stonith_agent
```

For example, the following command displays the options for the fence agent for APC over telent/SSH.

```
# pcs stonith describe fence_apc
Stonith options for: fence_apc
  ipaddr (required): IP Address or Hostname
  login (required): Login Name
  passwd: Login password or passphrase
  passwd_script: Script to retrieve password
  cmd_prompt: Force command prompt
  secure: SSH connection
  port (required): Physical plug number or name of virtual machine
  identity_file: Identity file for ssh
  switch: Physical switch number on device
  inet4_only: Forces agent to use IPv4 addresses only
  inet6_only: Forces agent to use IPv6 addresses only
  ipport: TCP port to use for connection with device
  action (required): Fencing Action
  verbose: Verbose mode
  debug: Write debug information to given file
  version: Display version information and exit
  help: Display help and exit
  separator: Separator for CSV created by operation list
  power_timeout: Test X seconds for status change after ON/OFF
  shell_timeout: Wait X seconds for cmd prompt after issuing command
  login_timeout: Wait X seconds for cmd prompt after login
  power_wait: Wait X seconds after issuing ON/OFF
  delay: Wait X seconds before fencing is started
  retry_on: Count of attempts to retry power on
```

6.4. Creating a Fencing Device

The following command creates a fencing device.

```
pcs stonith create stonith_id stonith_device_type [stonith_device_options]
```

```
# pcs stonith create MyStonith fence_virt pcmk_host_list=f1 op monitor
interval=30s
```

If you use a single fence device for several nodes, using a different port of each node, you do not need to create a device separately for each node. Instead you can use the **pcmk_host_map** option to define which port goes to which node. For example, the following command creates a single fencing device called **myapc-west-13** that uses an APC powerswitch called **west-apc** and uses port 15 for node **west-13**.

```
# pcs stonith create myapc-west-13 fence_apc pcmk_host_list="west-13"
ipaddr="west-apc" login="apc" passwd="apc" port="15"
```

The following example, however, uses the APC powerswitch named **west-apc** fo fence nodes **west-13** using port 15, **west-14** using port 17, **west-15** using port 18, and **west-16** using port 19.

```
# pcs stonith create myapc fence_apc pcmk_host_list="west-13,west-14,west-
15,west-16" pcmk_host_map="west-13:15;west-14:17;west-15:18;west-16:19"
ipaddr="west-apc" login="apc" passwd="apc"
```

6.5. Displaying Fencing Devices

The following command shows all currenly configured fencing devices. If a ***stonith_id*** is specified, the command shows the options for that configured stonith device only. If the **--full** option is specified, all configured stonith options are displayed.

```
pcs stonith show [stonith_id] [--full]
```

6.6. Modifying and Deleting Fencing Devices

Use the following command to modify or add options to a currently configured fencing device.

```
pcs stonith update stonith_id [stonith_device_options]
```

Use the following command to remove a fencing device from the current configuration.

```
pcs stonith delete stonith_id
```

6.7. Managing Nodes with Fence Devices

You can fence a node manually with the following command. If you specify **--off** this will use the **off** API call to stonith which will turn the node off instead of rebooting it.

```
pcs stonith fence node [--off]
```

You can confirm whether a specified node is current down with the following command.

Warning

If the node you specify is not actually down, data corruption/cluster failure can occur.

```
pcs stonith confirm node
```

6.8. Additional Fencing Configuration Options

Table 6.2, “Advanced Properties of Fencing Devices”. summarizes additional properties you can set for fencing devices. Note that these properties are for advanced use only.

Table 6.2. Advanced Properties of Fencing Devices

Field	Type	Default	Description
pcmk_host_argument	string	port	An alternate parameter to supply instead of port. Some devices do not support the standard port parameter or may provide additional ones. Use this to specify an alternate, device-specific, parameter that should indicate the machine to be fenced. A value of none can be used to tell the cluster not to supply any additional parameters.
pcmk_reboot_action	string	**reboot**	An alternate command to run instead of **reboot**. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific, command that implements the reboot action.
pcmk_reboot_timeout	time	60s	Specify an alternate timeout to use for reboot actions instead of **stonith-timeout**. Some devices need much more/less time to complete than normal. Use this to specify an alternate, device-specific, timeout for reboot actions.
pcmk_reboot_retries	integer	2	The maximum number of times to retry the **reboot** command within the timeout period. Some devices do not support multiple connections. 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 reboot actions before giving up.
pcmk_off_action	string	**off**	An alternate command to run instead of **off**. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific, command that implements the off action.
pcmk_off_timeout	time	60s	Specify an alternate timeout to use for off actions instead of **stonith-timeout**. Some devices need much more or much less time to complete than normal. Use this to specify an alternate, device-specific, timeout for off actions.
pcmk_off_retries	integer	2	The maximum number of times to retry the off command within the timeout period. Some devices do not support multiple connections. 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 off actions before giving up.
pcmk_list_action	string	list	An alternate command to run instead of **list**. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific, command that implements the list action.
pcmk_list_timeout	time	60s	Specify an alternate timeout to use for list actions instead of **stonith-timeout**. Some devices need much more or much less time to complete than normal. Use this to specify an alternate, device-specific, timeout for list actions.
pcmk_list_retries	integer	2	The maximum number of times to retry the **list** command within the timeout period. Some devices do not support multiple connections. 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 list actions before giving up.
pcmk_monitor_action	string	**monitor**	An alternate command to run instead of **monitor**. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific, command that implements the monitor action.
pcmk_monitor_timeout	time	60s	Specify an alternate timeout to use for monitor actions instead of **stonith-timeout**. Some devices need much more or much less time to complete than normal. Use this to specify an alternate, device-specific, timeout for monitor actions.
pcmk_monitor_retries	integer	2	The maximum number of times to retry the **monitor** command within the timeout period. Some devices do not support multiple connections. 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 monitor actions before giving up.
pcmk_status_action	string	status	An alternate command to run instead of **status**. Some devices do not support the

			standard commands or may provide additional ones. Use this to specify an alternate, device-specific, command that implements the status action.
pcmk_status_timeout	time	60s	Specify an alternate timeout to use for status actions instead of **stonith-timeout**. Some devices need much more or much less time to complete than normal. Use this to specify an alternate, device-specific, timeout for status actions.
pcmk_status_retries	integer	2	The maximum number of times to retry the status command within the timeout period. Some devices do not support multiple connections. 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 status actions before giving up.

6.9. Configuring Fencing Levels

Pacemaker supports fencing nodes with multiple devices through a feature called fencing topologies. To implement topologies, create the individual devices as you normally would and then define one or more fencing levels in the fencing-topology section in the configuration.

- Each level is attempted in ascending numeric order, starting at 1.
- 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 all devices are successfully fenced, then that level has succeeded and no other levels are tried.
- The operation is finished when a level has passed (success), or all levels have been attempted (failed).

Use the following command to add a fencing level to a node. The devices are given as a comma-separated list of stonith ids, which are attempted for the node at that level.

```
pcs stonith level add level node devices
```

The following command lists all of the fencing levels that are currently configured.

```
pcs stonith level
```

The following command removes the fence level for the specified node and devices. If no nodes or devices are specified then the fence level is removed.

```
pcs stonith level remove level [node_id] [stonith_id] ... [stonith_id]
```

The following command clears the fence levels on the specified node or stonith id. If you do not specify a node or stonith id, all fence levels are cleared.

```
pcs stonith level clear [node|stonith_id(s)]
```

If you specify more than one stonith id, they must be separated by a comma and no spaces, as in the following example.

```
# pcs stonith level clear dev_a,dev_b
```

The following command verifies that all fence devices and nodes specified in fence levels exist.

```
pcs stonith level verify
```

Chapter 7. Pacemaker Rules

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.

Table 7.1. Properties of a Rule

Field	Description
role	Limits the rule to apply only when the resource is in that role. Allowed values: **Started**, **Slave**, and **Master**. NOTE: A rule with **role=Master** can not determine the initial location of a clone instance. It will only affect which of the active instances will be promoted.
score	The score to apply if the rule evaluates to **true**. Limited to use in rules that are part of location constraints.
score-attribute	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.
boolean-op	How to combine the result of multiple expression objects. Allowed values: **and** and **or**.

7.1. Node Attribute Expressions

Node attribute expressions are used to control a resource based on the attributes defined by a node or nodes.

Table 7.2. Properties of an Expression

Field	Description
value	User supplied value for comparison.
attribute	The node attribute to test.
type	Determines how the value(s) should be tested. Allowed values: **string**, **integer**, **version**.
operation	The comparison to perform. Allowed values: * **lt** - True if the node attribute's value is less than **value** * **gt** - True if the node attribute's value is greater than **value** * **lte** - True if the node attribute's value is less than or equal to **value** * **gte** - True if the node attribute's value is greater than or equal to **value** * **eq** - True if the node attribute's value is equal to **value** * **ne** - True if the node attribute's value is not equal to **value** * **defined** - True if the node has the named attribute * **not_defined** - True if the node does not have the named attribute

7.2. Time/Date Based Expressions

Date expressions are used to control a resource or cluster option based on the current date/time. They can contain an optional date specification.

Table 7.3. Properties of a Date Expression

Field	Description
start	A date/time conforming to the ISO8601 specification.
end	A date/time conforming to the ISO8601 specification.
operation	Compares the current date/time with the start and/or end date, depending on the context. Allowed values: * **gt** - True if the current date/time is after **start** * **lt** - True if the current date/time is before **end** * **in-range** - True if the current date/time is after **start** and before **end** * **date-spec** - performs a cron-like comparison to the current date/time

7.3. Date Specifications

Date specifications 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). However, you cannot specify **weekdays="1,2"** or **weekdays="1-2,5-6"** since they contain multiple ranges.

Table 7.4. Properties of a Date Specification

Field	Description
id	A unique name for the date
hours	Allowed values: 0-23
monthdays	Allowed values: 0-31 (depending on month and year)
weekdays	Allowed values: 1-7 (1=Monday, 7=Sunday)
yeardays	Allowed values: 1-366 (depending on the year)
months	Allowed values: 1-12
weeks	Allowed values: 1-53 (depending on weekyear)
years	Year according the Gregorian calendar
weekyears	May differ from Gregorian years; for example, **2005-001 Ordinal** is also **2005-01-01 Gregorian** is also **2004-W53-6 Weekly**
moon	Allowed values: 0-7 (0 is new, 4 is full moon).

7.4. 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 (for example, you can have a duration of 19 months). As with **date_spec** objects, any field not supplied is ignored.

7.5. Configuring Rules with pcs

To configure a rule, use the following command. If **score** is omitted, it defaults to INFINITY. If **id** is omitted, one is generated from the ***constraint_id***. The ***rule_type*** should be **expression** or **date_expression**.

```
pcs constraint rule add constraint_id [rule_type] [score=score] [id=rule_id]
expression|date_expression|date_spec options
```

To remove a rule, use the following command to remove a rule. If the rule that you are removing is the last rule in its constraint, the constraint will be removed.

```
pcs constraint rule remove rule_id
```

7.6. Using Rules to Determine Resource Location

You can use a rule to determine a resource's location with the following command.

```
pcs constraint location resource_id rule [rule_id] [role=master|slave]
[score=score] expression
```

The ***expression*** can be one of the following:

- **defined|not_defined *attribute***
- ***attribute* lt|gt|lte|gte|eq|ne *value***
- **date [start=*start*] [end=*end*] operation=gt|lt|in-range**
- **date-spec *date_spec_options***

Chapter 8. Managing Cluster Resources

This chapter describes various commands you can use to manage cluster resources.

8.1. Manually Moving Resources Around the Cluster

You can override the cluster and force resources to move from their current location. There are two occasions when you would want to do this:

- When a node is under maintenance, and you need to move all resources running on that node to a different node
- When a single resource needs to be moved

To move all resources running on a node to a different node, you put the node in standby mode. For information on putting a cluster node in standby node, refer to Section 3.3.4, “Standby Mode”.

To move a resource off the node on which it is currently running, use the following command, specifying the ***resource_id*** of the node as defined.

```
pcs resource move resource_id
```

If you want so specify on which node to run the resource that you are moving, use the following command to specify the ***destination_node***.

```
pcs resource move resource_id destination_node
```

Use the following command to return the resource back to the node on which it was originally running, allowing the cluster to resume normal operation. This removes the constraints that the **move resource_id** command defined.

```
pcs resource clear resource_id [node]
```

Note that when you execute the **pcs resource move** command, this adds constraints to the resource to prevent it from running on the indicated node. When you execute the **pcs resource clear** command, this removes the constraints. This does not necessarily move the resources back to the indicated node; where the resources can run at that point depends on how you have configured your resources initially. For information on resource constraints, refer to Chapter 5, *Resource Constraints*.

8.2. Moving Resources Due to Failure

When you create a resource, you can configure the resource so that it will move to a new node after a defined number of failures by setting the **migration-threshold** option for that resource. Once the threshold has been reached, this node will no longer be allowed to run the failed resource until:

- The administrator manually resets the resource's failcount using the **pcs resource failcount** command.
- The resource's **failure-timeout** value is reached.

There is no threshold defined by default.

Note

Setting a **migration-threshold** for a resource is not the same as configuring a resource for migration, in which the resource moves to another location without loss of state.

To determine the resource's current failure status and limits, use the **pcs resource failcount**.

There are two exceptions to the migration threshold concept; they occur when a resource either fails to start or fails to stop. Start failures cause the failcount 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.

8.3. Enabling, Disabling, and Banning Cluster Resources

There are a variety of commands you can use to control the behavior of cluster resources.

You can manually stop a running resource and prevent the cluster from starting it again with the following command. Depending on the rest of the configuration (constraints, options, failures, etc), the resource may remain started. If you specify the **--wait** option, **pcs** will wait up to 30 seconds (or 'n' seconds, as specified) for the resource to stop and then return 0 if the resource is stopped or 1 if the resource has not stopped.

```
pcs resource disable resource_id [--wait[=n]]
```

You can use the following command to allow the cluster to start a resource. Depending on the rest of the configuration, the resource may remain stopped. If you specify the **--wait** option, **pcs** will wait up to 30 seconds (or 'n' seconds, as specified) for the resource to start and then return 0 if the resource is started or 1 if the resource has not started.

```
pcs resource enable resource_id [--wait[=n]]
```

Use the following command to prevent a resource from running on a specified node, or on the current node if no node is specified.

```
pcs resource ban resource_id [node]
```

Note that when you execute the **pcs resource ban** command, this adds constraints to the resource to prevent it from running on the indicated node. You can execute the **pcs resource clear** command to removes the constraints. This does not necessarily move the resources back to the indicated node; where the resources can run at that point depends on how you have configured your resources initially. For information on resource constraints, refer to Chapter 5, *Resource Constraints*.

```
pcs resource clear resource_id [node]
```

You can use the **debug-start** parameter of the **pcs resource** command to force a specified resource to start on the current node, ignoring the cluster recommendations and printing the output from starting the resource. This is mainly used for debugging resources; starting resources on a cluster is

(almost) always done by Pacemaker and not directly with a **pcs** command. If your resource is not starting, it is usually due to either a misconfiguration of the resource (which you debug in the system log), constraints that the resource from starting, or the resource being disabled. You can use this command to test resource configuration, but it should not normally be used to start resources in a cluster.

The format of the **debug-start** command is as follows.

```
pcs resource debug-start resource_id
```

Chapter 9. Advanced Resource types

This chapter describes advanced resource types that Pacemaker supports.

- Resource clones, which allow a resource to be active on multiple nodes, are described in Section 9.1, “Resource Clones”
- Multistate resources, which allow a resource to have multiple operating modes, are described in Section 9.2, “Multi-State Resources: Resources That Have Multiple Modes”.

9.1. Resource Clones

You can clone a resource so that the resource can be active on multiple nodes. For example, you can use cloned resources to configure multiple instances of an IP resource to distribute throughout a cluster for node balancing. You can clone any resource provided the resource agent supports it. A clone consists of one resource or one resource group.

9.1.1. Creating and Removing a Cloned Resource

You can create a resource and a clone of that resource at the same time with the following command.

```
pcs resource create resource_id standard:provider:type|type [resource options] --clone [meta clone_options]
```

The name of the clone will be ***resource_id*-clone**.

You cannot create a resource group and a clone of that resource group in a single command.

Alternately, you can create a clone of a previously-created resource or resource group with the following command.

```
pcs resource clone resource_id | group_name [clone_options]...
```

The name of the clone will be ***resource_id*-clone** or ***group_name*-clone**.

Use the following command to remove a clone of a resource or a resource group. This does not remove the resource or resource group itself.

```
pcs resource unclone resource_id | group_name
```

For information on resource options, refer to Section 4.1, “Resource Creation”.

Table 9.1, “Resource Clone Options” describes the options you can specify for a cloned resource.

Table 9.1. Resource Clone Options

Field	Description
priority, target-role, is-managed	Options inherited from resource that is being cloned.
clone-max	How many copies of the resource to start. Defaults to the number of nodes in the cluster.
clone-node-max	How many copies of the resource can be started on a single node; default **1**.
notify	When stopping or starting a copy of the clone, tell all the other copies beforehand and when the action was successful. Allowed values: **false**, **true**
globally-unique	Does each copy of the clone perform a different function? Allowed values: **false**, **true**.
ordered	Should the copies be started in series (instead of in parallel). Allowed values: **false**, **true**.
interleave	Changes the behavior of ordering constraints (between clones/masters) so that instances can start/stop as soon as their peer instance has (rather than waiting for every instance of the other clone has). Allowed values: **false**, **true**.

9.1.2. Clone Constraints

In most cases, a clone will have a single copy 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 to those for regular resources.

Ordering constraints behave slightly differently for clones. If you configure an ordering constraint so that a clone starts before a resource that is not a clone and there is more than one copy of the clone, the resource will wait until all copies of the clone that need to be started have done so before being started itself. Only if *no* copies can be started will the resource be prevented from being active. Additionally, the clone will wait for the resource to be stopped before stopping the clone.

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

Colocation between clones is also possible. In such cases, the set of allowed locations for the clone is limited to nodes on which the clone is (or will be) active. Allocation is then performed as normally.

9.1.3. 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.

9.2. Multi-State Resources: Resources That Have Multiple Modes

Multi-state resources are a specialization of clone resources. They allow the instances to be in one of two operating modes; these are called **master** and **slave**. The names of the modes do not have

specific meanings, except for the limitation that when an instance is started, it must come up in the **slave** state.

Before configuring a multi-state resource, you should ensure that the resource agent supports master/slave resources.

You can create a resource as a a master/slave clone with the following single command.

```
pcs resource create resource_id standard:provider:type|type [resource options] --master [meta master_options]
```

The name of the master/slave clone will be ***resource_id*-master**.

Alternately, you can create a master/slave clone from a previously-created resource or resource group with the following command. When you use this command, you can specify a name for the master/slave clone. If you do not specify a name, the name of the master/slave clone will be ***resource_id*-master** or ***group_name*-master**.

```
pcs resource master master/slave_name resource_id|group_name [master_options]
```

For information on resource options, refer to Section 4.1, “Resource Creation”.

Table 9.2, “Properties of a Multi-State Resource” describes the options you can specify for a multi-state resource.

Table 9.2. Properties of a Multi-State Resource

Field	Description
id	Your name for the multi-state resource
priority, **target-role**, **is-managed**	See Table 4.3, “Resource Meta Options”.
clone-max, **clone-node-max**, **notify**, **globally-unique**, **ordered**, **interleave**	See Table 9.1, “Resource Clone Options”.
master-max	How many copies of the resource can be promoted to **master** status; default 1.
master-node-max	How many copies of the resource can be promoted to **master** status on a single node; default 1.

9.2.1. Monitoring Multi-State Resources

The normal type of monitor actions are not sufficient to monitor a multi-state resource in the **master** state. To detect failures of the **master** instance, you need to define an additional monitor action with **role="Master"**.

Important

It is crucial that *every* monitor operation has a different interval!
This is because Pacemaker currently differentiates between operations only by resource and interval so if, for example, a master/slave resource has 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.

9.2.2. Multi-state Constraints

In most cases, a multi-state resources will have a single copy 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 than those for regular resources.

For information on resource location constraints, see Section 5.1, “Location Constraints”.

You can create a colocation constraint which specifies whether the resources are master or slave resources. The following command creates a resource colocation constraint.

```
pcs constraint colocation add [master|slave] source_resource with [master|slave]
target_resource [score] [options]
```

For information on resource location constraints, see Section 5.3, “Colocation of Resources”.

When configuring an ordering constraint that includes multi-state resources, one of the actions that you can specify for the resources is **promote**, indicating that the resource be promoted from slave to master. Additionally, you can specify an action of **demote**, indicated that the resource be demoted from master to slave.

The command for configuring an order constraint is as follows.

```
pcs constraint order [action] resource_id then [action] resource_id [options]
```

For information on resource order constraints, see Section 5.2, “Order Constraints”.

9.2.3. Multi-state Stickiness

To achieve a stable allocation pattern, multi-state resources are slightly sticky by default. If no value for **resource-stickiness** is provided, the multi-state resource 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.

Cluster Creation with rgmanager and with Pacemaker

Table A.1, “Comparison of Cluster Configuration with rgmanager and with Pacemaker” provides a comparative summary of how you configure the components of a cluster when using **rgmanager** and when using Pacemaker..

Table A.1. Comparison of Cluster Configuration with rgmanager and with Pacemaker

Configuration Component	rgmanager	Pacemaker
Cluster configuration file	The cluster configuration file on each node is **cluster.conf** file, which can can be edited directly if desired. Otherwise, use the **luci** or **ccs** interface to define the cluster configuration.	The cluster and Pacemaker configuration files are **cluster.conf** and **cib.xml**. Do not edit these files directly; use the **pcs** interface instead.
Network setup	Configure IP addresses and SSH before configuring the cluster.	Configure IP addresses and SSH before configuring the cluster.
Cluster Configuration Tools	**luci**, **ccs** command, manual editing of **cluster.conf** file.	**pcs**
Installation	Install **rgmanager** (which pulls in all dependencies, including **ricci**, **luci**, and the resource and fencing agents). If needed, install **lvm2-cluster** and **gfs2-utils**.	Install **pacemaker**, **cman**, **pcs**, and the resource and fencing agents you require. If needed, install **lvm2-cluster** and **gfs2-utils**.
Starting cluster services	Start and enable cluster services with the following procedure: 1. Start **rgmanager**, **cman**, and, if needed, **clvmd** and **gfs2**. 2. Start **ricci**, and start **luci** if using the **luci** interface. 3. Run **chkconfig on** for the needed services so that they start at each runtime. Alternately, you can run **ccs --start** to start and enable the cluster services.	On all nodes in the cluster, run **pcs cluster start** to start **cman** and **pacemaker**.
Controlling access to configuration tools	For **luci**, the root user or a user with **luci** permissions can access **luci**. All access requires the **ricci** password for the node.	There is no configuration GUI.
Cluster creation	Name the cluster and define which nodes to include in the cluster with **luci** or **ccs**, or directly edit the **cluster.conf** file.	Name the cluster and include nodes with the **pcs cluster setup** command. To add nodes you must run **pcs cluster setup** on all nodes.
Propagating cluster configuration to all nodes	When configuration a cluster with **luci**, propagation is automatic. With **ccs**, use the **--sync** option. You can also use the **cman_tool version -r** command.	Propagation of the cluster and Pacemaker configuration files, **cluster.conf** and **cib.xml**, is automatic on cluster setup or when adding a resource.
Global cluster properties	The following feature are supported with **rgmanager**:	Pacemaker supports the following features for a cluster:

	* You can configure the system so that the system chooses which multicast address to use for IP multicasting in the cluster network. * If IP multicasting is not available, you can use UDP Unicast transport mechanism. * You can configure a cluster to use RRP protocol.	* You can set **no-quorum-policy** for the cluster to specify what the system should do when the cluster does not have quorum. * For additional cluster properties you can set, refer to Table 2.1, "Cluster Properties".
Logging	You can set global and daemon-specific logging configuration.	See the file **/etc/sysconfig/pacemaker** for information on how to configure logging manually.
Validating the cluster	Cluster validation is automatic with **luci** and with **ccs**, using the cluster schema. The cluster is automatically validated on startup.	The cluster is automatically validated on startup, or you can validate the cluster with **pcs cluster verify**.
Quorum in 2-node clusters	With a two-node cluster, you can configure how the system determines quorum: * Configure a quorum disk * Use **ccs** or edit the **cluster.conf** file to set **two_node=1** and **expected_votes=1** to allow a single node to maintain quorum.	**pcs** automatically adds the necessary options for a two-node cluster to **cman**.
Cluster status	On **luci**, the current status of the cluster is visible in the various components of the interface, which can be refreshed. You can use the **--gethost** option of the **ccs** command to see current the configuration file. You can use the **clustat** command to display cluster status.	You can display the current cluster status with the **pcs status**.
Resources	You add resources of defined types and configure resource-specific properties with **luci** or the **ccs** command, or by editing the **cluster.conf** configuration file.	You add resources of defined types and configure resource-specific properties with the **pcs resource create**. For general information on configuring cluster resources with Pacemaker refer to Chapter 4, *Configuring Cluster Resources*.
Resource behavior, grouping, and start/stop order	Define cluster *services* to configure how resources interact.	With Pacemaker you use resource groups as a shorthand method of defining a set of resources that need to be located together and started and stopped sequentially. In addition,

		you define how resources behave and interact in the following ways: * You set some aspects of resource behavior as resource options. * You use location constraints to determine which nodes a resource can run on. * You use order constraints to determine the order in which resources run. * You use colocation constraints to determine that the location of one resource depends on the location of another resource. For more complete information on these topics, refer to Chapter 4, *Configuring Cluster Resources*.
Resource administration: Moving, starting, stopping resources	With **luci**, you can manage clusters, individual cluster nodes, and cluster services. With the **ccs** command, you can manage cluster. You can use the **clusvadm** to manage cluster services.	You can temporarily disable a node so that it can not host resources with the **pcs cluster standby** command, which causes the resources to migrate. You can stop a resource with the **pcs resource disable** command.
Removing a cluster configuration completely	With **luci**, you can select all nodes in a cluster for deletion to delete a cluster entirely. You can also remove the **cluster.conf** from each node in the cluster.	You can remove a cluster configuration from a node with the **pcs cluster destroy** command.
Resources active on multiple nodes, resources active on multiple nodes in multiple modes	No equivalent	With Pacemaker, you can clone resources so that they can run in multiple nodes, and you can define cloned resources as master and slave resources so that they can run in multiple modes. For information on cloned resources and master/slave resources, refer to Chapter 9, *Advanced Resource types*.
Fencing -- single fence device per node	Create fencing devices globally or locally and add them to nodes. You can define **post-fail delay** and **post-join delay** values for the cluster as a whole.	Create a fencing device for each node with the **pcs stonith create** command. For devices that can fence multiple nodes, you need to define them only once rather than separately for each node. You can also define **pcmk_host_map** to configure fencing devices for all nodes with a single command; for information on **pcmk_host_map**

		refer to Table 6.1, “General Properties of Fencing Devices”. You can define the **`stonith-timeout`** value for the cluster as a whole.
Multiple (backup) fencing devices per node	Define backup devices with **luci** or the **`ccs`** command, or by editing the **`cluster.conf`** file directly.	Configure fencing levels.

Configuration Example Using pcs Commands

This appendix provides a step-by-step procedure for configuring a two-node cluster, using the **pcs** command.

This appendix provides a procedure for configuring a cluster with the following characteristics:

- 2-node cluster.
- 4 defined resources in the cluster: an LVM volume, an ext4 file system on the volume, an IP address for a web server, Apache web server.
- The cluster resources must becolocated on one node.
- The resources must start in this order: LVM volume, ext4 file system, IP address, Apache web server. The resources will stop in the reverse order to which they are started.

B.1. Initial System Setup

This section describes the initial setup of the system that you will use to create the cluster.

B.1.1. Installing the Cluster Software

Use the following procedure to install the cluster software.

1. Ensure that **pacemaker**, **cman**, and **pcs** are installed.

```
yum install -y pacemaker cman
yum install -y pcs
```

2. After installation, to prevent **corosync** from starting without **cman**, execute the following command on all nodes in the cluster.

```
# chkconfig corosync off
```

3. If you want to ensure that **cman** should complete starting up even without quorum and there are more than two nodes in the cluster, execute the following command.

```
# sed -i.sed "s/.*CMAN_QUORUM_TIMEOUT=.*/CMAN_QUORUM_TIMEOUT=0/g"
/etc/sysconfig/cman
```

B.1.2. Configuring an LVM Volume with an ext4 File System

The web server that this example creates uses a web page on an **ext4** file system mounted on an LVM logical volume. In this example, the partition **/dev/sdb1** will be used to store the LVM physical volume from which the LVM logical volume will be created. This partition is storage that is shared among the nodes of the sample cluster.

The web server that this example creates uses a web page on an **ext4** file system mounted on an LVM logical volume. In this example, the partition **/dev/sdb1** will be used to store the LVM physical volume from which the LVM logical volume will be created. This partition is storage that is shared among the nodes of the sample cluster.

You must set up the volume group in a way that will ensure that only the cluster is capable of activating the volume group, and that the volume group will not be activated outside of the cluster on startup. If it is possible for the volume group to activate outside of the cluster on a node, there is a risk of corrupting the volume group's metadata. To ensure that this does not occur, before creating the LVM volume for the

cluster you must modify the **volume_list** entry in the **/etc/lvm/lvm.conf** configuration file to allow the node to activate volume groups that are not used by the cluster with the following procedure:

1. Determine which volume groups are currently configured on your share storage with the following command. This will output a list of the currently-configured volume groups.

```
# vgs --noheadings -o vg_name
```

2. On both nodes that you will use for your cluster, add the volume groups as entries to **volume_list** in the **lvm.conf** configuration file.

You will also need to add the **exclusive=true** parameter to the **pcs resource create** command that creates the LVM volume, as described in Section B.4.1, “Adding Resources”.

The following procedure creates a logical volume with an **ext4** file system

1. Create an LVM physical volume on partition **/dev/sdb1**.

```
# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created
```

2. Create the volume group **my_vg** that consists of the physical volume **/dev/sdb1**.

```
# vgcreate my_vg /dev/sdb1
  Volume group "my_vg" successfully created
```

3. Create a logical volume using the volume group **my_vg**.

```
# lvcreate -L450 -n my_lv my_vg
  Rounding up size to full physical extent 452.00 MiB
  Logical volume "my_lv" created
```

   You can use the **lvs** to display the logical volume.

```
# lvs
  LV      VG      Attr      LSize   Pool Origin Data%  Move Log Copy%
Convert
  my_lv   my_vg   -wi-a---- 452.00m
  ...
```

4. Create an **ext4** file system on the logical volume **my_lv**.

```
# mkfs.ext4 /dev/my_vg/my_lv
mke2fs 1.42.7 (21-Jan-2013)
Filesystem label=
OS type: Linux
...
```

B.1.3. Web Server Configuration

To create a web server, you need to be sure that Apache is installed on both hosts. You also need the **wget** tool for the cluster to be able to check the status of the apache web server.

On each node, execute the following command.

```
# yum install -y httpd wget
Loaded plugins: langpacks, product-id, subscription-manager
Server                                                   | 3.8 kB
00:00:00
Server-optional                                          | 3.8 kB
00:00:00
addons-HighAvailability                                  | 3.8 kB
00:00:00
addons-LoadBalancer                                      | 3.8 kB
00:00:00
...
```

In order for the Apache web server to use the **index.html** file , ensure that the following text is present in the **/etc/httpd/conf/httpd.conf** file on both nodes in the system, ensure that it has not been commented out.

```
<Location /server-status>
  SetHandler server-status
  Order deny,allow
  Deny from all
  Allow from 127.0.0.1
</Location>
```

For this example, we will need to create a web page for Apache to serve up. To do this, mount the file system you created in Section B.1.2, “Configuring an LVM Volume with an ext4 File System”, create the file **index.html** on that file system, then unmount the file system.

```
# mount /dev/my_vg/my_lv /mnt/
# cat <<-END >/mnt/index.html
> <html>
> <body>Hello>
> </html>
>END
# umount /dev/my_vg/my_lv
```

B.2. Creating the Initial Cluster

This section describes the **pcs** and provides the procedure for creating the initial cluster, on which you will configure the cluster resources.

B.2.1. The pcs Command

The **pcs** command-line shell breaks up cluster configuration and management into categores. Executing the **pcs** command with no options displays the categories.

```
# pcs

Usage: pcs [-f file] [-h] [commands]...
Control and configure pacemaker and corosync.

Options:
    -h, --help  Display usage and exit
    -f file     Perform actions on file instead of active CIB
    --debug     Print all network traffic and external commands run
    --version   Print pcs version information

Commands:
    resource    Manage cluster resources
    cluster     Configure cluster options and nodes
    stonith     Configure fence devices
    property    Set pacemaker properties
    constraint  Set resource constraints
    status      View cluster status
    config      Print full cluster configuration
```

For each of the categories of cluster management, you can display the functionality of that category by issuing the command **pcs *category* help**. For example, the following command displays the available **status** options.

```
# pcs status help

Usage: pcs status [commands]...
View current cluster and resource status
Commands:
    [status]
        View all information about the cluster and resources

    resources
        View current status of cluster resources

    groups
        View currently configured groups and their resources

    cluster
        View current cluster status

    corosync
        View current corosync status

    nodes [corosync|both|config]
        View current status of nodes from pacemaker. If 'corosync' is
        specified, print nodes currently configured in corosync, if 'both'
        is specified, print nodes from both corosync & pacemaker.  If 'config'
        is specified, print nodes from corosync & pacemaker configuration.

    actions
        View failed actions

    pcsd <node> ...
        Show the current status of pcsd on the specified nodes

    xml
        View xml version of status (output from crm_mon -r -1 -X)
```

B.2.2. Creating and Starting the Cluster

Execute the following command from each node in the cluster to create the two-node cluster **mycluster** that consists of nodes **z1.example.com** and **z2.example.com**. The node names must match the hostnames associated with the IP address of the network interface that is used for cluster communication on each node.

```
[root@z1 ~]# pcs cluster setup --name my_cluster z1.example.com z2.example.com
```

To start the cluster, execute the following command from each node in the cluster.

```
[root@z1 ~]# pcs cluster start
```

You can display the current status of the cluster with the **pcs cluster status** command.

```
[root@z1 ~]# pcs cluster status
Cluster Status:
 Last updated: Thu Jul 25 13:01:26 2013
 Last change: Thu Jul 25 13:04:45 2013 via crmd on z2.example.com
 Stack: corosync
 Current DC: z2.example.com (2) - partition with quorum
 Version: 1.1.10-5.el7-9abe687
 2 Nodes configured
 0 Resources configured
```

B.3. Configuring Fencing

When configuring a cluster, it is necessary to configure a STONITH fencing device for each node in the cluster. STONITH is an acronym for Shoot-The-Other-Node-In-The-Head and it protects your data from being corrupted by rogue nodes or concurrent access. Just because a node is unresponsive does not mean it is not accessing your data. The only way to be sure that your data is safe is to use STONITH to be certain that the node is 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.

Note

When configuring a fencing device, you should ensure that your fencing device does not share power with the node that it controls.

The **pcs stonith list** command displays the available STONITH agents. The **pcs stonith describe *stonith_agent*** command shows the options for the specified STONITH agent.

You can use the following command to display the parameters you can set for the **fence_apc** agent, which is what we use in this example.

```
[root@z ~]# pcs stonith describe fence_apc
Stonith options for: fence_apc
  ipaddr (required): IP Address or Hostname
  login (required): Login Name
  passwd: Login password or passphrase
  cmd_prompt: Force command prompt
  secure: SSH connection
  port (required): Physical plug number, name of virtual machine or UUID
  switch: Physical switch number on device
  ipport: TCP/UDP port to use for connection with device
  inet4_only: Forces agent to use IPv4 addresses only
  inet6_only: Forces agent to use IPv6 addresses only
  passwd_script: Script to retrieve password
  identity_file: Identity file for ssh
  ssh_options: SSH options to use
  action (required): Fencing Action
  verbose: Verbose mode
  debug: Write debug information to given file
  version: Display version information and exit
  help: Display help and exit
  separator: Separator for CSV created by operation list
  power_timeout: Test X seconds for status change after ON/OFF
  shell_timeout: Wait X seconds for cmd prompt after issuing command
  login_timeout: Wait X seconds for cmd prompt after login
  power_wait: Wait X seconds after issuing ON/OFF
  delay: Wait X seconds before fencing is started
  retry_on: Count of attempts to retry power on
```

This example uses the APC power switch with an IP address of **zapc.example.com** to fence the nodes, and it uses the **fence_apc** STONITH driver.

The following command configures a STONITH resource named **myapc-z1** that fences the node **z1.example.com** using port 1 of the fencing device. The login value and password for the APC device are both **apc**. By default, this device will use a monitor interval of 60s for the node.

```
[root@z1 ~]# pcs stonith create myapc-z1 fence_apc
pcmk_host_list="z1.example.com" ipaddr="zapc.example.com" login="apc"
passwd="apc" port="1"
```

Similarly, the following command configures a STONITH resource named **myapc-z2** that fences the node **z2.example.com** using port 2 of the fencing device.

```
[root@z1 ~]# pcs stonith create myapc-z2 fence_apc
pcmk_host_list="z2.example.com" ipaddr="zapc.example.com" login="apc"
passwd="apc" port="2"
```

You can use the following command to display the parameters of an existing STONITH device.

```
[root@z1 ~]# pcs stonith show myapc-z1
 Resource: myapc-z1 (class=stonith type=fence_apc)
  Attributes: pcmk_host_list=z1.example.com ipaddr=zapc.example.com login=apc
passwd=apc port=1
  Operations: monitor interval=60s (myapc-z1-monitor-interval-60s)
```

B.4. Creating Resources and Resource Groups

This section desribes the procedures for creating resources and resource groups.

B.4.1. Adding Resources

For this example, you will create four resources. These resources must all run on the same node, and must start in the following order.

1. An **LVM** resource named **my_lvm**, using the LVM volume group you created in Section B.1.2, "Configuring an LVM Volume with an ext4 File System".
2. A **Filesystem** resource named **my_fs**, using the filesystem device **/dev/my_vg/my_lv** you created in Section B.1.2, "Configuring an LVM Volume with an ext4 File System".
3. An **IPaddr2** resource that the web server uses. IP address must not be one already associated with a physical node
4. An **apache** resource named **Website**, using the **index.html** file and the Apache configuration you defined in Section B.1.3, "Web Server Configuration".

The following commands create the four resources we are defining for this configuration.

```
[root@z1 ~]# pcs resource create my_lvm LVM volgrpname=my_vg exclusive=true

[root@z1 ~]# pcs resource create my_fs Filesystem device="/dev/my_vg/my_lv"
directory="/var/www/html" fstype="ext4" options="ro"

[root@z1 ~]# pcs resource create Website apache
configfile="/etc/httpd/conf/httpd.conf" statusurl="http://127.0.0.1/server-
status"

[root@z1 ~]# pcs resource create ClusterIP IPaddr2 ip=10.15.89.129
cidr_netmask=24
```

You can view the status of the cluster. Note that when you create a resource, the resource is started automatically. You can manually disable and enable an individual resource with the **pcs resource disable** and **pcs resource enable** commands.

```
[root@z1 ~]# pcs status
Cluster name: my_cluster
Last updated: Thu Jul 25 14:19:42 2013
Last change: Thu Jul 25 14:22:58 2013 via cibadmin on z1.example.com
Stack: corosync
Current DC: z2.example.com (2) - partition with quorum
Version: 1.1.10-5.el7-9abe687
2 Nodes configured
6 Resources configured


Online: [ z1.example.com z2.example.com ]

Full list of resources:

 myapc-z1       (stonith:fence_apc):    Started z1.example.com
 myapc-z2       (stonith:fence_apc):    Started z2.example.com
 my_lvm (ocf::heartbeat:LVM):   Started z1.example.com
 my_fs  (ocf::heartbeat:Filesystem):    Started z1.example.com
 ClusterIP      (ocf::heartbeat:IPaddr2):       Started z2.example.com
 Website        (ocf::heartbeat:apache):        Started z2.example.com
```

You can display the available parameters and descriptions for a resource with the **pcs resource describe**. For example, the following command displays the resource options for an **apache** resources.

```
# pcs resource describe apache
Resource options for: apache
  configfile: The full pathname of the Apache configuration file. This file is
              parsed to provide defaults for various other resource agent
              parameters.
  httpd: The full pathname of the httpd binary (optional).
  port: A port number that we can probe for status information using the
        statusurl. This will default to the port number found in the
        configuration file, or 80, if none can be found in the configuration
        file.
  statusurl: The URL to monitor (the apache server status page by default). If
             left unspecified, it will be inferred from the apache configuration
             file. If you set this, make sure that it succeeds *only* from the
             localhost (127.0.0.1). Otherwise, it may happen that the cluster
             complains about the resource being active on multiple nodes.
  testregex: Regular expression to match in the output of statusurl. Case
             insensitive.
  client: Client to use to query to Apache. If not specified, the RA will try to
          find one on the system. Currently, wget and curl are supported. For
          example, you can set this parameter to "curl" if you prefer that to
          wget.
  testurl: URL to test. If it does not start with "http", then it's considered
           to be relative to the Listen address.
  testregex10: Regular expression to match in the output of testurl. Case
               insensitive.
  testconffile: A file which contains test configuration. Could be useful if you
                have to check more than one web application or in case sensitive
                info should be passed as arguments (passwords). Furthermore,
                using a config file is the only way to specify certain
                parameters. Please see README.webapps for examples and file
                description.
  testname: Name of the test within the test configuration file.
  options: Extra options to apply when starting apache. See man httpd(8).
  envfiles: Files (one or more) which contain extra environment variables. If
            you want to prevent script from reading the default file, set this
            parameter to empty string.
  use_ipv6: We will try to detect if the URL (for monitor) is IPv6, but if that
            doesn't work set this to true to enforce IPv6.
```

B.4.2. Resource Groups: Resource Placement and Start Order

To ensure that the resources run on the same node and run in a specified order, create a resource group. The followiing command creates a resource group named **apachegroup** consisting of the four defined resources. These resources will start in the specified order. The resources will stop in the reverse order in which they are specified here.

```
[root@z1 ~]# pcs resource group add apachegroup my_lvm my_fs ClusterIP Website
```

After executing this command, you can check the status of the cluster. Note that all four resources are running on the same node.

```
[root@z1 ~]# pcs status
Cluster name: my_cluster
Last updated: Wed Jul 31 16:38:51 2013
Last change: Wed Jul 31 16:42:14 2013 via crm_attribute on z1.example.com
Stack: corosync
Current DC: z2.example.com (2) - partition with quorum
Version: 1.1.10-5.el7-9abe687
2 Nodes configured
6 Resources configured


Online: [ z1.example.com z2.example.com ]

Full list of resources:

 myapc-z1 (stonith:fence_apc): Started z1.example.com
 myapc-z2 (stonith:fence_apc): Started z2.example.com
 Resource Group: apachegroup
     my_lvm (ocf::heartbeat:LVM): Started z1.example.com
     my_fs (ocf::heartbeat:Filesystem): Started z1.example.com
     ClusterIP (ocf::heartbeat:IPaddr2): Started z1.example.com
     Website (ocf::heartbeat:apache): Started z1.example.com
```

Once the cluster is up and running, you can point a browser to the IP address you defined as the **IPaddr2** resource to view the sample display, consisting of the simple word "Hello".

```
Hello
```

B.5. Testing the Cluster

In the cluster status display shown in Section B.4.2, “Resource Groups: Resource Placement and Start Order”, all of the resources are running on node **z1.example.com**. You can test whether the resource group fails over to node **z2.example.com** by putting the first node in **standby** mode, after which the node will no longer be able to host resources.

```
root@z1 ~]# pcs cluster standby z1.example.com
```

After putting node **z1** in standby mode, check the cluster status. Note that the resources should now all be running on **z2**.

```
[root@z1 ~]# pcs status
Cluster name: my_cluster
Last updated: Wed Jul 31 17:16:17 2013
Last change: Wed Jul 31 17:18:34 2013 via crm_attribute on z1.example.com
Stack: corosync
Current DC: z2.example.com (2) - partition with quorum
Version: 1.1.10-5.el7-9abe687
2 Nodes configured
6 Resources configured


Node z1.example.com (1): standby
Online: [ z2.example.com ]

Full list of resources:

 myapc-z1 (stonith:fence_apc): Started z2.example.com
 myapc-z2 (stonith:fence_apc): Started z2.example.com
 Resource Group: apachegroup
     my_lvm (ocf::heartbeat:LVM): Started z2.example.com
     my_fs (ocf::heartbeat:Filesystem): Started z2.example.com
     ClusterIP (ocf::heartbeat:IPaddr2): Started z2.example.com
     Website (ocf::heartbeat:apache): Started z2.example.com
```

The web site at the defined IP address should still display, without interruption.

To remove **z1** from **standby** mode, run the following command.

```
root@z1 ~]# pcs cluster unstandby z1.example.com
```

Note

Removing a node from **standby** mode does not in itself cause the resources to fail back over to that node. For information on controlling which node resources can run on, see the chapter on configuring cluster resources in *Red Hat High Availability Add-On Reference*.

B.6. Example Configuration Command Summary

After you have set up the initial LVM volume and Apache web server and set a password for user **hacluster** on both nodes, the commands this example uses to configure the cluster configuration are as follows.

On each node in the cluster, execute the following commands.

```
# pcs cluster setup --name my_cluster z1.example.com z2.example.com
# pcs cluster setup --name my_cluster z1.example.com z2.example.com
```

From one node in the cluster, execute the following commands.

```
[root@z1 ~]# pcs stonith create myapc-z1 fence_apc
pcmk_host_list="z1.example.com" ipaddr="zapc.example.com" login="apc"
passwd="apc" port="1"

[root@z1 ~]# pcs stonith create myapc-z2 fence_apc
pcmk_host_list="z2.example.com" ipaddr="zapc.example.com" login="apc"
passwd="apc" port="2"

[root@z1 ~]# pcs resource create my_lvm LVM volgrpname=my_vg exclusive=true

[root@z1 ~]# pcs resource create my_fs Filesystem device="/dev/my_vg/my_lv"
directory="/var/www/html" fstype="ext4" options="ro"

[root@z1 ~]# pcs resource create Website apache
configfile="/etc/httpd/conf/httpd.conf" statusurl="http://127.0.0.1/server-
status"

[root@z1 ~]# pcs resource create ClusterIP IPaddr2 ip=10.15.89.129
cidr_netmask=24

[root@z1 ~]# pcs resource group add apachegroup my_lvm my_fs ClusterIP Website
```

Revision History

Revision 1.1-2.404	**Mon Nov 25 2013**	**Rüdiger Landmann**
Rebuild with Publican 4.0.0		
Revision 1.1-2	**Wed Nov 20 2013**	**Steven Levine**
Version for 6.5 GA release		
Revision 0.1-4	**Wed Oct 2 2013**	**Steven Levine**
First printing of 6.5 beta draft		