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

Last Updated: 2018-03-23

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

Steven Levine
Red Hat Customer Content Services
slevine@redhat.com

Legal Notice

Abstract

Configuring the Red Hat High Availability Add-On with Pacemaker 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, see 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, see 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 documents are available in HTML, PDF, and RPM versions on the Red Hat Enterprise Documentation CD and online at https://access.redhat.com/site/documentation/.

1. 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 (2017-1-4T16:26)
```

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, 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 Enterprise Linux Release 6.6 and later.

NOTE

For information on best practices for deploying and upgrading Red Hat Enterprise Linux clusters using the High Availability Add-On and Red Hat Global File System 2 (GFS2) see the article "Red Hat Enterprise Linux Cluster, High Availability, and GFS Deployment Best Practices" on Red Hat Customer Portal at https://access.redhat.com/kb/docs/DOC-40821.

1.1. INSTALLING PACEMAKER CONFIGURATION TOOLS

You can use the following **yum install** command to install the Red Hat High Availability Add-On software packages along with all available: fence agents from the High Availability channel.

```
# yum install pcs pacemaker cman fence-agents
```

The **lvm2-cluster** and **gfs2-utils** packages are part of ResilientStorage channel. You can install them, as needed, with the following command.

```
# yum install lvm2-cluster gfs2-utils
```

WARNING

After you install the Red Hat High Availability Add-On packages, you should ensure that your software update preferences are set so that nothing is installed automatically. Installation on a running cluster can cause unexpected behaviors.

1.2. CONFIGURING THE IPTABLES FIREWALL TO ALLOW CLUSTER COMPONENTS

Table 1.1, “Ports to Enable for High Availability Add-On” shows the ports to enable for the Red Hat High Availability Add-On.

Table 1.1. Ports to Enable for High Availability Add-On

Port	When Required
TCP 2224	Required on all nodes (needed by the **pcsd** daemon)
TCP 3121	Required on all nodes if the cluster has any Pacemaker Remote nodes
TCP 21064	Required on all nodes if the cluster contains any resources requiring DLM (such as **clvm** or **GFS2**)
UDP 5405	Required on all cluster nodes (needed by **corosync**)
UDP 5404	Required on cluster nodes if **corosync** is configured for multicast UDP

1.3. THE CLUSTER AND PACEMAKER CONFIGURATION FILES

The configuration files for the Red Hat High Availability add-on are **cluster.conf** and **cib.xml**. Do not edit the **cib.xml** file directly; use the **pcs** interface instead.

The **cluster.conf** file provides the cluster parameters used by **corosync**, the cluster manager that Pacemaker is built on.

The **cib.xml** file is an XML file that represents both the cluster's configuration and current state of all resources in the cluster. This file is used by Pacemaker's Cluster Information Base (CIB). The contents of the CIB are automatically kept in sync across the entire cluster

1.4. CLUSTER CONFIGURATION CONSIDERATIONS

When configuring a Red Hat High Availability Add-On cluster, you must take the following considerations into account:

- Red Hat does not support cluster deployments greater than 16 full cluster nodes. It is possible, however, to scale beyond that limit with remote nodes running the **pacemaker_remote** service. For information on the **pacemaker_remote** service, see Section 8.4, "The pacemaker_remote Service".
- The use of Dynamic Host Configuration Protocol (DHCP) for obtaining an IP address on a network interface that is utilized by the **corosync** daemons is not supported. The DHCP client can periodically remove and re-add an IP address to its assigned interface during address renewal. This will result in **corosync** detecting a connection failure, which will result in fencing activity from any other nodes in the cluster using **corosync** for heartbeat connectivity.

CHAPTER 2. THE PCS COMMAND LINE INTERFACE

The **pcs** command line interface provides the ability to control and configure **corosync** and **pacemaker**.

The general format of the **pcs** command is as follows.

```
pcs [-f file] [-h] [commands]...
```

2.1. THE PCS COMMANDS

The **pcs** commands are as follows.

- **cluster**

  Configure cluster options and nodes. For information on the **pcs cluster** command, see Chapter 3, *Cluster Creation and Administration*.

- **resource**

  Create and manage cluster resources. For information on the **pcs cluster** command, see Chapter 5, *Configuring Cluster Resources*, Chapter 7, *Managing Cluster Resources*, and Chapter 8, *Advanced Resource types*.

- **stonith**

  Configure fence devices for use with Pacemaker. For information on the **pcs stonith** command, see Chapter 4, *Fencing: Configuring STONITH*.

- **constraint**

  Manage resource constraints. For information on the **pcs constraint** command, see Chapter 6, *Resource Constraints*.

- **property**

  Set Pacemaker properties. For information on setting properties with the **pcs property** command, see Chapter 10, *Pacemaker Cluster Properties*.

- **status**

  View current cluster and resource status. For information on the **pcs status** command, see Section 2.5, “Displaying Status”.

- **config**

  Display complete cluster configuration in user-readable form. For information on the **pcs config** command, see Section 2.6, “Displaying the Full Cluster Configuration”.

2.2. PCS USAGE HELP DISPLAY

You can use the **-h** option of **pcs** to display the parameters of a **pcs** command and a description of those parameters. For example, the following command displays the parameters of the **pcs resource** command. Only a portion of the output is shown.

```
# pcs resource -h
Usage: pcs resource [commands]...
Manage pacemaker resources
Commands:
    show [resource id] [--all]
        Show all currently configured resources or if a resource is
specified
        show the options for the configured resource.  If --all is
specified
        resource options will be displayed

    start <resource id>
        Start resource specified by resource_id
...
```

2.3. VIEWING THE RAW CLUSTER CONFIGURATION

Although you should not edit the cluster configuration file directly, you can view the raw cluster configuration with the **pcs cluster cib** command.

You can save the raw cluster configuration to a specified file with the **pcs cluster cib *filename*** as described in Section 2.4, “Saving a Configuration Change to a File” .

2.4. SAVING A CONFIGURATION CHANGE TO A FILE

When using the **pcs** command, you can use the **-f** option to save a configuration change to a file without affecting the active CIB.

If you have previously configured a cluster and there is already an active CIB, you use the following command to save the raw xml a file.

```
pcs cluster cib filename
```

For example, the following command saves the raw xml from the CIB into a file name **testfile**.

```
pcs cluster cib testfile
```

The following command creates a resource in the file **testfile1** but does not add that resource to the currently running cluster configuration.

```
# pcs -f testfile1 resource create VirtualIP ocf:heartbeat:IPaddr2
ip=192.168.0.120 cidr_netmask=24 op monitor interval=30s
```

You can push the current content of **testfile** to the CIB with the following command.

```
pcs cluster cib-push filename
```

2.5. DISPLAYING STATUS

You can display the status of the cluster and the cluster resources with the following command.

```
pcs status commands
```

If you do not specify a *commands* parameter, this command displays all information about the cluster and the resources. You display the status of only particular cluster components by specifying **resources**, **groups**, **cluster**, **nodes**, or **pcsd**.

2.6. DISPLAYING THE FULL CLUSTER CONFIGURATION

Use the following command to display the full current cluster configuration.

```
pcs config
```

2.7. DISPLAYING THE CURRENT PCS VERSION

The following command displays the current version of **pcs** that is running.

```
pcs --version
```

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. Start the **pcsd** on each node in the cluster.
2. Authenticate the nodes that will constitute the cluster.
3. Configure and sync the cluster nodes.
4. Start cluster services on the cluster nodes.

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

3.1.1. Starting the pcsd daemon

The following commands start the **pcsd** service and enable **pcsd** at system start. These commands should be run on each node in the cluster.

```
# service pcsd start
# chkconfig pcsd on
```

3.1.2. Authenticating the Cluster Nodes

The following command authenticates **pcs** to the **pcs** daemon on the nodes in the cluster.

- The user name for the **pcs** administrator must be **hacluster** on every node. It is recommended that the password for user **hacluster** be the same on each node.
- If you do not specify user name or password, the system will prompt you for those parameters for each node when you execute the command.
- If you do not specify any nodes, this command will authenticate **pcs** on the nodes that are specified with a **pcs cluster setup** command, if you have previously executed that command.

```
pcs cluster auth [node] [...] [-u username] [-p password]
```

For example, the following command authenticates user **hacluster** on **z1.example.com** for both of the nodes in the cluster that consist of **z1.example.com** and **z2.example.com**. This command prompts for the password for user **hacluster** on the cluster nodes.

```
root@z1 ~]# pcs cluster auth z1.example.com z2.example.com
Username: hacluster
Password:
z1.example.com: Authorized
z2.example.com: Authorized
```

Authorization tokens are stored in the file **~/.pcs/tokens** (or **/var/lib/pcsd/tokens**).

3.1.3. Configuring and Starting the Cluster Nodes

The following command configures the cluster configuration file and syncs the configuration to the specified nodes.

- If you specify the **--start** option, the command will also start the cluster services on the specified nodes. If necessary, you can also start the cluster services with a separate **pcs cluster start** command.

  When you create a cluster with the **pcs cluster setup --start** command or when you start cluster services with the **pcs cluster start** command, there may be a slight delay before the cluster is up and running. Before performing any subsequent actions on the cluster and its configuration, it is recommended that you use the **pcs cluster status** command to be sure that the cluster is up and running.

- If you specify the **--local** option, the command will perform changes on the local node only.

```
pcs cluster setup [--start] [--local] --name cluster_ name node1 [node2]
[...]
```

The following command starts cluster services on the specified node or nodes.

- If you specify the **--all** option, the command starts cluster services on all nodes.
- If you do not specify any nodes, cluster services are started on the local node only.

```
pcs cluster start [--all] [node] [...]
```

3.1.4. Enabling and Disabling Cluster Services

Use the following command to configure the cluster services to run on startup on the specified node or nodes.

- If you specify the **--all** option, the command enables cluster services on all nodes.
- If you do not specify any nodes, cluster services are enabled on the local node only.

```
pcs cluster enable [--all] [node] [...]
```

Use the following command to configure the cluster services not to run on startup on the specified node or nodes.

- If you specify the **--all** option, the command disables cluster services on all nodes.
- If you do not specify any nodes, cluster services are disabled on the local node only.

```
pcs cluster disable [--all] [node] [...]
```

You can verify whether cluster services are enabled by running the **pcs status** command on a running cluster. At the bottom of the output of this command, you should see that **pacemaker** is enabled, which ensures that the cluster starts on reboot. All other services should be disabled.

```
# pcs status
...
Daemon Status:
  cman: active/disabled
  corosync: active/disabled
  pacemaker: active/enabled
  pcsd: active/disabled
```

If the cluster is not running, you can verify whether cluster services are enabled by running the following **chkconfig** command. If 2, 3, 4, and 5 are on, the cluster is enabled.

```
# chkconfig --list pacemaker
pacemaker       0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

3.2. MANAGING CLUSTER NODES

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

3.2.1. Stopping Cluster Services

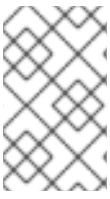
The following command stops cluster services on the specified node or nodes. As with the **pcs cluster start**, the **--all** option stops cluster services on all nodes and if you do not specify any nodes, cluster services are stopped on the local node only.

```
pcs cluster stop [--all] [node] [...]
```

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
```

3.2.2. Adding Cluster Nodes



NOTE

It is highly recommended that you add nodes to existing clusters only during a production maintenance window. This allows you to perform appropriate resource and deployment testing for the new node and its fencing configuration.

Use the following procedure to add a new node to an existing cluster. In this example, the existing cluster nodes are **clusternode-01.example.com**, **clusternode-02.example.com**, and **clusternode-03.example.com**. The new node is **newnode.example.com**.

On the new node to add to the cluster, perform the following tasks.

1. Install the cluster packages. If the cluster uses SBD, you must manually install the **sbd** package on the new node as well.

   ```
   [root@newnode ~]# yum install -y pacemaker cman pcs
   ```

2. To prevent **corosync** from starting without **cman**, execute the following command:

```
[root@newnode ~]# chkconfig corosync off
```

3. Set a password for the user ID **hacluster**. It is recommended that you use the same password for each node in the cluster.

```
[root@newnode ~]# passwd hacluster
Changing password for user hacluster.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

4. Start and enable **pcsd** on the new node:

```
[root@newnode ~]# service pcsd start
[root@newnode ~]# chkconfig pcsd on
```

On a node in the existing cluster, perform the following tasks.

1. Authenticate user **hacluster** on the new cluster node.

```
[root@clusternode-01 ~]# pcs cluster auth newnode.example.com
Username: hacluster
Password:
newnode.example.com: Authorized
```

2. Add the new node to the existing cluster. This command also syncs the cluster configuration file **cluster.conf** to all nodes in the cluster, including the new node you are adding.

```
[root@clusternode-01 ~]# pcs cluster node add newnode.example.com
clusternode-01.example.com: Corosync updated
clusternode-02.example.com: Corosync updated
clusternode-03.example.com: Corosync updated
newnode.example.com: Updated cluster.conf...
newnode.example.com: Starting Cluster...
```

On the new node to add to the cluster, perform the following tasks.

1. Enable cluster services on the new node. In Red Hat Enterprise Linux 6, the new node is started automatically by PCS.

```
[root@newnode ~]# pcs cluster enable
```

2. Ensure that you configure and test a fencing device for the new cluster node. For information on configuring fencing devices, see Chapter 4, *Fencing: Configuring STONITH*.

3.2.3. Removing Cluster Nodes

The following command shuts down the specified node and removes it from the cluster configuration file, **cluster.conf**, on all of the other nodes in the cluster. For information on removing all information about the cluster from the cluster nodes entirely, thereby destroying the cluster

permanently, see Section 3.4, “Removing the Cluster Configuration” .

```
pcs cluster node remove node
```

3.2.4. 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, see Chapter 6, *Resource Constraints*.

3.3. SETTING USER PERMISSIONS

By default, the root user and any user who is a member of the group **haclient** has full read/write access to the cluster configuration. As of Red Hat Enterprise Linux 6.6, you can use the **pcs acl** command to set permission for local users to allow read-only or read-write access to the cluster configuration by using access control lists (ACLs).

Setting permissions for local users is a two-step process:

1. Execute the **pcs acl role create...** command to create a *role* which defines the permissions for that role.

2. Assign the role you created to a user with the **pcs acl user create** command.

The following example procedure provides read-only access for a cluster configuration to a local user named **rouser**.

1. This procedure requires that the user **rouser** exists on the local system and that the user **rouser** is a member of the group **haclient**.

   ```
   # adduser rouser
   # usermod -a -G haclient rouser
   ```

2. Enable Pacemaker ACLs with the **enable-acl** cluster property.

   ▪

```
# pcs property set enable-acl=true --force
```

3. Create a role named **read-only** with read-only permissions for the cib.

```
# pcs acl role create read-only description="Read access to cluster"
read xpath /cib
```

4. Create the user **rouser** in the pcs ACL system and assign that user the **read-only** role.

```
# pcs acl user create rouser read-only
```

5. View the current ACLs.

```
# pcs acl
User: rouser
  Roles: read-only
Role: read-only
  Description: Read access to cluster
  Permission: read xpath /cib (read-only-read)
```

The following example procedure provides write access for a cluster configuration to a local user named **wuser**.

1. This procedure requires that the user **wuser** exists on the local system and that the user **wuser** is a member of the group **haclient**.

```
# adduser wuser
# usermod -a -G haclient wuser
```

2. Enable Pacemaker ACLs with the **enable-acl** cluster property.

```
# pcs property set enable-acl=true --force
```

3. Create a role named **write-access** with write permissions for the cib.

```
# pcs acl role create write-access description="Full access" write
xpath /cib
```

4. Create the user **wuser** in the pcs ACL system and assign that user the **write-access** role.

```
# pcs acl user create wuser write-access
```

5. View the current ACLs.

```
# pcs acl
User: rouser
  Roles: read-only
User: wuser
  Roles: write-access
Role: read-only
  Description: Read access to cluster
```

```
    Permission: read xpath /cib (read-only-read)
Role: write-access
    Description: Full Access
    Permission: write xpath /cib (write-access-write)
```

For further information about cluster ACLs, see the help screen for the **pcs acl** command.

3.4. REMOVING THE CLUSTER CONFIGURATION

To remove all cluster configuration files and stop all cluster services, thus permanently destroying a cluster, use the following command.

WARNING

This command permanently removes any cluster configuration that has been created. 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 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
```

The following command displays the status of the cluster resources.

```
pcs status resources
```

CHAPTER 4. 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 fence the node using 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.

4.1. AVAILABLE STONITH (FENCING) AGENTS

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

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

4.2. GENERAL PROPERTIES OF FENCING DEVICES

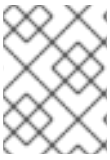
NOTE

To disable a fencing device/resource, you can set the **target-role** as you would for a normal resource.

NOTE

To prevent a specific node from using a fencing device, you can configure location constraints for the fencing resource.

Table 4.1, “General Properties of Fencing Devices” describes the general properties you can set for fencing devices. Refer to Section 4.3, “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, see Section 4.9, “Additional Fencing Configuration Options”

Table 4.1. General Properties of Fencing Devices

Field	Type	Default	Description
priority	integer	0	The priority of the stonith resource. Devices are tried in order of highest priority to lowest.

Field	Type	Default	Description
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)

4.3. 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 telnet/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
```

4.4. CREATING A FENCING DEVICE

The following command creates a stonith 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** to 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"
```

4.5. CONFIGURING STORAGE-BASED FENCE DEVICES WITH UNFENCING

When creating a SAN/storage fence device (that is, one that uses a non-power based fencing agent), you must set the meta option **provides=unfencing** when creating the **stonith** device. This ensures that a fenced node is unfenced before the node is rebooted and the cluster services are started on the node.

Setting the **provides=unfencing** meta option is not necessary when configuring a power-based fence device, since the device itself is providing power to the node in order for it to boot (and attempt to rejoin the cluster). The act of booting in this case implies that unfencing occurred.

The following command configures a stonith device named **my-scsi-shooter** that uses the **fence_scsi** fence agent, enabling unfencing for the device.

```
pcs stonith create my-scsi-shooter fence_scsi devices=/dev/sda meta
provides=unfencing
```

4.6. DISPLAYING FENCING DEVICES

The following command shows all currently 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]
```

4.7. 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
```

4.8. 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]
```

In a situation where no stonith device is able to fence a node even if it is no longer active, the cluster may not be able to recover the resources on the node. If this occurs, after manually ensuring that the node is powered down you can run the following command to confirm to the cluster that the node is powered down and free its resources for recovery.

WARNING

If the node you specify is not actually off, but running the cluster software or services normally controlled by the cluster, data corruption/cluster failure will occur.

```
pcs stonith confirm node
```

4.9. ADDITIONAL FENCING CONFIGURATION OPTIONS

Table 4.2, “Advanced Properties of Fencing Devices” . summarizes additional properties you can set for fencing devices. Note that these properties are for advanced use only.

Table 4.2. Advanced Properties of Fencing Devices

Field	Type	Default	Description

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.

Field	Type	Default	Description
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.

Field	Type	Default	Description
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.

4.10. 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
```

In the following example, there are two fence devices configured for node **rh7-2**: an ilo fence device called **my_ilo** and an apc fence device called **my_apc**. These commands sets up fence levels so that if the device **my_ilo** fails and is unable to fence the node, then Pacemaker will attempt to use the device **my_apc**. This example also shows the output of the **pcs stonith level** command after the levels are configured.

```
# pcs stonith level add 1 rh7-2 my_ilo
# pcs stonith level add 2 rh7-2 my_apc
# pcs stonith level
 Node: rh7-2
  Level 1 - my_ilo
  Level 2 - my_apc
```

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

```
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
```

4.11. CONFIGURING FENCING FOR REDUNDANT POWER SUPPLIES

When configuring fencing for redundant power supplies, the cluster must ensure that when attempting to reboot a host, both power supplies are turned off before either power supply is turned back on.

If the node never completely loses power, the node may not release its resources. This opens up the possibility of nodes accessing these resources simultaneously and corrupting them.

Prior to Red Hat Enterprise Linux 6.8, you needed to explicitly configure different versions of the devices which used either the 'on' or 'off' actions. Since Red Hat Enterprise Linux 6.8, it is now only required to define each device once and to specify that both are required to fence the node, as in the following example.

```
# pcs stonith create apc1 fence_apc_snmp ipaddr=apc1.example.com
login=user passwd='7a4D#1j!pz864'
pcmk_host_map="node1.example.com:1;node2.example.com:2"

# pcs stonith create apc2 fence_apc_snmp ipaddr=apc2.example.com
login=user passwd='7a4D#1j!pz864'
pcmk_host_map="node1.example.com:1;node2.example.com:2"

# pcs stonith level add 1 node1.example.com apc1,apc2
# pcs stonith level add 1 node2.example.com apc1,apc2
```

4.12. CONFIGURING ACPI FOR USE WITH INTEGRATED FENCE DEVICES

If your cluster uses integrated fence devices, you must configure ACPI (Advanced Configuration and Power Interface) to ensure immediate and complete fencing.

If a cluster node is configured to be fenced by an integrated fence device, disable ACPI Soft-Off for that node. Disabling ACPI Soft-Off allows an integrated fence device to turn off a node immediately and completely rather than attempting a clean shutdown (for example, **shutdown -h now**). Otherwise, if ACPI Soft-Off is enabled, an integrated fence device can take four or more seconds to turn off a node (refer to note that follows). In addition, if ACPI Soft-Off is enabled and a node panics or freezes during shutdown, an integrated fence device may not be able to turn off the node. Under those circumstances, fencing is delayed or unsuccessful. Consequently, when a node is fenced with an integrated fence device and ACPI Soft-Off is enabled, a cluster recovers slowly or requires administrative intervention to recover.

NOTE

The amount of time required to fence a node depends on the integrated fence device used. Some integrated fence devices perform the equivalent of pressing and holding the power button; therefore, the fence device turns off the node in four to five seconds. Other integrated fence devices perform the equivalent of pressing the power button momentarily, relying on the operating system to turn off the node; therefore, the fence device turns off the node in a time span much longer than four to five seconds.

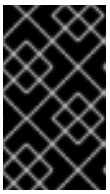
To disable ACPI Soft-Off, use **chkconfig** management and verify that the node turns off immediately when fenced. The preferred way to disable ACPI Soft-Off is with **chkconfig** management: however, if that method is not satisfactory for your cluster, you can disable ACPI Soft-Off with one of the following alternate methods:

- Changing the BIOS setting to "instant-off" or an equivalent setting that turns off the node without delay

NOTE

Disabling ACPI Soft-Off with the BIOS may not be possible with some computers.

- Appending **acpi=off** to the kernel boot command line of the **/boot/grub/grub.conf** file



IMPORTANT

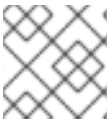
This method completely disables ACPI; some computers do not boot correctly if ACPI is completely disabled. Use this method *only* if the other methods are not effective for your cluster.

The following sections provide procedures for the preferred method and alternate methods of disabling ACPI Soft-Off:

- Section 4.12.1, "Disabling ACPI Soft-Off with **chkconfig** Management" — Preferred method
- Section 4.12.2, "Disabling ACPI Soft-Off with the BIOS" — First alternate method
- Section 4.12.3, "Disabling ACPI Completely in the **grub.conf** File" — Second alternate method

4.12.1. Disabling ACPI Soft-Off with chkconfig Management

You can use **chkconfig** management to disable ACPI Soft-Off either by removing the ACPI daemon (**acpid**) from **chkconfig** management or by turning off **acpid**.



NOTE

This is the preferred method of disabling ACPI Soft-Off.

Disable ACPI Soft-Off with **chkconfig** management at each cluster node as follows:

1. Run either of the following commands:
   - **chkconfig --del acpid** — This command removes **acpid** from **chkconfig** management.

     — OR —

   - **chkconfig --level 345 acpid off** — This command turns off **acpid**.
2. Reboot the node.
3. When the cluster is configured and running, verify that the node turns off immediately when fenced. For information on testing a fence device, see How to test fence devices and fencing configuration in a RHEL 5, 6, or 7 High Availability cluster?.

4.12.2. Disabling ACPI Soft-Off with the BIOS

The preferred method of disabling ACPI Soft-Off is with **chkconfig** management (Section 4.12.1, "Disabling ACPI Soft-Off with **chkconfig** Management"). However, if the preferred method is not effective for your cluster, follow the procedure in this section.

NOTE

Disabling ACPI Soft-Off with the BIOS may not be possible with some computers.

You can disable ACPI Soft-Off by configuring the BIOS of each cluster node as follows:

1. Reboot the node and start the **BIOS CMOS Setup Utility** program.
2. Navigate to the **Power** menu (or equivalent power management menu).
3. At the **Power** menu, set the **Soft-Off by PWR-BTTN** function (or equivalent) to **Instant-Off** (or the equivalent setting that turns off the node by means of the power button without delay). Example 4.1, "**BIOS CMOS Setup Utility: Soft-Off by PWR-BTTN set to Instant-Off**" shows a **Power** menu with **ACPI Function** set to **Enabled** and **Soft-Off by PWR-BTTN** set to **Instant-Off**.

NOTE

The equivalents to **ACPI Function**, **Soft-Off by PWR-BTTN**, and **Instant-Off** may vary among computers. However, the objective of this procedure is to configure the BIOS so that the computer is turned off by means of the power button without delay.

4. Exit the **BIOS CMOS Setup Utility** program, saving the BIOS configuration.
5. When the cluster is configured and running, verify that the node turns off immediately when fenced. For information on testing a fence device, see How to test fence devices and fencing configuration in a RHEL 5, 6, or 7 High Availability cluster?.

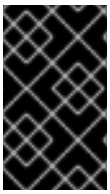
Example 4.1. BIOS CMOS Setup Utility: Soft-Off by PWR-BTTN set to Instant-Off

```
+---------------------------------------------|-------------------+
|    ACPI Function             [Enabled]      |    Item Help      |
|    ACPI Suspend Type         [S1(POS)]      |-------------------|
|  x Run VGABIOS if S3 Resume   Auto          |   Menu Level   *  |
|    Suspend Mode              [Disabled]     |                   |
|    HDD Power Down            [Disabled]     |                   |
|    Soft-Off by PWR-BTTN      [Instant-Off   |                   |
|    CPU THRM-Throttling       [50.0%]        |                   |
|    Wake-Up by PCI card       [Enabled]      |                   |
|    Power On by Ring          [Enabled]      |                   |
|    Wake Up On LAN            [Enabled]      |                   |
|  x USB KB Wake-Up From S3     Disabled      |                   |
|    Resume by Alarm           [Disabled]     |                   |
|  x  Date(of Month) Alarm       0            |                   |
|  x  Time(hh:mm:ss) Alarm       0 :  0 :     |                   |
|    POWER ON Function         [BUTTON ONLY   |                   |
|  x KB Power ON Password       Enter         |                   |
|  x Hot Key Power ON           Ctrl-F1       |                   |
|                                             |                   |
|                                             |                   |
+---------------------------------------------|-------------------+
```

This example shows **ACPI Function** set to **Enabled**, and **Soft-Off by PWR-BTTN** set to **Instant-Off**.

4.12.3. Disabling ACPI Completely in the grub.conf File

The preferred method of disabling ACPI Soft-Off is with **chkconfig** management (Section 4.12.1, "Disabling ACPI Soft-Off with **chkconfig** Management"). If the preferred method is not effective for your cluster, you can disable ACPI Soft-Off with the BIOS power management (Section 4.12.2, "Disabling ACPI Soft-Off with the BIOS"). If neither of those methods is effective for your cluster, you can disable ACPI completely by appending **acpi=off** to the kernel boot command line in the **grub.conf** file.



IMPORTANT

This method completely disables ACPI; some computers do not boot correctly if ACPI is completely disabled. Use this method *only* if the other methods are not effective for your cluster.

You can disable ACPI completely by editing the **grub.conf** file of each cluster node as follows:

1. Open **/boot/grub/grub.conf** with a text editor.
2. Append **acpi=off** to the kernel boot command line in **/boot/grub/grub.conf** (see Example 4.2, "Kernel Boot Command Line with **acpi=off** Appended to It").
3. Reboot the node.
4. When the cluster is configured and running, verify that the node turns off immediately when fenced. For information on testing a fence device, see How to test fence devices and fencing configuration in a RHEL 5, 6, or 7 High Availability cluster?.

Example 4.2. Kernel Boot Command Line with acpi=off Appended to It

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this
file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/mapper/vg_doc01-lv_root
#          initrd /initrd-[generic-]version.img
#boot=/dev/hda
default=0
timeout=5
serial --unit=0 --speed=115200
terminal --timeout=5 serial console
title Red Hat Enterprise Linux Server (2.6.32-193.el6.x86_64)
        root (hd0,0)
        kernel /vmlinuz-2.6.32-193.el6.x86_64 ro
root=/dev/mapper/vg_doc01-lv_root console=ttyS0,115200n8 acpi=off
        initrd /initramfs-2.6.32-131.0.15.el6.x86_64.img
```

In this example, **acpi=off** has been appended to the kernel boot command line – the line starting with "kernel /vmlinuz-2.6.32-193.el6.x86_64.img".

CHAPTER 5. CONFIGURING CLUSTER RESOURCES

This chapter provides information on configuring resources in a cluster.

5.1. 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 VirtualIP 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 VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.120
cidr_netmask=24 op monitor interval=30s
```

Alternately, you can omit the *standard* and *provider* fields and use the following command. This will default to a standard of **ocf** and a provider of **heartbeat**.

```
# pcs resource create VirtualIP IPaddr2 ip=192.168.0.120 cidr_netmask=24
op monitor interval=30s
```

As of the Red Hat Enterprise Linux 6.8 release, resources start as soon as their state has been confirmed on all nodes and all dependencies have been satisfied, rather than waiting for the state of all resources to be confirmed. This allows for faster startup of some services, and more even startup load.

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 **VirtualIP**

```
# pcs resource delete VirtualIP
```

- For information on the *resource_id*, *standard*, *provider*, and *type* fields of the **pcs resource create** command, see Section 5.2, “Resource Properties”.
- For information on defining resource parameters for individual resources, see Section 5.3, “Resource-Specific Parameters”.
- For information on defining resource meta options, which are used by the cluster to decide how a resource should behave, see Section 5.4, “Resource Meta Options” .
- For information on defining the operations to perform on a resource, see Section 5.6, “Resource Operations”.

5.2. 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 5.1, "Resource Properties" describes these properties.

Table 5.1. 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 5.2, "Commands to Display Resource Properties". summarizes the commands that display the available resource properties.

Table 5.2. 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.

5.3. RESOURCE-SPECIFIC PARAMETERS

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

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

For example, the following command displays 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 physical volumes available. It helps to set to
true, when you are using mirroring logical volumes.
```

5.4. 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 5.3, "Resource Meta Options" describes this options.

Table 5.3. 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.

Field	Default	Description
requires	Calculated	Indicates under what conditions the resource can be started. Defaults to **fencing** except under the conditions noted below. Possible values: * **nothing** - The cluster can always start the resource. * **quorum** - The cluster can only start this resource if a majority of the configured nodes are active. This is the default value if **stonith-enabled** is **false** or the resource's **standard** is **stonith**. * **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*. This is the default value if the **provides=unfencing stonith** meta option has been set for a fencing device. For information on the **provides=unfencing stonith** meta option, see Section 4.5, "Configuring Storage-Based Fence Devices with unfencing".
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 toSection 7.2, "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 toSection 7.2, "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 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 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 VirtualIP 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
```

In the following example, there is an existing resource named **dummy_resource**. This command sets the **failure-timeout** meta option to 20 seconds, so that the resource can attempt to restart on the same node in 20 seconds.

```
# pcs resource meta dummy_resource failure-timeout=20s
```

After executing this command, you can display the values for the resource to verity that **failure-timeout=20s** is set.

```
# pcs resource show dummy_resource
 Resource: dummy_resource (class=ocf provider=heartbeat type=Dummy)
  Meta Attrs: failure-timeout=20s
  Operations: start interval=0s timeout=20 (dummy_resource-start-timeout-
20)
              stop interval=0s timeout=20 (dummy_resource-stop-timeout-
20)
              monitor interval=10 timeout=20 (dummy_resource-monitor-
interval-10)
```

For information on resource clone meta options, see Section 8.1, "Resource Clones". For information on resource master meta options, see Section 8.2, "Multi-State Resources: Resources That Have Multiple Modes".

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 of their starting 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] --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, **IPaddr** first, then **Email**).
- Resources are stopped in the reverse order in which you specify them. (**Email** first, then **IPaddr**).

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

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

Obviously as the group grows bigger, the reduced configuration effort of creating resource groups can become significant.

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, see Table 5.3, "Resource Meta Options".

5.5.2. 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.

5.6. 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 monitoring operation, with an interval that is determined by the resource agent. If the resource agent does not provide a default monitoring interval, the pcs command will create a monitoring operation with an interval of 60 seconds.

Table 5.4, "Properties of an Operation" summarizes the properties of a resource monitoring operation.

Table 5.4. 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".

Field	Description
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 **restart**.
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 **VirtualIP** 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.

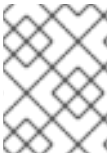
```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.99
cidr_netmask=24 nic=eth2 op monitor interval=30s
```

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]
```

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.

To change the values of a monitoring option, you remove the existing operation, then add the new operation. For example, you can create a **VirtualIP** with the following command.

```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.99
cidr_netmask=24 nic=eth2
```

By default, this command creates these operations.

```
Operations: start interval=0s timeout=20s (VirtualIP-start-timeout-20s)
            stop interval=0s timeout=20s (VirtualIP-stop-timeout-20s)
            monitor interval=10s timeout=20s (VirtualIP-monitor-interval-
10s)
```

The change the stop timeout operation, execute the following commands.

```
# pcs resource op remove VirtualIP stop interval=0s timeout=20s
# pcs resource op add VirtualIP stop interval=0s timeout=40s

# pcs resource show VirtualIP
 Resource: VirtualIP (class=ocf provider=heartbeat type=IPaddr2)
  Attributes: ip=192.168.0.99 cidr_netmask=24 nic=eth2
  Operations: start interval=0s timeout=20s (VirtualIP-start-timeout-20s)
              monitor interval=10s timeout=20s (VirtualIP-monitor-
interval-10s)
              stop interval=0s timeout=40s (VirtualIP-name-stop-interval-
0s-timeout-40s)
```

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** command.

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
```

5.7. 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 **VirtualIP** and a resource named **WebSite**, the **pcs resource show** command yields the following output.

```
# pcs resource show
 VirtualIP (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: VirtualIP (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 **VirtualIP**.

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

5.8. 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 **VirtualIP**, the command to change the value of the **ip** parameter, and the values following the update command.

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

5.9. MULTIPLE MONITORING OPERATIONS

You can configure a single resource with as many monitor operations as a resource agent supports. In this way you can do a superficial health check every minute and progressively more intense ones at higher intervals.

NOTE

When configuring multiple monitor operations, you must ensure that no two operations are performed at the same interval.

To configure additional monitoring operations for a resource that supports more in-depth checks at different levels, you add an **OCF_CHECK_LEVEL=*n*** option.

For example, if you configure the following **IPaddr2** resource, by default this creates a monitoring operation with an interval of 10 seconds and a timeout value of 20 seconds.

```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.99
cidr_netmask=24 nic=eth2
```

If the Virtual IP supports a different check with a depth of 10, the following command causes Pacemaker to perform the more advanced monitoring check every 60 seconds in addition to the normal Virtual IP check every 10 seconds. (As noted, you should not configure the additional monitoring operation with a 10-second interval as well.)

```
# pcs resource op add VirtualIP monitor interval=60s OCF_CHECK_LEVEL=10
```

5.10. ENABLING AND DISABLING CLUSTER RESOURCES

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

```
pcs resource enable resource_id
```

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

```
pcs resource disable resource_id
```

5.11. CLUSTER RESOURCES CLEANUP

If a resource has failed, a failure message appears when you display the cluster status. If you resolve that resource, you can clear that failure status with the **pcs resource cleanup** command. This command resets the resource status and failcount, telling the cluster to forget the operation history of a resource and re-detect its current state.

The following command cleans up the resource specified by **resource_id**.

```
pcs resource cleanup resource_id
```

CHAPTER 6. 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 6.1, "Location Constraints".
- **order** constraints — An order constraint determines the order in which the resources run. Order constraints are described in Section 6.2, "Order Constraints".
- **colocation** constraints — A colocation constraint determines where resources will be placed relative to other resources. Colocation constraints are described in Section 6.3, "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".

6.1. LOCATION CONSTRAINTS

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

Table 6.1, "Location Constraint Options". summarizes the options for configuring location constraints.

Table 6.1. 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. A Value of **INFINITY** changes "should" to "must";**INFINITY** is the default score value for a resource location constraint.

Field	Description
resource-discovery	Value to indicate the preference for whether Pacemaker should perform resource discovery on this node for the specified resource. Limiting resource discovery to a subset of nodes the resource is physically capable of running on can significantly boost performance when a large set of nodes is present. When pacemaker_remote is in use to expand the node count into the hundreds of nodes range, this option should be considered. Possible values include: **always**: Always perform resource discovery for the specified resource on this node. **never**: Never perform resource discovery for the specified resource on this node. **exclusive**: Perform resource discovery for the specified resource only on this node (and other nodes similarly marked as **exclusive**). Multiple location constraints using **exclusive** discovery for the same resource across different nodes creates a subset of nodes **resource-discovery** is exclusive to. If a resource is marked for **exclusive** discovery on one or more nodes, that resource is only allowed to be placed within that subset of nodes. Note that setting this option to **never** or **exclusive** allows the possibility for the resource to be active in those locations without the cluster's knowledge. This can lead to the resource being active in more than one location if the service is started outside the cluster's control (for example, by **systemd** or by an administrator). This can also occur if the **resource-discovery** property is changed while part of the cluster is down or suffering split-brain, or if the **resource-discovery** property is changed for a resource and node while the resource is active on that node. For this reason, using this option is appropriate only when you have more than eight nodes and there is a way to guarantee that the resource can run only in a particular location (for example, when the required software is not installed anywhere else). **always** is the default **resource-discovery** value for a resource location constraint.

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 6.1.1, “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 6.1.2, “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.

6.1.1. 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
```

6.1.2. 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 6.1.1, “Configuring an "Opt-In" Cluster”. Both resources can fail over to node **example-3** if their preferred node fails, since every node has an implicit score of 0.

```
# 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.

6.2. 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 6.2, "Properties of an Order Constraint". summarizes the properties and options for configuring order constraints.

Table 6.2. 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 a resource. 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, see Section 8.2, "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, see Section 6.2.2, "Advisory Ordering". * **`Mandatory`** - Always (default value). 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, see Section 6.2.1, "Mandatory Ordering". * **`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`**

6.2.1. Mandatory Ordering

A mandatory constraints indicates that the second resource you specify cannot run without the first resource you specify being active. This is the default value of the **`kind`** option. Leaving the default value ensures 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.

6.2.2. 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.

The following command configures an advisory ordering constraint for the resources named **VirtualIP** and **dummy_resource**,

```
# pcs constraint order VirtualIP then dummy_resource kind=Optional
```

6.2.3. 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. If your configuration requires that you create a set of resources that is colocated and started in order, you can configure a resource group that contains those resources, as described in Section 5.5, "Resource Groups". If, however, you need to configure resources to start in order and the resources are not necessarily colocated, you can create an order constraint on a set or sets of resources with the **pcs constraint order set** command.

You can set the following options for a set of resources with the **pcs constraint order set** command.

- **sequential**, which can be set to **true** or **false** to indicate whether the set of resources must be ordered relative to each other.

  Setting **sequential** to **false** allows a set to be ordered relative to other sets in the ordering constraint, without its members being ordered relative to each other. Therefore, this option makes sense only if multiple sets are listed in the constraint; otherwise, the constraint has no effect.

- **require-all**, which can be set to **true** or **false** to indicate whether all of the resources in the set must be active before continuing. Setting **require-all** to **false** means that only one resource in the set needs to be started before continuing on to the next set. Setting **require-all** to **false** has no effect unless used in conjunction with unordered sets, which are sets for which **sequential** is set to **false**.

- **action**, which can be set to **start**, **promote**, **demote** or **stop**, as described in Table 6.2, "Properties of an Order Constraint".

You can set the following constraint options for a set of resources following the **setoptions** parameter of the **pcs constraint order set** command.

- **id**, to provide a name for the constraint you are defining.
- **score**, to indicate the degree of preference for this constraint. For information on this option, see Table 6.3, "Properties of a Colocation Constraint".

```
pcs constraint order set resource1 resource2 [resourceN]... [options] [set
resourceX resourceY ... [options]] [setoptions [constraint_options]]
```

If you have three resources named **D1**, **D2**, and **D3**, the following command configures them as an ordered resource set.

```
# pcs constraint order set D1 D2 D3
```

6.2.4. Removing Resources From Ordering Constraints

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

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

6.3. 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 8.2, “Multi-State Resources: Resources That Have Multiple Modes”.

Table 6.3, “Properties of a Colocation Constraint”. summarizes the properties and options for configuring colocation constraints.

Table 6.3. 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.

Field	Description
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*.

6.3.1. 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 with myresource2 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.

6.3.2. Advisory Placement

If mandatory placement is about "must" and "must not", then advisory placement is the "I would 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.

6.3.3. Colocating Sets of Resources

If your configuration requires that you create a set of resources that is colocated and started in order, you can configure a resource group that contains those resources, as described in Section 5.5, "Resource Groups". If, however, you need to colocate a set of resources but the resources do not necessarily need to start in order, you can create a colocation constraint on a set or sets of resources with the **pcs constraint colocation set** command.

You can set the following options for a set of resources with the **pcs constraint colocation set** command.

- **sequential**, which can be set to **true** or **false** to indicate whether the members of the set must be colocated with each other.

  Setting **sequential** to **false** allows the members of this set to be colocated with another set listed later in the constraint, regardless of which members of this set are active. Therefore, this option makes sense only if another set is listed after this one in the constraint; otherwise, the constraint has no effect.

- **role**, which can be set to **Stopped**, **Started**, **Master**, or **Slave**. For information on multi-state resources, see Section 8.2, "Multi-State Resources: Resources That Have Multiple Modes".

You can set the following constraint options for a set of resources following the **setoptions** parameter of the **pcs constraint colocation set** command.

- **kind**, to indicate how to enforce the constraint. For information on this option, see Table 6.2, "Properties of an Order Constraint".

- **symmetrical**, to indicate the order in which to stop the resources. If true, which is the default, stop the resources in the reverse order. Default value: **true**

- **id**, to provide a name for the constraint you are defining.

When listing members of a set, each member is colocated with the one before it. For example, "set A B" means "B is colocated with A". However, when listing multiple sets, each set is colocated with the one after it. For example, "set C D sequential=false set A B" means "set C D (where C and D have no relation between each other) is colocated with set A B (where B is colocated with A)".

The following command creates a colocation constraint on a set or sets of resources.

```
pcs constraint colocation set resource1 resource2 [resourceN]... [options]
[set resourceX resourceY ... [options]] [setoptions [constraint_options]]
```

6.3.4. Removing Colocation Constraints

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

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

6.4. DISPLAYING CONSTRAINTS

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

The following command lists 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.

```
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 ...
```

CHAPTER 7. MANAGING CLUSTER RESOURCES

This chapter describes various commands you can use to manage cluster resources. It provides information on the following procedures.

- Section 7.1, “Manually Moving Resources Around the Cluster”
- Section 7.2, “Moving Resources Due to Failure”
- Section 7.4, “Enabling, Disabling, and Banning Cluster Resources”
- Section 7.5, “Disabling a Monitor Operations”

7.1. 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, see Section 3.2.4, “Standby Mode”.

You can move individually specified resources in either of the following ways.

- You can use the **pcs resource move** command to move a resource off a node on which it is currently running, as described in Section 7.1.1, “Moving a Resource from its Current Node” .
- You can use the **pcs resource relocate run** command to move a resource to its preferred node, as determined by current cluster status, constraints, location of resources and other settings. For information on this command, see Section 7.1.2, “Moving a Resource to its Preferred Node”.

7.1.1. Moving a Resource from its Current Node

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. Specify the **destination_node**. if you want to indicate on which node to run the resource that you are moving.

```
pcs resource move resource_id [destination_node] [--master]
[lifetime=lifetime]
```

NOTE

When you execute the **pcs resource move** command, this adds a constraint to the resource to prevent it from running on the node on which it is currently running. You can execute the **pcs resource clear** or the **pcs constraint delete** command to remove the constraint. This does not necessarily move the resources back to the original node; where the resources can run at that point depends on how you have configured your resources initially.

If you specify the **--master** parameter of the **pcs resource ban** command, the scope of the constraint is limited to the master role and you must specify *master_id* rather than *resource_id*.

You can optionally configure a **lifetime** parameter for the **pcs resource move** command to indicate a period of time the constraint should remain. You specify the units of a **lifetime** parameter according to the format defined in ISO 8601, which requires that you specify the unit as a capital letter such as Y (for years), M (for months), W (for weeks), D (for days), H (for hours), M (for minutes), and S (for seconds).

To distinguish a unit of minutes(M) from a unit of months(M), you must specify PT before indicating the value in minutes. For example, a **lifetime** parameter of 5M indicates an interval of five months, while a **lifetime** parameter of PT5M indicates an interval of five minutes.

The **lifetime** parameter is checked at intervals defined by the **cluster-recheck-interval** cluster property. By default this value is 15 minutes. If your configuration requires that you check this parameter more frequently, you can reset this value with the following command.

```
pcs property set cluster-recheck-interval=value
```

You can optionally configure a **--wait[=*n*]** parameter for the **pcs resource ban** command to indicate the number of seconds to wait for the resource to start on the destination node before returning 0 if the resource is started or 1 if the resource has not yet started. If you do not specify n, the default resource timeout will be used.

The following command moves the resource **resource1** to node **example-node2** and prevents it from moving back to the node on which it was originally running for one hour and thirty minutes.

```
pcs resource move resource1 example-node2 lifetime=PT1H30M
```

The following command moves the resource **resource1** to node **example-node2** and prevents it from moving back to the node on which it was originally running for thirty minutes.

```
pcs resource move resource1 example-node2 lifetime=PT30M
```

For information on resource constraints, see Chapter 6, *Resource Constraints*.

7.1.2. Moving a Resource to its Preferred Node

After a resource has moved, either due to a failover or to an administrator manually moving the node, it will not necessarily move back to its original node even after the circumstances that caused the failover have been corrected. To relocate resources to their preferred node, use the following command. A preferred node is determined by the current cluster status, constraints, resource location, and other settings and may change over time.

```
pcs resource relocate run [resource1] [resource2] ...
```

If you do not specify any resources, all resource are relocated to their preferred nodes.

This command calculates the preferred node for each resource while ignoring resource stickiness. After calculating the preferred node, it creates location constraints which will cause the resources to move to their preferred nodes. Once the resources have been moved, the constraints are deleted automatically. To remove all constraints created by the **pcs resource relocate run** command,

you can run the **pcs resource relocate clear** command. To display the current status of resources and their optimal node ignoring resource stickiness, run the **pcs resource relocate show** command.

7.2. 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.

The following example adds a migration threshold of 10 to the resource named **dummy_resource**, which indicates that the resource will move to a new node after 10 failures.

```
# pcs resource meta dummy_resource migration-threshold=10
```

You can add a migration threshold to the defaults for the whole cluster with the following command.

```
# pcs resource defaults migration-threshold=10
```

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

There are two exceptions to the migration threshold concept; they occur when a resource either fails to start or fails to stop. If the cluster property **start-failure-is-fatal** is set to **true** (which is the default), start failures cause the 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.

7.3. MOVING RESOURCES DUE TO CONNECTIVITY CHANGES

Setting up the cluster to move resources when external connectivity is lost is a two-step process.

1. Add a **ping** resource to the cluster. The **ping** resource uses the system utility of the same name to test if a list of machines (specified by DNS host name or IPv4/IPv6 address) are reachable and uses the results to maintain a node attribute called **pingd**.

2. Configure a location constraint for the resource that will move the resource to a different node when connectivity is lost.

Table 5.1, "Resource Properties" describes the properties you can set for a **ping** resource.

Table 7.1. Properties of a ping resources

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

The following example command creates a **ping** resource that verifies connectivity to **www.example.com**. In practice, you would verify connectivity to your network gateway/router. You configure the **ping** resource as a clone so that the resource will run on all cluster nodes.

```
# pcs resource create ping ocf:pacemaker:ping dampen=5s multiplier=1000
host_list=www.example.com --clone
```

The following example configures a location constraint rule for the existing resource named **Webserver**. This will cause the **Webserver** resource to move to a host that is able to ping **www.example.com** if the host that it is currently running on cannot ping **www.example.com**

```
# pcs constraint location Webserver rule score=-INFINITY pingd lt 1 or
not_defined pingd
```

7.4. ENABLING, DISABLING, AND BANNING CLUSTER RESOURCES

In addition to the **pcs resource move** command described in Section 7.1, "Manually Moving Resources Around the Cluster", there are a variety of other 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, and so), the resource may remain started. If you specify the **--wait** option, **pcs** will wait up to 'n' seconds for the resource to stop and then return 0 if the resource is stopped or 1 if the resource has not stopped. If 'n' is not specified it defaults to 60 minutes.

```
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 'n' seconds for the resource to start and then return 0 if the resource is started or 1 if the resource has not started. If 'n' is not specified it defaults to 60 minutes.

```
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] [--master] [lifetime=lifetime] [--
wait[=n]]
```

Note that when you execute the **pcs resource ban** command, this adds a -INFINITY location constraint to the resource to prevent it from running on the indicated node. You can execute the **pcs resource clear** or the **pcs constraint delete** command to remove the constraint. 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, see Chapter 6, *Resource Constraints*.

If you specify the **--master** parameter of the **pcs resource ban** command, the scope of the constraint is limited to the master role and you must specify *master_id* rather than *resource_id*.

You can optionally configure a **lifetime** parameter for the **pcs resource ban** command to indicate a period of time the constraint should remain. For information on specifying units for the **lifetime** parameter and on specifying the intervals at which the **lifetime** parameter should be checked, see Section 7.1, “Manually Moving Resources Around the Cluster” .

You can optionally configure a **--wait[=*n*]** parameter for the **pcs resource ban** command to indicate the number of seconds to wait for the resource to start on the destination node before returning 0 if the resource is started or 1 if the resource has not yet started. If you do not specify n, the default resource timeout will be used.

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
```

7.5. DISABLING A MONITOR OPERATIONS

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

7.6. MANAGED RESOURCES

You can set a resource to unmanaged mode, which indicates that the resource is still in the configuration but Pacemaker does not manage the resource.

The following command sets the indicated resources to unmanaged mode.

```
pcs resource unmanage resource1  [resource2] ...
```

The following command sets resources to managed mode, which is the default state.

```
pcs resource manage resource1  [resource2] ...
```

You can specify the name of a resource group with the **pcs resource manage** or **pcs resource unmanage** command. The command will act on all of the resources in the group, so that you can manage or unmanage all of the resource in a group with a single command and then manage the contained resources individually.

CHAPTER 8. ADVANCED RESOURCE TYPES

This chapter describes advanced resource types that Pacemaker supports.

8.1. 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.

NOTE

Only resources that can be active on multiple nodes at the same time are suitable for cloning. For example, a **Filesystem** resource mounting a non-clustered file system such as **ext4** from a shared memory device should not be cloned. Since the **ext4** partition is not cluster aware, this file system is not suitable for read/write operations occurring from multiple nodes at the same time.

8.1.1. 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]
```

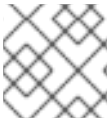
The name of the clone will be ***resource_id*-clone**.

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.

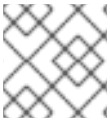
```
pcs resource clone resource_id | group_name [clone_options]...
```

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



NOTE

You need to configure resource configuration changes on one node only.



NOTE

When configuring constraints, always use the name of the group or clone.

When you create a clone of a resource, the clone takes on the name of the resource with **-clone** appended to the name. The following commands creates a resource of type **apache** named **webfarm** and a clone of that resource named **webfarm-clone**.

```
# pcs resource create webfarm apache 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, see Section 5.1, "Resource Creation".

Table 8.1, "Resource Clone Options" describes the options you can specify for a cloned resource.

Table 8.1. Resource Clone Options

Field	Description
priority, target-role, is-managed	Options inherited from resource that is being cloned, as described in Table 5.3, "Resource Meta Options".
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; the default value is **1**.
clone-min	The number of instances that must be running before any dependent resources can run. This parameter can be of particular use for services behind a virtual IP and HAProxy, such as is often required for an OpenStack platform.
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**. The default value is **false**.
globally-unique	Does each copy of the clone perform a different function? Allowed values: **false**, **true** If the value of this option is **false**, these resources behave identically everywhere they are running and thus there can be only one copy of the clone active per machine. If the value of this option is **true**, a copy of the clone running on one machine is not equivalent to another instance, whether that instance is running on another node or on the same node. The default value is **true** if the value of **clone-node-max** is greater than one; otherwise the default value is **false**.
ordered	Should the copies be started in series (instead of in parallel). Allowed values: **false**, **true**. The default value is **false**.
interleave	Changes the behavior of ordering constraints (between clones/masters) so that copies of the first clone can start or stop as soon as the copy on the same node of the second clone has started or stopped (rather than waiting until every instance of the second clone has started or stopped). Allowed values: **false**, **true**. The default value is **false**.

8.1.2. Clone Constraints

In most cases, a clone will have a single copy on each active cluster node. You can, however, set **clone-max** for the resource clone to a value that is less than the total number of nodes in the cluster. If this is 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 except that the clone's id must be used.

The following command creates a location constraint for the cluster to preferentially assign resource clone **webfarm-clone** to **node1**.

```
# pcs constraint location webfarm-clone prefers node1
```

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

```
# pcs constraint order start webfarm-clone then webfarm-stats
```

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.

The following command creates a colocation constraint to ensure that the resource **webfarm-stats** runs on the same node as an active copy of **webfarm-clone**.

```
# pcs constraint colocation add webfarm-stats with webfarm-clone
```

8.1.3. 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.

8.2. 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.

You can create a resource as 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 resource 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, see Section 5.1, "Resource Creation".

Table 8.2, "Properties of a Multi-State Resource" describes the options you can specify for a multi-state resource.

Table 8.2. Properties of a Multi-State Resource

Field	Description
id	Your name for the multi-state resource
priority, **target-role**, **is-managed**	See Table 5.3, "Resource Meta Options".
clone-max, **clone-node-max**, **notify**, **globally-unique**, **ordered**, **interleave**	See Table 8.1, "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.

8.2.1. Monitoring Multi-State Resources

To add a monitoring operation for the master resource only, you can add an additional monitor operation to the resource. Note, however, that every monitor operation on a resource must have a different interval.

The following example configures a monitor operation with an interval of 11 seconds on the master resource for **ms_resource**. This monitor operation is in addition to the default monitor operation with the default monitor interval of 10 seconds.

```
# pcs resource op add ms_resource interval=11s role=Master
```

8.2.2. 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 6.1, “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 colocation constraints, see Section 6.3, “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 6.2, “Order Constraints” .

8.2.3. 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.

8.3. CONFIGURING A VIRTUAL DOMAIN AS A RESOURCE

You can configure a virtual domain that is managed by the **libvirt** virtualization framework as a cluster resource with the **pcs resource create** command, specifying **VirtualDomain** as the resource type.

When configuring a virtual domain as a resource, take the following considerations into account:

- A virtual domain should be stopped before you configure it as a cluster resource.
- Once a virtual domain is a cluster resource, it should not be started, stopped, or migrated except through the cluster tools.
- Do not configure a virtual domain that you have configured as a cluster resource to start when its host boots.
- All nodes must have access to the necessary configuration files and storage devices for each managed virtual domain.

If you want the cluster to manage services within the virtual domain itself, you can configure the virtual domain as a guest node. For information on configuring guest nodes, see Section 8.4, “The pacemaker_remote Service”

For information on configuring virtual domains, see the Virtualization Deployment and Administration Guide.

Table 8.3, "Resource Options for Virtual Domain Resources" describes the resource options you can configure for a **VirtualDomain** resource.

Table 8.3. Resource Options for Virtual Domain Resources

Field	Default	Description
config		(required) Absolute path to the **libvirt** configuration file for this virtual domain.
hypervisor	System dependent	Hypervisor URI to connect to. You can determine the system's default URI by running the **virsh --quiet uri** command.
force_stop	**0**	Always forcefully shut down ("destroy") the domain on stop. The default behavior is to resort to a forceful shutdown only after a graceful shutdown attempt has failed. You should set this to **true** only if your virtual domain (or your virtualization back end) does not support graceful shutdown.
migration_transport	System dependent	Transport used to connect to the remote hypervisor while migrating. If this parameter is omitted, the resource will use **libvirt**'s default transport to connect to the remote hypervisor.
migration_network_suffix		Use a dedicated migration network. The migration URI is composed by adding this parameter's value to the end of the node name. If the node name is a fully qualified domain name (FQDN), insert the suffix immediately prior to the first period (.) in the FQDN. Ensure that this composed host name is locally resolvable and the associated IP address is reachable through the favored network.
monitor_scripts		To additionally monitor services within the virtual domain, add this parameter with a list of scripts to monitor. *Note*: When monitor scripts are used, the **start** and **migrate_from** operations will complete only when all monitor scripts have completed successfully. Be sure to set the timeout of these operations to accommodate this delay
autoset_utilization_cpu	**true**	If set to **true**, the agent will detect the number of **domainU**'s **vCPU**s from **virsh**, and put it into the CPU utilization of the resource when the monitor is executed.
autoset_utilization_hv_memory	**true**	If set it true, the agent will detect the number of **Max memory** from **virsh**, and put it into the **hv_memory** utilization of the source when the monitor is executed.
migrateport	random highport	This port will be used in the **qemu** migrate URI. If unset, the port will be a random highport.

Field	Default	Description
snapshot		Path to the snapshot directory where the virtual machine image will be stored. When this parameter is set, the virtual machine's RAM state will be saved to a file in the snapshot directory when stopped. If on start a state file is present for the domain, the domain will be restored to the same state it was in right before it stopped last. This option is incompatible with the **force_stop** option.

In addition to the **VirtualDomain** resource options, you can configure the **allow-migrate** metadata option to allow live migration of the resource to another node. When this option is set to **true**, the resource can be migrated without loss of state. When this option is set to **false**, which is the default state, the virtual domain will be shut down on the first node and then restarted on the second node when it is moved from one node to the other.

The following example configures a **VirtualDomain** resource named **VM**. Since the **allow-migrate** option is set to **true** a **pcs move VM *nodeX*** command would be done as a live migration.

```
# pcs resource create VM VirtualDomain config=.../vm.xml \
       migration_transport=ssh meta allow-migrate=true
```

8.4. THE PACEMAKER_REMOTE SERVICE

The **pacemaker_remote** service allows nodes not running **corosync** to integrate into the cluster and have the cluster manage their resources just as if they were real cluster nodes.

Among the capabilities that the **pacemaker_remote** service provides are the following:

- The **pacemaker_remote** service allows you to scale beyond the **corosync** 16-node limit.
- The **pacemaker_remote** service allows you to manage a virtual environment as a cluster resource and also to manage individual services within the virtual environment as cluster resources.

The following terms are used to describe the **pacemaker_remote** service.

- *cluster node*— A node running the High Availability services (**pacemaker** and **corosync**).
- *remote node*— A node running **pacemaker_remote** to remotely integrate into the cluster without requiring **corosync** cluster membership. A remote node is configured as a cluster resource that uses the **ocf:pacemaker:remote** resource agent.
- *guest node*— A virtual guest node running the **pacemaker_remote** service. A guest node is configured using the **remote-node** metadata option of a resource agent such as **ocf:pacemaker:VirtualDomain**. The virtual guest resource is managed by the cluster; it is both started by the cluster and integrated into the cluster as a remote node.
- *pacemaker_remote*— A service daemon capable of performing remote application management within remote nodes and guest nodes (KVM and LXC) in a Pacemaker cluster environment. This service is an enhanced version of Pacemaker's local resource management daemon (LRMD) that is capable of managing resources remotely on a node not running corosync.

- *LXC* — A Linux Container defined by the **libvirt-lxc** Linux container driver.

A Pacemaker cluster running the **pacemaker_remote** service has the following characteristics.

- The remote nodes and/or the guest nodes run the **pacemaker_remote** service (with very little configuration required on the virtual machine side).
- The cluster stack (**pacemaker** and **corosync**), running on the cluster nodes, connects to the **pacemaker_remote** service on the remote nodes, allowing them to integrate into the cluster.
- The cluster stack (**pacemaker** and **corosync**), running on the cluster nodes, launches the guest nodes and immediately connects to the **pacemaker_remote** service on the guest nodes, allowing them to integrate into the cluster.

The key difference between the cluster nodes and the remote and guest nodes that the cluster nodes manage is that the remote and guest nodes are not running the cluster stack. This means the remote and guest nodes have the following limitations:

- they do not take place in quorum
- they do not execute fencing device actions
- they are not eligible to be be the cluster's Designated Controller (DC)
- they do not themselves run the full range of **pcs** commands

On the other hand, remote nodes and guest nodes are not bound to the scalability limits associated with the cluster stack.

Other than these noted limitations, the remote nodes behave just like cluster nodes in respect to resource management, and the remote and guest nodes can themselves be fenced. The cluster is fully capable of managing and monitoring resources on each remote and guest node: You can build constraints against them, put them in standby, or perform any other action you perform on cluster nodes with the **pcs** commands. Remote and guest nodes appear in cluster status output just as cluster nodes do.

8.4.1. Host and Guest Authentication

The connection between cluster nodes and pacemaker_remote is secured using Transport Layer Security (TLS) with pre-shared key (PSK) encryption and authentication over TCP (using port 3121 by default). This means both the cluster node and the node running **pacemaker_remote** must share the same private key. By default this key must be placed at **/etc/pacemaker/authkey** on both cluster nodes and remote nodes.

8.4.2. Guest Node Resource Options

When configuring a virtual machine or LXC resource to act as a guest node, you create a **VirtualDomain** resource, which manages the virtual machine. For descriptions of the options you can set for a **VirtualDomain** resource, use the following command.

```
# pcs resource describe VirtualDomain
```

In addition to the **VirtualDomain** resource options, you can configure metadata options to both enable the resource as a guest node and define the connection parameters. Table 8.4, “Metadata Options for Configuring KVM/LXC Resources as Remote Nodes” describes these metadata options.

Table 8.4. Metadata Options for Configuring KVM/LXC Resources as Remote Nodes

Field	Default	Description
remote-node	<none>	The name of the guest node this resource defines. This both enables the resource as a guest node and defines the unique name used to identify the guest node. *WARNING*: This value cannot overlap with any resource or node IDs.
remote-port	3121	Configures a custom port to use for the guest connection to **pacemaker_remote**.
remote-addr	**remote-node** value used as host name	The IP address or host name to connect to if remote node's name is not the host name of the guest
remote-connect-timeout	60s	Amount of time before a pending guest connection will time out

8.4.3. Remote Node Resource Options

You configure a remote node as a cluster resource with the **pcs resource create** command, specifying **ocf:pacemaker:remote** as the resource type. Table 8.5, "Resource Options for Remote Nodes" describes the resource options you can configure for a **remote** resource.

Table 8.5. Resource Options for Remote Nodes

Field	Default	Description
reconnect_interval	0	Time in seconds to wait before attempting to reconnect to a remote node after an active connection to the remote node has been severed. This wait is recurring. If reconnect fails after the wait period, a new reconnect attempt will be made after observing the wait time. When this option is in use, Pacemaker will keep attempting to reach out and connect to the remote node indefinitely after each wait interval.
server		Server location to connect to. This can be an IP address or host name.
port		TCP port to connect to.

8.4.4. Changing Default pacemaker_remote Options

If you need to change the default port or **authkey** location for either Pacemaker or **pacemaker_remote**, there are environment variables you can set that affect both of those daemons. These environment variables can be enabled by placing them in the **/etc/sysconfig/pacemaker** file as follows.

```
#==#==# Pacemaker Remote
# Use a custom directory for finding the authkey.
PCMK_authkey_location=/etc/pacemaker/authkey
```

```
#
# Specify a custom port for Pacemaker Remote connections
PCMK_remote_port=3121
```

Note that when you change the default key location on a particular node (cluster node, guest node or remote node), it is sufficient to set PCMK_authkey_location on that node (and put the key in that location). It is not necessary that the location be the same on every node, although doing so makes administration easier.

When changing the default port used by a particular guest node or remote node, the **PCMK_remote_port** variable must be set in that node's **/etc/sysconfig/pacemaker** file, and the cluster resource creating the guest node or remote node connection must also be configured with the same port number (using the **remote-port** metadata option for guest nodes, or the **port** option for remote nodes).

8.4.5. Configuration Overview: KVM Guest Node

This section provides a high-level summary overview of the steps to perform to have Pacemaker launch a virtual machine and to integrate that machine as a guest node, using **libvirt** and KVM virtual guests.

1. After installing the virtualization software and enabling the **libvirtd** service on the cluster nodes, put the same encryption key with the path **/etc/pacemaker/authkey** on every cluster node and virtual machine. This secures remote communication and authentication.

   Run the following set of commands on every node to create the **authkey** directory with secure permissions.

   ```
   # mkdir -p --mode=0750 /etc/pacemaker
   # chgrp haclient /etc/pacemaker
   ```

   The following command shows one method to create an encryption key. You should create the key only once and then copy it to all of the nodes.

   ```
   # dd if=/dev/urandom of=/etc/pacemaker/authkey bs=4096 count=1
   ```

2. On every virtual machine, install **pacemaker_remote** packages, start the **pacemaker_remote** service and enable it to run on startup, and allow TCP port 3121 through the firewall.

   ```
   # yum install pacemaker-remote resource-agents
   # systemctl start pacemaker_remote.service
   # systemctl enable pacemaker_remote.service
   # firewall-cmd --add-port 3121/tcp --permanent
   # firewall-cmd --reload
   ```

3. Give each virtual machine a static network address and unique host name, which should be known to all nodes. For information on setting a static IP address for the guest virtual machine, see the *Virtualization Deployment and Administration Guide*

4. To create the **VirtualDomain** resource agent for the management of the virtual machine, Pacemaker requires the virtual machine's xml config file to be dumped to a file on disk. For example, if you created a virtual machine named **guest1**, dump the xml to a file somewhere on

the host. You can use a file name of your choosing; this example uses **/etc/pacemaker/guest1.xml**.

```
# virsh dumpxml guest1 > /etc/pacemaker/guest1.xml
```

5. If it is running, shut down the guest node. Pacemaker will start the node when it is configured in the cluster.

6. Create the **VirtualDomain** resource, configuring the **remote-note** resource meta option to indicate that the virtual machine is a guest node capable of running resources.

   In the example below, the meta-attribute **remote-node=guest1** tells pacemaker that this resource is a guest node with the host name **guest1** that is capable of being integrated into the cluster. The cluster will attempt to contact the virtual machine's **pacemaker_remote** service at the host name **guest1** after it launches.

   From a cluster node, enter the following command.

```
# pcs resource create vm-guest1 VirtualDomain
hypervisor="qemu:///system" config="/virtual_machines/vm-guest1.xml"
meta remote-node=guest1
```

7. After creating the **VirtualDomain** resource, you can treat the guest node just as you would treat any other node in the cluster. For example, you can create a resource and place a resource constraint on the resource to run on the guest node as in the following commands, which are run from a cluster node. As of Red Hat Enterprise Linux 6.8, you can include guest nodes in groups, which allows you to group a storage device, file system, and VM.

```
# pcs resource create webserver apache
configfile=/etc/httpd/conf/httpd.conf op monitor interval=30s
# pcs constraint location webserver prefers guest1
```

8.4.6. Configuration Overview: Remote Node

This section provides a high-level summary overview of the steps to perform to configure a Pacemaker remote node and to integrate that node into an existing Pacemaker cluster environment.

1. On the node that you will be configuring as a remote node, allow cluster-related services through the local firewall.

```
# firewall-cmd --permanent --add-service=high-availability
success
# firewall-cmd --reload
success
```

NOTE

If you are using **iptables** directly, or some other firewall solution besides **firewalld**, simply open the following ports, which can be used by various clustering components: TCP ports 2224, 3121, and 21064, and UDP port 5405.

2. Install the **pacemaker_remote** daemon on the remote node.

```
# yum install -y pacemaker-remote resource-agents pcs
```

3. All nodes (both cluster nodes and remote nodes) must have the same authentication key installed for the communication to work correctly. If you already have a key on an existing node, use that key and copy it to the remote node. Otherwise, create a new key on the remote node.

   Run the following set of commands on the remote node to create a directory for the authentication key with secure permissions.

   ```
   # mkdir -p --mode=0750 /etc/pacemaker
   # chgrp haclient /etc/pacemaker
   ```

   The following command shows one method to create an encryption key on the remote node.

   ```
   # dd if=/dev/urandom of=/etc/pacemaker/authkey bs=4096 count=1
   ```

4. Start and enable the **pacemaker_remote** daemon on the remote node.

   ```
   # systemctl enable pacemaker_remote.service
   # systemctl start pacemaker_remote.service
   ```

5. On the cluster node, create a location for the shared authentication key with the same path as the authentication key on the remote node and copy the key into that directory. In this example, the key is copied from the remote node where the key was created.

   ```
   # mkdir -p --mode=0750 /etc/pacemaker
   # chgrp haclient /etc/pacemaker
   # scp remote1:/etc/pacemaker/authkey /etc/pacemaker/authkey
   ```

6. Run the following command from a cluster node to create a **remote** resource. In this case the remote node is **remote1**.

   ```
   # pcs resource create remote1 ocf:pacemaker:remote
   ```

7. After creating the **remote** resource, you can treat the remote node just as you would treat any other node in the cluster. For example, you can create a resource and place a resource constraint on the resource to run on the remote node as in the following commands, which are run from a cluster node.

   ```
   # pcs resource create webserver apache
   configfile=/etc/httpd/conf/httpd.conf op monitor interval=30s
   # pcs constraint location webserver prefers remote1
   ```

WARNING

Never involve a remote node connection resource in a resource group, colocation constraint, or order constraint.

8. Configure fencing resources for the remote node. Remote nodes are fenced the same way as cluster nodes. Configure fencing resources for use with remote nodes the same as you would with cluster nodes. Note, however, that remote nodes can never initiate a fencing action. Only cluster nodes are capable of actually executing a fencing operation against another node.

8.4.7. System Upgrades and pacemaker_remote

As of Red Hat Enterprise Linux 6.8, if the **pacemaker_remote** service is stopped on an active Pacemaker Remote node, the cluster will gracefully migrate resources off the node before stopping the node. This allows you to perform software upgrades and other routine maintenance procedures without removing the node from the cluster. Once **pacemaker_remote** is shut down, however, the cluster will immediately try to reconnect. If **pacemaker_remote** is not restarted within the resource's monitor timeout, the cluster will consider the monitor operation as failed.

If you wish to avoid monitor failures when the **pacemaker_remote** service is stopped on an active Pacemaker Remote node, you can use the following procedure to take the node out of the cluster before performing any system administration that might stop **pacemaker_remote**

WARNING

For Red Hat Enterprise Linux release 6.7 and earlier, if **pacemaker_remote** stops on a node that is currently integrated into a cluster, the cluster will fence that node. If the stop happens automatically as part of a **yum update** process, the system could be left in an unusable state (particularly if the kernel is also being upgraded at the same time as **pacemaker_remote**). For Red Hat Enterprise Linux release 6.7 and earlier you must use the following procedure to take the node out of the cluster before performing any system administration that might stop **pacemaker_remote**.

Use the following procedure to take a node out of a cluster when performing maintenance on a node running **pacemaker_remote**:

1. Stop the node's connection resource with the **pcs resource disable** ***resourcename***, which will move all services off the node. For guest nodes, this will also stop the VM, so the VM must be started outside the cluster (for example, using **virsh**) to perform any maintenance.

2. Perform the desired maintenance.

3. When ready to return the node to the cluster, re-enable the resource with the **pcs resource enable**.

8.4.8. Converting a VM Resource to a Guest Node

Use the following command to convert an existing **VirtualDomain** resource into a guest node. You do not need to run this command if the resource was originally created as a guest node.

```
pcs cluster remote-node add hostname resource_id [options]
```

Use the following command to disable a resource configured as a guest node on the specified host.

```
pcs cluster remote-node remove hostname
```

CHAPTER 9. 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 9.1. 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"` cannot 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`. The default value is `and`.

9.1. NODE ATTRIBUTE EXPRESSIONS

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

Table 9.2. 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`

Field	Description
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

9.2. 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 9.3. 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

9.3. 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 9.4. 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 week and year)
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).

9.4. 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 (ie. you can have a duration of 19 months). Like **date_specs**, any field not supplied is ignored.

9.5. 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. 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
```

9.6. SAMPLE TIME BASED EXPRESSIONS

The following command configures an expression that is true if now is any time in the year 2005.

```
# pcs constraint location Webserver rule score=INFINITY date-spec
```

```
years=2005
```

The following command configures an expression that is true from 9am to 5pm, Monday through Friday. Note that the hours value of 16 matches up to 16:59:59, as the numeric value (hour) still matches.

```
# pcs constraint location Webserver rule score=INFINITY date-spec
hours="9-16" weekdays="1-5"
```

The following command configures an expression that is true when there is a full moon on Friday the 13th.

```
# pcs constraint location Webserver rule date-spec weekdays=5 monthdays=13
moon=4
```

9.7. 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 10. PACEMAKER CLUSTER PROPERTIES

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

- Table 10.1, "Cluster Properties" describes the cluster properties options.
- Section 10.2, "Setting and Removing Cluster Properties" describes how to set cluster properties.
- Section 10.3, "Querying Cluster Property Settings" describes how to list the currently set cluster properties.

10.1. SUMMARY OF CLUSTER PROPERTIES AND OPTIONS

Table 10.1, "Cluster Properties" summaries the Pacemaker cluster properties, showing the default values of the properties and the possible values you can set for those properties.

NOTE

In addition to the properties described in this table, there are additional cluster properties that are exposed by the cluster software. For these properties, it is recommended that you not change their values from their defaults.

Table 10.1. Cluster Properties

Option	Default	Description
`batch-limit`	30	The number of jobs that the transition engine (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.

Option	Default	Description
stonith-enabled	true	Indicates that failed nodes and nodes with resources that cannot 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 canceled.
start-failure-is-fatal	true	Indicates whether a failure to start a resource on a particular node prevents further start attempts on that node. When set to **false**, the cluster will decide whether to try starting on the same node again based on the resource's current failure count and migration threshold. For information on setting the **migration-threshold** option for a resource, see Section 7.2, “Moving Resources Due to Failure”.
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.
cluster-infrastructure		The messaging stack on which Pacemaker is currently running. Used for informational and diagnostic purposes; not user-configurable.
dc-version		Version of Pacemaker on the cluster's Designated Controller (DC). Used for diagnostic purposes; not user-configurable.
last-lrm-refresh		Last refresh of the Local Resource Manager, given in units of seconds since epoca. Used for diagnostic purposes; not user-configurable.

Option	Default	Description
cluster-recheck-interval	60	Polling interval for time-based changes to options, resource parameters and constraints. Allowed values: Zero disables polling, positive values are an interval in seconds (unless other SI units are specified, such as 5min).
maintenance-mode	false	Maintenance Mode tells the cluster to go to a "hands off" mode, and not start or stop any services until told otherwise. When maintenance mode is completed, the cluster does a sanity check of the current state of any services, and then stops or starts any that need it.
shutdown-escalation	20min	The time after which to give up trying to shut down gracefully and just exit. Advanced use only.
stonith-timeout	60s	How long to wait for a STONITH action to complete.
stop-all-resources	false	Should the cluster stop all resources.
enable-acl	false	(Red Hat Enterprise Linux 6.6 and later) Indicates whether the cluster can use access control lists, as set with the **pcs acl** command.

10.2. 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 symmetic-cluster=
```

10.3. 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 **cluster-infrastructure** property, execute the following command:

```
# pcs property show cluster-infrastructure
Cluster Properties:
 cluster-infrastructure: cman
```

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 11. TRIGGERING SCRIPTS FOR CLUSTER EVENTS

A Pacemaker cluster is an event-driven system, where an event might be a resource or node failure, a configuration change, or a resource starting or stopping. You can configure Pacemaker cluster alerts to take some external action when a cluster event occurs. You can configure cluster alerts in one of two ways:

- As of Red Hat Enterprise Linux 6.9, you can configure Pacemaker alerts by means of alert agents, which are external programs that the cluster calls in the same manner as the cluster calls resource agents to handle resource configuration and operation. This is the preferred, simpler method of configuring cluster alerts. Pacemaker alert agents are described in Section 11.1, "Pacemaker Alert Agents (Red Hat Enterprise Linux 6.9 and later)" .
- The **ocf:pacemaker:ClusterMon** resource can monitor the cluster status and trigger alerts on each cluster event. This resource runs the **crm_mon** command in the background at regular intervals. For information on the **ClusterMon** resource see Section 11.2, "Event Notification with Monitoring Resources".

11.1. PACEMAKER ALERT AGENTS (RED HAT ENTERPRISE LINUX 6.9 AND LATER)

You can create Pacemaker alert agents to take some external action when a cluster event occurs. The cluster passes information about the event to the agent by means of environment variables. Agents can do anything desired with this information, such as send an email message or log to a file or update a monitoring system.

- Pacemaker provides several sample alert agents, which are installed in **/usr/share/pacemaker/alerts** by default. These sample scripts may be copied and used as is, or they may be used as templates to be edited to suit your purposes. Refer to the source code of the sample agents for the full set of attributes they support. See Section 11.1.1, "Using the Sample Alert Agents" for an example of a basic procedure for configuring an alert that uses a sample alert agent.
- General information on configuring and administering alert agents is provided in Section 11.1.2, "Alert Creation", Section 11.1.3, "Displaying, Modifying, and Removing Alerts" , Section 11.1.4, "Alert Recipients", Section 11.1.5, "Alert Meta Options" , and Section 11.1.6, "Alert Configuration Command Examples".
- You can write your own alert agents for a Pacemaker alert to call. For information on writing alert agents, see Section 11.1.7, "Writing an Alert Agent" .

11.1.1. Using the Sample Alert Agents

When you use one of sample alert agents, you should review the script to ensure that it suits your needs. These sample agents are provided as a starting point for custom scripts for specific cluster environments.

To use one of the sample alert agents, you must install the agent on each node in the cluster. For example, the following command installs the **alert_snmp.sh.sample** script as **alert_snmp.sh**.

```
# install --mode=0755 /usr/share/pacemaker/alerts/alert_snmp.sh.sample
/var/lib/pacemaker/alert_snmp.sh
```

After you have installed the script, you can create an alert that uses the script. The following example

configures an alert that uses the installed **alert_snmp.sh** alert agent to send cluster events as SNMP traps. By default, the script will send all events except successful monitor calls to the SNMP server. This example configures the timestamp format as a meta option. For information about meta options, see Section 11.1.5, "Alert Meta Options" . After configuring the alert, this example configures a recipient for the alert and displays the alert configuration.

```
# pcs alert create id=snmp_alert path=/var/lib/pacemaker/alert_snmp.sh
meta timestamp-format="%Y-%m-%d,%H:%M:%S.%01N".
# pcs alert recipient add snmp_alert 192.168.1.2
# pcs alert
Alerts:
 Alert: snmp_alert (path=/var/lib/pacemaker/alert_snmp.sh)
  Meta options: timestamp-format=%Y-%m-%d,%H:%M:%S.%01N.
  Recipients:
   Recipient: snmp_alert-recipient (value=192.168.1.2)
```

The following example installs the **alert_smtp.sh** agent and then configures an alert that uses the installed alert agent to send cluster events as email messages. After configuring the alert, this example configures a recipient and displays the alert configuration.

```
# install --mode=0755 /usr/share/pacemaker/alerts/alert_smtp.sh.sample
/var/lib/pacemaker/alert_smtp.sh
# pcs alert create id=smtp_alert path=/var/lib/pacemaker/alert_smtp.sh
options email_sender=donotreply@example.com
# pcs alert recipient add smtp_alert admin@example.com
# pcs alert
Alerts:
 Alert: smtp_alert (path=/var/lib/pacemaker/alert_smtp.sh)
  Options: email_sender=donotreply@example.com
  Recipients:
   Recipient: smtp_alert-recipient (value=admin@example.com)
```

For more information on the format of the **pcs alert create** and **pcs alert recipient add** commands, see Section 11.1.2, "Alert Creation" and Section 11.1.4, "Alert Recipients".

11.1.2. Alert Creation

The following command creates a cluster alert. The options that you configure are agent-specific configuration values that are passed to the alert agent script at the path you specify as additional environment variables. If you do not specify a value for **id**, one will be generated. For information on alert meta options, Section 11.1.5, "Alert Meta Options" .

```
pcs alert create path=path [id=alert-id] [description=description]
[options [option=value]...] [meta [meta-option=value]...]
```

Multiple alert agents may be configured; the cluster will call all of them for each event. Alert agents will be called only on cluster nodes. They will be called for events involving Pacemaker Remote nodes, but they will never be called on those nodes.

The following example creates a simple alert that will call **my-script.sh** for each event.

```
# pcs alert create id=my_alert path=/path/to/myscript.sh
```

For an example that shows how to create a cluster alert that uses one of the sample alert agents, see Section 11.1.1, "Using the Sample Alert Agents".

11.1.3. Displaying, Modifying, and Removing Alerts

The following command shows all configured alerts along with the values of the configured options.

```
pcs alert [config|show]
```

The following command updates an existing alert with the specified *alert-id* value.

```
pcs alert update alert-id [path=path] [description=description] [options
[option=value]...] [meta [meta-option=value]...]
```

The following command removes an alert with the specified *alert-id* value.

```
pcs alert remove alert-id
```

11.1.4. Alert Recipients

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

The recipient may be anything the alert agent can recognize: an IP address, an email address, a file name, or whatever the particular agent supports.

The following command adds a new recipient to the specified alert.

```
pcs alert recipient add alert-id recipient-value [id=recipient-id]
[description=description] [options [option=value]...] [meta [meta-
option=value]...]
```

The following command updates an existing alert recipient.

```
pcs alert recipient update recipient-id [value=recipient-value]
[description=description] [options [option=value]...] [meta [meta-
option=value]...]
```

The following command removes the specified alert recipient.

```
pcs alert recipient remove recipient-id
```

The following example command adds the alert recipient **my-alert-recipient** with a recipient ID of **my-recipient-id** to the alert **my-alert**. This will configure the cluster to call the alert script that has been configured for **my-alert** for each event, passing the recipient **some-address** as an environment variable.

```
#  pcs alert recipient add my-alert my-alert-recipient id=my-recipient-id
options value=some-address
```

11.1.5. Alert Meta Options

As with resource agents, meta options can be configured for alert agents to affect how Pacemaker calls them. Table 11.1, "Alert Meta Options" describes the alert meta options. Meta options can be configured per alert agent as well as per recipient.

Table 11.1. Alert Meta Options

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

The following example configures an alert that calls the script **my-script.sh** and then adds two recipients to the alert. The first recipient has an ID of **my-alert-recipient1** and the second recipient has an ID of **my-alert-recipient2**. The script will get called twice for each event, with each call using a 15-second timeout. One call will be passed to the recipient **someuser@example.com** with a timestamp in the format %D %H:%M, while the other call will be passed to the recipient **otheruser@example.com** with a timestamp in the format %c. `

```
# pcs alert create id=my-alert path=/path/to/my-script.sh meta timeout=15s
# pcs alert recipient add my-alert someuser@example.com id=my-alert-
recipient1 meta timestamp-format=%D %H:%M
# pcs alert recipient add my-alert otheruser@example.com id=my-alert-
recipient2 meta timestamp-format=%c
```

11.1.6. Alert Configuration Command Examples

The following sequential examples show some basic alert configuration commands to show the format to use to create alerts, add recipients, and display the configured alerts.

The following commands create a simple alert, add two recipients to the alert, and display the configured values.

- Since no alert ID value is specified, the system creates an alert ID value of **alert**.
- The first recipient creation command specifies a recipient of **rec_value**. Since this command does not specify a recipient ID, the value of **alert-recipient** is used as the recipient ID.
- The second recipient creation command specifies a recipient of **rec_value2**. This command specifies a recipient ID of **my-recipient** for the recipient.

```
# pcs alert create path=/my/path
# pcs alert recipient add alert rec_value
# pcs alert recipient add alert rec_value2 id=my-recipient
# pcs alert config
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
   Recipient: my-recipient (value=rec_value2)
```

This following commands add a second alert and a recipient for that alert. The alert ID for the second alert is **my-alert** and the recipient value is **my-other-recipient**. Since no recipient ID is specified, the system provides a recipient id of **my-alert-recipient**.

```
# pcs alert create id=my-alert path=/path/to/script
description=alert_description options option1=value1 opt=val meta meta-
option1=2 m=val
# pcs alert recipient add my-alert my-other-recipient
# pcs alert
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
   Recipient: my-recipient (value=rec_value2)
 Alert: my-alert (path=/path/to/script)
  Description: alert_description
  Options: opt=val option1=value1
  Meta options: m=val meta-option1=2
  Recipients:
   Recipient: my-alert-recipient (value=my-other-recipient)
```

The following commands modify the alert values for the alert **my-alert** and for the recipient **my-alert-recipient**.

```
# pcs alert update my-alert options option1=newvalue1 meta m=newval
# pcs alert recipient update my-alert-recipient options option1=new meta
metaopt1=newopt
# pcs alert
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
   Recipient: my-recipient (value=rec_value2)
 Alert: my-alert (path=/path/to/script)
  Description: alert_description
  Options: opt=val option1=newvalue1
  Meta options: m=newval meta-option1=2
  Recipients:
   Recipient: my-alert-recipient (value=my-other-recipient)
    Options: option1=new
    Meta options: metaopt1=newopt
```

The following command removes the recipient **my-alert-recipient** from **alert**.

```
# pcs alert recipient remove my-recipient
# pcs alert
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
 Alert: my-alert (path=/path/to/script)
  Description: alert_description
  Options: opt=val option1=newvalue1
  Meta options: m=newval meta-option1=2
  Recipients:
```

```
    Recipient: my-alert-recipient (value=my-other-recipient)
     Options: option1=new
     Meta options: metaopt1=newopt
```

The following command removes **myalert** from the configuration.

```
# pcs alert remove my-alert
# pcs alert
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
```

11.1.7. Writing an Alert Agent

There are three types of Pacemaker alerts: node alerts, fencing alerts, and resource alerts. The environment variables that are passed to the alert agents can differ, depending on the type of alert. Table 11.2, "Environment Variables Passed to Alert Agents" describes the environment variables that are passed to alert agents and specifies when the environment variable is associated with a specific alert type.

Table 11.2. Environment Variables Passed to Alert Agents

Environment Variable	Description
CRM_alert_kind	The type of alert (node, fencing, or resource)
CRM_alert_version	The version of Pacemaker sending the alert
CRM_alert_recipient	The configured recipient
CRM_alert_node_sequence	A sequence number increased whenever an alert is being issued on the local node, which can be used to reference the order in which alerts have been issued by Pacemaker. An alert for an event that happened later in time reliably has a higher sequence number than alerts for earlier events. Be aware that this number has no cluster-wide meaning.
CRM_alert_timestamp	A timestamp created prior to executing the agent, in the format specified by the **timestamp-format** meta option. This allows the agent to have a reliable, high-precision time of when the event occurred, regardless of when the agent itself was invoked (which could potentially be delayed due to system load or other circumstances).
CRM_alert_node	Name of affected node
CRM_alert_desc	Detail about event. For node alerts, this is the node's current state (member or lost). For fencing alerts, this is a summary of the requested fencing operation, including origin, target, and fencing operation error code, if any. For resource alerts, this is a readable string equivalent of **CRM_alert_status**.
CRM_alert_nodeid	ID of node whose status changed (provided with node alerts only)

Environment Variable	Description
CRM_alert_task	The requested fencing or resource operation (provided with fencing and resource alerts only)
CRM_alert_rc	The numerical return code of the fencing or resource operation (provided with fencing and resource alerts only)
CRM_alert_rsc	The name of the affected resource (resource alerts only)
CRM_alert_interval	The interval of the resource operation (resource alerts only)
CRM_alert_target_rc	The expected numerical return code of the operation (resource alerts only)
CRM_alert_status	A numerical code used by Pacemaker to represent the operation result (resource alerts only)

When writing an alert agent, you must take the following concerns into account.

- Alert agents may be called with no recipient (if none is configured), so the agent must be able to handle this situation, even if it only exits in that case. Users may modify the configuration in stages, and add a recipient later.
- If more than one recipient is configured for an alert, the alert agent will be called once per recipient. If an agent is not able to run concurrently, it should be configured with only a single recipient. The agent is free, however, to interpret the recipient as a list.
- When a cluster event occurs, all alerts are fired off at the same time as separate processes. Depending on how many alerts and recipients are configured and on what is done within the alert agents, a significant load burst may occur. The agent could be written to take this into consideration, for example by queuing resource-intensive actions into some other instance, instead of directly executing them.
- Alert agents are run as the **hacluster** user, which has a minimal set of permissions. If an agent requires additional privileges, it is recommended to configure **sudo** to allow the agent to run the necessary commands as another user with the appropriate privileges.
- Take care to validate and sanitize user-configured parameters, such as **CRM_alert_timestamp** (whose content is specified by the user-configured **timestamp-format**), **CRM_alert_recipient**, and all alert options. This is necessary to protect against configuration errors. In addition, if some user can modify the CIB without having **hacluster**-level access to the cluster nodes, this is a potential security concern as well, and you should avoid the possibility of code injection.
- If a cluster contains resources for which the **onfail** parameter is set to **fence**, there will be multiple fence notifications on failure, one for each resource for which this parameter is set plus one additional notification. Both the STONITH daemon and the **crmd** daemon will send notifications. Pacemaker performs only one actual fence operation in this case, however, no matter how many notifications are sent.

NOTE

The alerts interface is designed to be backward compatible with the external scripts interface used by the **ocf:pacemaker:ClusterMon** resource. To preserve this compatibility, the environment variables passed to alert agents are available prepended with **CRM_notify_** as well as **CRM_alert_**. One break in compatibility is that the **ClusterMon** resource ran external scripts as the root user, while alert agents are run as the **hacluster** user. For information on configuring scripts that are triggered by the **ClusterMon**, see Section 11.2, “Event Notification with Monitoring Resources” .

11.2. EVENT NOTIFICATION WITH MONITORING RESOURCES

The **ocf:pacemaker:ClusterMon** resource can monitor the cluster status and trigger alerts on each cluster event. This resource runs the **crm_mon** command in the background at regular intervals.

By default, the **crm_mon** command listens for resource events only; to enable listing for fencing events you can provide the **--watch-fencing** option to the command when you configure the **ClusterMon** resource. The **crm_mon** command does not monitor for membership issues but will print a message when fencing is started and when monitoring is started for that node, which would imply that a member just joined the cluster.

The **ClusterMon** resource can execute an external program to determine what to do with cluster notifications by means of the **extra_options** parameter. Table 11.3, “Environment Variables Passed to the External Monitor Program” lists the environment variables that are passed to that program, which describe the type of cluster event that occurred.

Table 11.3. Environment Variables Passed to the External Monitor Program

Environment Variable	Description
CRM_notify_recipient	The static external-recipient from the resource definition
CRM_notify_node	The node on which the status change happened
CRM_notify_rsc	The name of the resource that changed the status
CRM_notify_task	The operation that caused the status change
CRM_notify_desc	The textual output relevant error code of the operation (if any) that caused the status change
CRM_notify_rc	The return code of the operation
CRM_target_rc	The expected return code of the operation
CRM_notify_status	The numerical representation of the status of the operation

The following example configures a **ClusterMon** resource that executes the external program **crm_logger.sh** which will log the event notifications specified in the program.

The following procedure creates the **crm_logger.sh** program that this resource will use.

1. On one node of the cluster, create the program that will log the event notifications.

```
# cat <<-END >/usr/local/bin/crm_logger.sh
#!/bin/sh
logger -t "ClusterMon-External" "${CRM_notify_node}
${CRM_notify_rsc} \
${CRM_notify_task} ${CRM_notify_desc} ${CRM_notify_rc} \
${CRM_notify_target_rc} ${CRM_notify_status}
${CRM_notify_recipient}";
exit;
END
```

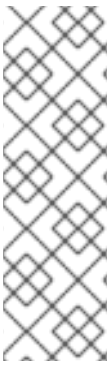
2. Set the ownership and permissions for the program.

```
# chmod 700 /usr/local/bin/crm_logger.sh
# chown root.root /usr/local/bin/crm_logger.sh
```

3. Use the **scp** command to copy the **crm_logger.sh** program to the other nodes of the cluster, putting the program in the same location on those nodes and setting the same ownership and permissions for the program.

The following example configures the **ClusterMon** resource, named **ClusterMon-External**, that runs the program **/usr/local/bin/crm_logger.sh**. The **ClusterMon** resource outputs the cluster status to an **html** file, which is **/var/www/html/cluster_mon.html** in this example. The **pidfile** detects whether **ClusterMon** is already running; in this example that file is **/var/run/crm_mon-external.pid**. This resource is created as a clone so that it will run on every node in the cluster. The **watch-fencing** is specified to enable monitoring of fencing events in addition to resource events, including the start/stop/monitor, start/monitor. and stop of the fencing resource.

```
# pcs resource create ClusterMon-External ClusterMon user=root \
update=10 extra_options="-E /usr/local/bin/crm_logger.sh --watch-fencing"
\
htmlfile=/var/www/html/cluster_mon.html \
pidfile=/var/run/crm_mon-external.pid clone
```



NOTE

The **crm_mon** command that this resource executes and which could be run manually is as follows:

```
# /usr/sbin/crm_mon -p /var/run/crm_mon-manual.pid -d -i 5 \
-h /var/www/html/crm_mon-manual.html -E
"/usr/local/bin/crm_logger.sh" \
--watch-fencing
```

The following example shows the format of the output of the monitoring notifications that this example yields.

```
Aug  7 11:31:32 rh6node1pcmk ClusterMon-External:
```

```
rh6node2pcmk.examplerh.com ClusterIP st_notify_fence Operation
st_notify_fence requested by rh6node1pcmk.examplerh.com for peer
rh6node2pcmk.examplerh.com: OK (ref=b206b618-e532-42a5-92eb-44d363ac848e)
0 0 0 #177
Aug  7 11:31:32 rh6node1pcmk ClusterMon-External:
rh6node1pcmk.examplerh.com ClusterIP start OK 0 0 0
Aug  7 11:31:32 rh6node1pcmk ClusterMon-External:
rh6node1pcmk.examplerh.com ClusterIP monitor OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External:
rh6node1pcmk.examplerh.com fence_xvms monitor OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External:
rh6node1pcmk.examplerh.com ClusterIP monitor OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External:
rh6node1pcmk.examplerh.com ClusterMon-External start OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External:
rh6node1pcmk.examplerh.com fence_xvms start OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External:
rh6node1pcmk.examplerh.com ClusterIP start OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External:
rh6node1pcmk.examplerh.com ClusterMon-External monitor OK 0 0 0
Aug  7 11:34:00 rh6node1pcmk crmd[2887]:   notice: te_rsc_command:
Initiating action 8: monitor ClusterMon-External:1_monitor_0 on
rh6node2pcmk.examplerh.com
Aug  7 11:34:00 rh6node1pcmk crmd[2887]:   notice: te_rsc_command:
Initiating action 16: start ClusterMon-External:1_start_0 on
rh6node2pcmk.examplerh.com
Aug  7 11:34:00 rh6node1pcmk ClusterMon-External:
rh6node1pcmk.examplerh.com ClusterIP stop OK 0 0 0
Aug  7 11:34:00 rh6node1pcmk crmd[2887]:   notice: te_rsc_command:
Initiating action 15: monitor ClusterMon-External_monitor_10000 on
rh6node2pcmk.examplerh.com
Aug  7 11:34:00 rh6node1pcmk ClusterMon-External:
rh6node2pcmk.examplerh.com ClusterMon-External start OK 0 0 0
Aug  7 11:34:00 rh6node1pcmk ClusterMon-External:
rh6node2pcmk.examplerh.com ClusterMon-External monitor OK 0 0 0
Aug  7 11:34:00 rh6node1pcmk ClusterMon-External:
rh6node2pcmk.examplerh.com ClusterIP start OK 0 0 0
Aug  7 11:34:00 rh6node1pcmk ClusterMon-External:
rh6node2pcmk.examplerh.com ClusterIP monitor OK 0 0 0
```

APPENDIX A. CLUSTER CREATION IN RED HAT ENTERPRISE LINUX RELEASE 6.5 AND RED HAT ENTERPRISE LINUX RELEASE 6.6 (AND LATER)

Configuring a Red Hat High Availability Cluster in Red Hat Enterprise Linux 6.6 and later with Pacemaker requires a different set of configuration tools with a different administrative interface than configuring a cluster in Red Hat Enterprise Linux 6 with **rgmanager**. Section A.1, "Cluster Creation with rgmanager and with Pacemaker" summarizes the configuration differences between the various cluster components.

The Red Hat Enterprise Linux 6.6 release provides some new features for cluster configuration with Pacemaker. Section A.2, "Cluster Creation with Pacemaker in Red Hat Enterprise Linux Release 6.5 and Red Hat Enterprise Linux Release 6.6 (and later)" summarizes some small configuration differences between **pcs** support in Red Hat Enterprise Linux release 6.5 and **pcs** support in Red Hat Enterprise Linux release 6.6 and later.

A.1. 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 in Red Hat Enterprise Linux release 6.6 and later.

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 the **cib.xml** file 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**.

Configuration Component	rgmanager	Pacemaker
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.	Start and enable cluster services with the following procedure: 1. On every node, execute **service pcsd start**, then **service pcsd enable** to enable **pcsd** to start at runtime. 2. On one node in the cluster, run **pcs cluster start --all** 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.
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**: * 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.	Pacemaker supports the following features for a cluster: * 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, see Table 10.1, “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.

Configuration Component	rgmanager	Pacemaker
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 **--getconf** 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 see Chapter 5, *Configuring Cluster Resources*.

Configuration Component	rgmanager	Pacemaker
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, see Chapter 5, *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 cannot 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, see Chapter 8, *Advanced Resource types*.

Configuration Component	rgmanager	Pacemaker
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** see Table 4.1, "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.

A.2. CLUSTER CREATION WITH PACEMAKER IN RED HAT ENTERPRISE LINUX RELEASE 6.5 AND RED HAT ENTERPRISE LINUX RELEASE 6.6 (AND LATER)

To create a Pacemaker cluster in Red Hat Enterprise Linux 6.5, you must create the cluster and start the cluster services on each node in the cluster. For example, to create a cluster named **my_cluster** that consists of nodes **z1-rhel65.example.com** and **z2-rhel65.example.com** and start cluster services on those nodes, run the following commands from both **z1-rhel65.example.com** and **z2-rhel65.example.com**.

```
[root@z1-rhel65]# pcs cluster setup --name my_cluster \
z1-rhel65.example.com z2-rhel65.example.com
[root@z1-rhel65]# pcs cluster start
```

```
[root@z2-rhel65]# pcs cluster setup --name my_cluster \
z1-rhel65.example.com z2-rhel65.example.com
[root@z2-rhel65]# pcs cluster start
```

In Red Hat Enterprise Linux 6.6 and later, you run the cluster creation command from one node of the cluster. The following command, run from one node only, creates the cluster named **my_cluster** that consists of nodes **z1-rhel66.example.com** and **z2-rhel66.example.com** and starts cluster services on those nodes.

```
[root@z1-rhel66]# pcs cluster setup --start --name my_cluster \
z1-rhel66.example.com z2-rhel66.example.com
```

APPENDIX B. CONFIGURATION EXAMPLE USING PCS COMMANDS

This appendix provides a step-by-step procedure for configuring a two-node Red Hat Enterprise Linux High Availability Add-On cluster, using the **pcs** command, in Red Hat Enterprise Linux release 6.6 and later. It also describes how to configure an Apache HTTP server in this cluster.

Configuring the cluster provided in this chapter requires that your system include the following components:

- 2 nodes, which will be used to create the cluster. In this example, the nodes used are **z1.example.com** and **z2.example.com**.
- Network switches for the private network, required for communication among the cluster nodes and other cluster hardware such as network power switches and Fibre Channel switches.
- A power fencing device for each node of the cluster. This example uses two ports of the APC power switch with a host name of **zapc.example.com**.

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 **pcs**, **pacemaker**, **cman**, and **fence-agents** are installed.

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

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

```
# chkconfig corosync off
# chkconfig cman off
```

B.1.2. Creating and Starting the Cluster

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

1. In order to use **pcs** to configure the cluster and communicate among the nodes, you must set a password on each node for the user ID **hacluster**, which is the **pcs** administration account. It is recommended that the password for user **hacluster** be the same on each node.

```
# passwd hacluster
Changing password for user hacluster.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

2. Before the cluster can be configured, the **pcsd** daemon must be started. This daemon works with the **pcs** command to manage configuration across the nodes in the cluster.

   On each node in the cluster, execute the following commands to start the **pcsd** service and to enable **pcsd** at system start.

   ```
   # service pcsd start
   # chkconfig pcsd on
   ```

3. Authenticate the **pcs** user **hacluster** for each node in the cluster on the node from which you will be running **pcs**.

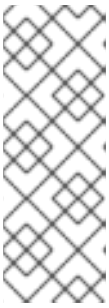
   The following command authenticates user **hacluster** on **z1.example.com** for both of the nodes in the example two-node cluster, **z1.example.com** and **z2.example.com**.

   ```
   root@z1 ~]# pcs cluster auth z1.example.com z2.example.com
   Username: hacluster
   Password:
   z1.example.com: Authorized
   z2.example.com: Authorized
   ```

4. Execute the following command from **z1.example.com** to create the two-node cluster **mycluster** that consists of nodes **z1.example.com** and **z2.example.com**. This will propagate the cluster configuration files to both nodes in the cluster. This command includes the **--start** option, which will start the cluster services on both nodes in the cluster.

   ```
   [root@z1 ~]# pcs cluster setup --start --name my_cluster \
   z1.example.com z2.example.com
   z1.example.com: Succeeded
   z1.example.com: Starting Cluster...
   z2.example.com: Succeeded
   z2.example.com: Starting Cluster...
   ```

5. Optionally, you can enable the cluster services to run on each node in the cluster when the node is booted.

   NOTE

   For your particular environment, you may choose to leave the cluster services disabled by skipping this step. This allows you to ensure that if a node goes down, any issues with your cluster or your resources are resolved before the node rejoins the cluster. If you leave the cluster services disabled, you will need to manually start the services when you reboot a node by executing the **pcs cluster start** command on that node.

   ```
   # pcs cluster enable --all
   ```

You can display the current status of the cluster with the **pcs cluster status** command. Because there may be a slight delay before the cluster is up and running when you start the cluster services with the **--start** option of the **pcs cluster setup** command, you should ensure that the cluster is up and running before performing any subsequent actions on the cluster and its configuration.

```
[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.2. FENCING CONFIGURATION

You must configure a fencing device for each node in the cluster. For general information about configuring fencing devices, see Chapter 4, *Fencing: Configuring STONITH*.

NOTE

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

This example uses the APC power switch with a host name of **zapc.example.com** to fence the nodes, and it uses the **fence_apc_snmp** fencing agent. Because both nodes will be fenced by the same fencing agent, you can configure both fencing devices as a single resource, using the **pcmk_host_map** and **pcmk_host_list** options.

You create a fencing device by configuring the device as a **stonith** resource with the **pcs stonith create** command. The following command configures a **stonith** resource named **myapc** that uses the **fence_apc_snmp** fencing agent for nodes **z1.example.com** and **z2.example.com**. The **pcmk_host_map** option maps **z1.example.com** to port 1, and **z2.example.com** to port 2. The login value and password for the APC device are both **apc**. By default, this device will use a monitor interval of 60s for each node.

Note that you can use an IP address when specifying the host name for the nodes.

```
[root@z1 ~]# pcs stonith create myapc fence_apc_snmp \
ipaddr="zapc.example.com"
pcmk_host_map="z1.example.com:1;z2.example.com:2" \
pcmk_host_check="static-list"
pcmk_host_list="z1.example.com,z2.example.com" \
login="apc" passwd="apc"
```

NOTE

When you create a **fence_apc_snmp stonith** device, you may see the following warning message, which you can safely ignore:

```
Warning: missing required option(s): 'port, action' for
resource type: stonith:fence_apc_snmp
```

The following command displays the parameters of an existing STONITH device.

```
[root@rh7-1 ~]# pcs stonith show myapc
 Resource: myapc (class=stonith type=fence_apc_snmp)
  Attributes: ipaddr=zapc.example.com
pcmk_host_map=z1.example.com:1;z2.example.com:2 pcmk_host_check=static-
list pcmk_host_list=z1.example.com,z2.example.com login=apc passwd=apc
  Operations: monitor interval=60s (myapc-monitor-interval-60s)
```

B.3. CONFIGURING AN APACHE HTTP SERVER IN A RED HAT HIGH AVAILABILITY CLUSTER WITH THE PCS COMMAND

This section describes how to configure an Apache HTTP server in a two-node Red Hat Enterprise Linux High Availability Add-On cluster using **pcs** to configure cluster resources. In this example case, clients access Apache through a floating IP address. The web server runs on one of two nodes in the cluster. If the node on which the web server is running becomes inoperative, the web server starts up again on the second node of the cluster with minimal service interruption.

This example requires that your system include the following components:

- A 2-node Red Hat High Availability cluster with power fencing configured for each node. This procedure uses the cluster example provided in Section B.1.2, “Creating and Starting the Cluster”.
- A public virtual IP address, required for Apache.
- Shared storage for the nodes in the cluster, using iSCSI or Fibre Channel.

The cluster is configured with an Apache resource group, which contains the cluster components that the web server requires: an LVM resource, a file system resource, an IP address resource, and a web server resource. This resource group can fail over from one node of the cluster to the other, allowing either node to run the web server. Before creating the resource group for this cluster, you will perform the following procedures:

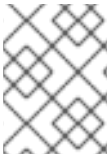
1. Configure an **ext4** file system mounted on the logical volume **my_lv**, as described in Section B.3.1, “Configuring an LVM Volume with an ext4 File System” .
2. Configure a web server, as described in Section B.3.2, “Web Server Configuration”.
3. Ensure that only the cluster is capable of activating the volume group that contains **my_lv**, and that the volume group will not be activated outside of the cluster on startup, as described in Section B.3.3, “Exclusive Activation of a Volume Group in a Cluster” .

After performing these procedures, you create the resource group and the resources it contains, as described in Section B.3.4, “Creating the Resources and Resource Groups with the pcs Command” .

B.3.1. Configuring an LVM Volume with an ext4 File System

This example requires that you create an LVM logical volume on storage that is shared between the nodes of the cluster.

The following procedure creates an LVM logical volume and then creates an **ext4** file system on that volume. In this example, the shared partition **/dev/sdb1** is used to store the LVM physical volume from which the LVM logical volume will be created.



NOTE

LVM volumes and the corresponding partitions and devices used by cluster nodes must be connected to the cluster nodes only.

Since the **/dev/sdb1** partition is storage that is shared, you perform this procedure on one node only,

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** command 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.3.2. Web Server Configuration

The following procedure configures an Apache HTTP server.

1. Ensure that the Apache HTTP server is installed on each node in the cluster. You also need the **wget** tool installed on the cluster to be able to check the status of Apache.

   On each node, execute the following command.

```
# yum install -y httpd wget
```

2. In order for the Apache resource agent to get the status of Apache, ensure that the following text is present in the **/etc/httpd/conf/httpd.conf** file on each node in the cluster, and

ensure that it has not been commented out. If this text is not already present, add the text to the end of the file.

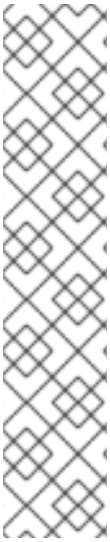
```
<Location /server-status>
  SetHandler server-status
  Order deny,allow
  Deny from all
  Allow from 127.0.0.1
</Location>
```

3. Create a web page for Apache to serve up. On one node in the cluster, mount the file system you created in Section B.3.1, "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 /var/www/
# mkdir /var/www/html
# mkdir /var/www/cgi-bin
# mkdir /var/www/error
# restorecon -R /var/www
# cat <<-END >/var/www/html/index.html
<html>
<body>Hello</body>
</html>
END
# umount /var/www
```

B.3.3. Exclusive Activation of a Volume Group in a Cluster

The following procedure configures 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 the volume group is activated by a system outside of the cluster, there is a risk of corrupting the volume group's metadata.



NOTE

You must ensure that the **lvmetad** daemon is disabled when using Pacemaker. You can check whether the daemon is disabled and whether any **lvmetad** processes are running by executing the following commands.

```
# grep use_lvmetad /etc/lvm/lvm.conf
use_lvmetad = 0
# ps -ef | grep -i [l]vm
root     23843 15478  0 11:31 pts/0    00:00:00 grep --
color=auto -i lvm
```

If your results differ from this example, set **use_lvmetad = 0** in the **/etc/lvm/lvm.conf** file and stop any running **lvmetad** processes.

This procedure modifies the **volume_list** entry in the **/etc/lvm/lvm.conf** configuration file. Volume groups listed in the **volume_list** entry are allowed to automatically activate on the local node outside of the cluster manager's control. Volume groups related to the node's local root and

home directories should be included in this list. All volume groups managed by the cluster manager must be excluded from the **volume_list** entry. Note that this procedure does not require the use of **clvmd**.

Perform the following procedure on each node in the cluster.

1. Determine which volume groups are currently configured on your local storage with the following command. This will output a list of the currently-configured volume groups. If you have space allocated in separate volume groups for root and for your home directory on this node, you will see those volumes in the output, as in this example.

```
# vgs --noheadings -o vg_name
  my_vg
  rhel_home
  rhel_root
```

2. Add the volume groups other than **my_vg** (the volume group you have just defined for the cluster) as entries to **volume_list** in the **/etc/lvm/lvm.conf** configuration file. For example, if you have space allocated in separate volume groups for root and for your home directory, you would uncomment the **volume_list** line of the **lvm.conf** file and add these volume groups as entries to **volume_list** as follows:

```
volume_list = [ "rhel_root", "rhel_home" ]
```

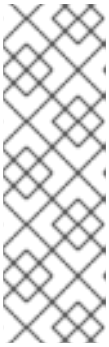
NOTE

If no local volume groups are present on a node to be activated outside of the cluster manager, you must still initialize the **volume_list** entry as **volume_list = []**.

3. Rebuild the **initramfs** boot image to guarantee that the boot image will not try to activate a volume group controlled by the cluster. Update the **initramfs** device with the following command. This command may take up to a minute to complete.

```
# dracut -H -f /boot/initramfs-$(uname -r).img $(uname -r)
```

4. Reboot the node.



NOTE

If you have installed a new Linux kernel since booting the node on which you created the boot image, the new **initrd** image will be for the kernel that was running when you created it and not for the new kernel that is running when you reboot the node. You can ensure that the correct **initrd** device is in use by running the **uname -r** command before and after the reboot to determine the kernel release that is running. If the releases are not the same, update the **initrd** file after rebooting with the new kernel and then reboot the node.

5. When the node has rebooted, check whether the cluster services have started up again on that node by executing the **pcs cluster status** command on that node. If this yields the message **Error: cluster is not currently running on this node** then enter the

following command.

```
# pcs cluster start
```

Alternately, you can wait until you have rebooted each node in the cluster and start cluster services on each of the nodes with the following command.

```
# pcs cluster start --all
```

B.3.4. Creating the Resources and Resource Groups with the pcs Command

This use case requires that you create four cluster resources. To ensure these resources all run on the same node, they are configured as part of the resource group **apachegroup**. The resources to create are as follows, listed in the order in which they will start.

1. An **LVM** resource named **my_lvm** that uses the LVM volume group you created in Section B.3.1, "Configuring an LVM Volume with an ext4 File System".
2. A **Filesystem** resource named **my_fs**, that uses the filesystem device **/dev/my_vg/my_lv** you created in Section B.3.1, "Configuring an LVM Volume with an ext4 File System" .
3. An **IPaddr2** resource, which is a floating IP address for the **apachegroup** resource group. The IP address must not be one already associated with a physical node. If the **IPaddr2** resource's NIC device is not specified, the floating IP must reside on the same network as the statically assigned IP addresses used by the cluster nodes, otherwise the NIC device to assign the floating IP address cannot be properly detected.
4. An **apache** resource named **Website** that uses the **index.html** file and the Apache configuration you defined in Section B.3.2, "Web Server Configuration".

The following procedure creates the resource group **apachegroup** and the resources that the group contains. The resources will start in the order in which you add them to the group, and they will stop in the reverse order in which they are added to the group. Run this procedure from one node of the cluster only.

1. The following command creates the LVM resource **my_lvm**. This command specifies the **exclusive=true** parameter to ensure that only the cluster is capable of activating the LVM logical volume. Because the resource group **apachegroup** does not yet exist, this command creates the resource group.

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

When you create a resource, the resource is started automatically. You can use the following command to confirm that the resource was created and has started.

```
# pcs resource show
 Resource Group: apachegroup
     my_lvm (ocf::heartbeat:LVM): Started
```

You can manually stop and start an individual resource with the **pcs resource disable** and **pcs resource enable** commands.

2. The following commands create the remaining resources for the configuration, adding them to the existing resource group **apachegroup**.

```
[root@z1 ~]# pcs resource create my_fs Filesystem \
device="/dev/my_vg/my_lv" directory="/var/www" fstype="ext4" --group
\
apachegroup

[root@z1 ~]# pcs resource create VirtualIP IPaddr2 ip=198.51.100.3 \
cidr_netmask=24 --group apachegroup

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

3. After creating the resources and the resource group that contains them, 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 (stonith:fence_apc_snmp): Started z1.example.com
 Resource Group: apachegroup
     my_lvm (ocf::heartbeat:LVM): Started z1.example.com
     my_fs (ocf::heartbeat:Filesystem): Started z1.example.com
     VirtualIP (ocf::heartbeat:IPaddr2): Started z1.example.com
     Website (ocf::heartbeat:apache): Started z1.example.com
```

Note that if you have not configured a fencing device for your cluster, as described in Section B.2, “Fencing Configuration”, by default the resources do not start.

4. 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
```

If you find that the resources you configured are not running, you can run the **pcs resource debug-start *resource*** command to test the resource configuration. For information on the **pcs resource debug-start** command, see the *High Availability Add-On Reference* manual.

B.3.5. Testing the Resource Configuration

In the cluster status display shown in Section B.3.4, "Creating the Resources and Resource Groups with the pcs Command", 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 using the following procedure to put the first node in **standby** mode, after which the node will no longer be able to host resources.

1. The following command puts node **z1.example.com** in **standby** mode.

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

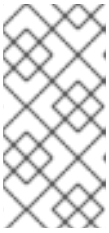
2. After putting node **z1** in standby mode, check the cluster status. Note that the resources should now all be running on **z2**.

```
[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 (stonith:fence_apc_snmp): Started z1.example.com
 Resource Group: apachegroup
     my_lvm (ocf::heartbeat:LVM): Started z2.example.com
     my_fs (ocf::heartbeat:Filesystem): Started z2.example.com
     VirtualIP (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.

3. To remove **z1** from **standby** mode, enter 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 the *Red Hat High Availability Add-On Reference*.

APPENDIX C. UPDATING SOFTWARE PACKAGES ON A RUNNING CLUSTER

With one of the primary responsibilities of a High Availability or Resilient Storage cluster being to provide continuous service for applications or resources, it is especially important that updates be applied in a systematic and consistent fashion to avoid any potential disruption to the availability of those critical services. This appendix provides the outline of procedure for a rolling cluster update.

WARNING

It is critical when performing software update procedures for Red Hat Enterprise Linux High Availability and Resilient Storage clusters to ensure that any node that will undergo updates is not an active member of the cluster before those updates are initiated. Swapping out the software that the cluster stack relies on while it is in use can lead to various problems and unexpected behaviors, including but not limited to issues that can cause complete outages of the cluster and services it is managing.

Performing a rolling update involves the following risks and considerations:

- When performing a rolling update, the presence of different versions of the High Availability and Resilient Storage packages within the same cluster introduces a risk that there may be unexpected behavior. The only way to completely eliminate this risk is to update the entire cluster by stopping the cluster software on all nodes, update those nodes by following this procedure, then start the cluster software again.

- New software versions always come with the potential for unexpected behavior, changes in functionality that may require advance preparation, or in rare cases, bugs causing that could impact the operation of the product. Red Hat strongly recommends having a test, development, or staging cluster configured identically to any production clusters, and using such a cluster to roll out any updates to first for thorough testing prior to the roll-out in production.

- Performing a rolling update necessarily means reducing the overall capacity and redundancy within the cluster. The size of the cluster dictates whether the absence of a single node poses a significant risk, with larger clusters being able to absorb more node failures before reaching the critical limit, and with smaller clusters being less capable or not capable at all of withstanding the failure of another node while one is missing. It is important that the potential for failure of additional nodes during the update procedure be considered and accounted for. If at all possible, taking a complete outage and updating the cluster entirely may be the preferred option so as to not leave the cluster operating in a state where additional failures could lead to an unexpected outage.

Perform the following steps to update the base Red Hat Enterprise Linux packages, High Availability Add-On packages, and Resilient Storage Add-On packages on each node in a rolling fashion.

1. Choose a single node where the software will be updated. If any preparations need to be made before stopping or moving the resources or software running on that node, carry out those steps now.

2. Move any managed resources off of this node as needed. If there are specific requirements or preferences for where resources should be relocated to, then consider creating new location constraints to place the resources on the correct node. The location of resources can be strategically chosen to result in the least number of moves throughout the rolling update procedure, rather than moving resources in preparation for every single node update. If allowing the cluster to automatically manage placement of resources on its own is acceptable, then the next step will automatically take care of this.

3. Place the chosen node in standby mode to ensure it is not considered in service, and to cause any remaining resources to be relocated elsewhere or stopped.

```
# pcs cluster standby node1.example.com
```

4. Stop the cluster software on the chosen node.

```
# pcs cluster stop node1.example.com
```

5. Perform any necessary software updates on the chosen node.

6. If any software was updated that necessitates a reboot, prepare to perform that reboot. It is recommended that cluster software be disabled from starting on boot so that the host can be checked to ensure it is fully functional on its new software versions before bringing it into the cluster. The following command disables the cluster from starting on boot.

```
# pcs cluster disable node1.example.com
```

   Perform the reboot when ready, and when the boot is complete, ensure that the host is fully functional and is using the correct software in any relevant areas (such as having booted into the latest kernel). If anything does not seem correct, then do not proceed until the situation is resolved.

   Once everything is set up correctly, re-enable the cluster software on this chosen node if it was previously enabled.

```
# pcs cluster enable node1.example.com
```

7. Start cluster services on the updated node so that the node will rejoin the cluster.

```
# pcs cluster start node1.example.com
```

   Check the output of the **pcs status** command to determine that appears as it should. Once the node is functioning properly, reactivate it for service by taking it out of standby mode.

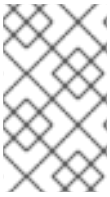
```
# pcs cluster unstandby node1.example.com
```

8. If any temporary location constraints were created to move managed resources off the node, adjust or remove the constraints to allow resources to return to their normally preferred locations.

9. Perform this entire procedure on each cluster node in turn.

APPENDIX D. CREATING NEW LOGICAL VOLUMES FOR AN EXISTING CLUSTER

The Red Hat High Availability Add-On provides support for high availability LVM volumes (HA-LVM) in a failover configuration.



NOTE

In Red Hat Enterprise Release 6 HA Cluster with Pacemaker, **clvmd** is not supported in combination with the LVM resource agent used by Pacemaker when using the volumes in an exclusive, active/passive manner.

To create new volumes, either volumes need to be added to a managed volume group on the node where it is already activated by the service, or the **volume_list** must be temporarily bypassed or overridden to allow for creation of the volumes until they can be prepared to be configured by a cluster resource.

To create a new logical volume on the same node where the LVM resource for that volume group is already started,m use the following procedure:

1. The volume group should already be tagged on the node owning that service, so simply create the volumes with a standard **lvcreate** command on that node.

   Determine the current owner of the relevant service.

   ```
   # pcs resource
   ```

   On the node where the service is started, create the logical volume.

   ```
   # lvcreate -l 100%FREE -n lv2 myVG
   ```

2. Add the volume into the service configuration in whatever way is necessary.

To create a new volume group entirely, use the following procedure.

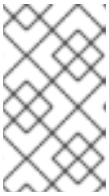
1. Create the volume group on one node with the tag **pacemaker** while overriding the **volume_list** to allow this tag. Specify any desired settings for this volume group as normal and specify the **--addtag** and **--config** options as in the following example:

   ```
   #  vgcreate myNewVG /dev/mapper/mpathb --addtag pacemaker --config
   'activation { volume_list = [ "@pacemaker" ] }'
   ```

2. Create logical volumes within this volume group as in the following example, otherwise perform any necessary administration on the volume group. This example creates a logical volume that is 1G in size. Create the filesystem as well on the logical volume if required.

   ```
   # lvcreate --addtag pacemaker -L +1G -n testlv1 myNewVG --config
   'activation { volume_list = [ "@pacemaker" ] }'
   Logical volume "testlv1" created
   ```

3. When the volume group activity is complete, deactivate the volume group and remove the tag, as in the following example.



NOTE

Use the same tag as you used previously, if you used a tag other than **pacemaker**. When you add the volume group as a resource in the cluster, all the logical volumes in that volume group will get activated on a single node.

```
# lvchange -an myNewVG/testlv1 --deltag pacemaker
Logical volume "testlv1" changed.
# vgchange -an myNewVG --deltag pacemaker
Volume group "myNewVG" successfully changed
0 logical volume(s) in volume group "myNewVG" now active
```

4. Create the volume group resource, as in the following example.

```
# pcs resource create mynewvg LVM volgrpname=myNewVG exclusive=true
```

APPENDIX E. REVISION HISTORY

Revision 5.0-9 **Thu Sep 14 2017** **Steven Levine**
Update to version for 6.9 GA publication.

Revision 5.0-3 **Wed Mar 8 2017** **Steven Levine**
Version for 6.9 GA publication.

Revision 5.0-2 **Fri Dec 16 2016** **Steven Levine**
Version for 6.9 Beta publication.

Revision 4.0-10 **Fri Jun 24 2016** **Steven Levine**
Republish for 6.8.

Revision 4.0-9 **Wed Apr 27 2016** **Steven Levine**
Preparing document for 6.8 GA Publication.

Revision 4.0-3 **Wed Mar 9 2016** **Steven Levine**
Initial Revision for Red Hat Enterprise Linux 6.8 Beta release

Revision 3.0-9 **Tue Sep 29 2015** **Steven Levine**
Republished version for Red Hat Enterprise Linux 6.7

Revision 3.0-6 **Wed Jul 8 2015** **Steven Levine**
Initial revision for Red Hat Enterprise Linux 6.7

Revision 3.0-4 **Fri Apr 17 2015** **Steven Levine**
Initial Revision for Red Hat Enterprise Linux 6.7 Beta release

Revision 2.0-5 **Thu Oct 9 2014** **Steven Levine**
Initial revision for Red Hat Enterprise Linux 6.6

Revision 2.0-2 **Wed Aug 7 2014** **Steven Levine**
Initial Revision for Red Hat Enterprise Linux 6.6 Beta release

Revision 1.1-2 **Wed Nov 20 2013** **Steven Levine**
Initial revision for Red Hat Enterprise Linux 6.5